Title: FEATURE-PRESERVING PROXY MESH GENERATION

Abstract: Techniques of generating proxy meshes include capturing intersections of an input mesh representation of a complex, three-dimensional system with a grid of voxels, including voxel edges, voxel faces, and voxel interiors. To generate a proxy mesh, a computer overlays the voxel grid on the input mesh. The computer then generates exterior intersections of the voxel grid with the input mesh on voxel grid edges and faces; the computer also generates interior intersections, i.e., mesh vertices within the interior of a voxel of the voxel grid. For each voxel of the voxel grid, the computer generates a convex hull of exterior and interior intersections. The computer then aggregates the convex hulls of voxels of the voxel grid to form the proxy mesh of the complex, three-dimensional system.
FEATURE-PRESERVING PROXY MESH GENERATION

TECHNICAL FIELD

[0001] This description relates to generating a proxy mesh for representing a three-dimensional object.

BACKGROUND

[0002] Some proxy meshes replace an assembly of three-dimensional (3D) models for visualizing a complex assembly of 3D models with a single 3D mesh that approximates the shape of the assembly. For example, a CAD model of a datacenter may have every component, even a single fastener, as a separate 3D model. Additionally, some proxy meshes may be used for collision detection and/or physics simulation.

SUMMARY

[0003] Implementations described herein are related to generating proxy meshes for representing complex, three-dimensional systems such as a data center when such a system has small features (e.g., screws, wires). Other such complex, three-dimensional systems include a person having hair follicles and buildings having small exterior features. To generate a proxy mesh, a voxel grid is overlaid on a three-dimensional input mesh representation of a complex, three-dimensional system. Exterior intersections of the voxel grid with the input mesh are generated on voxel grid edges and faces; interior intersections, i.e., mesh vertices within the interior of a voxel of the voxel grid, are also generated. For each voxel of the voxel grid, a convex hull of exterior and interior intersections is generated. The convex hulls of voxels of the voxel grid are aggregated to form the proxy mesh of the complex, three-dimensional system.

[0004] In one general aspect, a method can include receiving input mesh data representing a mesh for a three-dimensional object, the mesh including a plurality of vertices, a plurality of edges, and a plurality of faces. The method can also include, for each of a subset of a grid of voxels, determining a respective set of intersection points from at least one intersection between that voxel and the mesh; and generating a convex hull of the respective set of intersection points for that voxel. The method
can further include aggregating the respective convex hulls for the subset of the grid of voxels to produce a proxy mesh of the three-dimensional object.

[0005] In another general aspect, a computer program product comprises a non-transitory storage medium, the computer program product including code that, when executed by processing circuitry of a computing device, causes the processing circuitry to perform a method. The method can include receiving input mesh data representing a mesh for a three-dimensional object, the mesh including a plurality of vertices, a plurality of edges, and a plurality of faces. The method can also include, for each of a subset of a grid of voxels, determining a respective set of intersection points from at least one intersection between that voxel and the mesh; and generating a convex hull of the respective set of intersection points for that voxel. The method can further include aggregating the respective convex hulls for the subset of the grid of voxels to produce a proxy mesh of the three-dimensional object.

[0006] In another general aspect, an electronic apparatus comprises memory and controlling circuitry coupled to the memory. The controlling circuitry can be configured to receive input mesh data representing a mesh for a three-dimensional object, the mesh including a plurality of vertices, a plurality of edges, and a plurality of faces. The controlling circuitry can also be configured to, for each of a subset of a grid of voxels, determine a respective set of intersection points from at least one intersection between that voxel and the mesh; and generate a convex hull of the respective set of intersection points for that voxel. The controlling circuitry can further be configured to aggregate the respective convex hulls for the subset of the grid of voxels to produce a proxy mesh of the three-dimensional object.

[0007] The details of one or more implementations are set forth in the accompanying drawings and the description below. Other features will be apparent from the description and drawings, and from the claims.

**BRIEF DESCRIPTION OF THE DRAWINGS**

[0008] FIG. 1 is a diagram that illustrates an example electronic environment in which improved techniques described herein may be implemented.

[0009] FIG. 2A is a diagram that illustrates example locations of intersection points from a cylinder with voxel edges, faces, and interior.

[0010] FIG. 2B is a diagram that illustrates an example convex hull for the voxel derived from the intersection points.
[0011] FIG. 2C is a diagram that illustrates an example aggregation of convex hulls from different voxels in a voxel grid.

[0012] FIG. 2D is a diagram that illustrates an example proxy mesh resulting from the aggregation of convex hulls.

[0013] FIG. 3A is a diagram that illustrates example locations of intersection points from a narrow component with voxel edges, faces, and interior.

[0014] FIG. 3B is a diagram that illustrates an example convex hull for the voxel derived from the intersection points.

[0015] FIG. 3C is a diagram that illustrates an example proxy mesh resulting from the convex hull.

[0016] FIGs 4A, 4B, and 4C are diagrams that illustrate proxy meshes of an object performed by conventional approaches and the improved technique, at different voxel grid resolutions.

[0017] FIG. 5 is a flow chart that illustrates an example method of generating a proxy mesh for representing a three-dimensional object, according to disclosed implementations.

[0018] FIG. 6 is a diagram that illustrates an example of a computer device and a mobile computer device that can be used to implement the described techniques.

**DETAILED DESCRIPTION**

[0019] The complexity of a proxy mesh is generally lower compared to models approximated by the proxy mesh. This allows complex scenes to be visualized on low-end devices with limited resources. A common use for proxy meshes is in a level-of-detail framework for visualizing complex assets at a distance from where the visual artifacts inherent to proxy meshes are less noticeable.

[0020] Many approaches to generating proxy meshes involve operating on voxels. Voxels are stored in a uniform 3D grid and each voxel can store certain features (e.g., associated component model identifier, mesh information). A conventional approach to generating proxy meshes involves storing an oriented distance of a voxel to its nearest surface. In this conventional approach, one may reconstruct the mesh finding the iso-surface between neighboring voxels.

[0021] A technical problem with the above-described conventional approaches to generating proxy meshes is that such approaches are unable to capture any features smaller than the size of the voxel. In fact, most scene components smaller than the
voxel size would not be preserved at all and they would not be represented in the proxy mesh. This can pose considerable issues for CAD models in which many parts are narrow components (e.g., beams, ladders, etc.) which would require extremely small voxel sizes to be processed properly, leading to unacceptable memory and performance problems.

[0022] In accordance with the implementations described herein, a technical solution to the above-described technical problem includes capturing intersections of an input mesh representation of a complex, three-dimensional system with a grid of voxels, including voxel edges, voxel faces, and voxel interiors. To generate a proxy mesh, a computer overlays the voxel grid on the input mesh. The computer then generates exterior intersections of the voxel grid with the input mesh on voxel grid edges and faces; the computer also generates interior intersections, i.e., mesh vertices within the interior of a voxel of the voxel grid. For each voxel of the voxel grid, the computer generates a convex hull of exterior and interior intersections. The computer then aggregates the convex hulls of voxels of the voxel grid to form the proxy mesh of the complex, three-dimensional system.

[0023] In contrast to the conventional approaches to generating proxy meshes, the improved technique preserves all features of an input scene including narrow wire-like structures (e.g., cables, pipes) without resorting to an excessive resolution. Moreover, proxy meshes generated according to the improved technique have a higher visual quality than those generated using the conventional approaches.

[0024] In some implementations, each voxel of the subset includes a respective interior and a respective boundary. In such implementations, determining the respective set of intersection points includes identifying points on the boundary of that voxel through which an edge of the plurality of edges or a vertex of the plurality of vertices of the mesh intersect the boundary of that voxel.

[0025] In some implementations, each voxel of the subset includes a respective interior and a respective boundary. In such implementations, determining the respective set of intersection points includes identifying vertices of the plurality of vertices of the mesh located in the interior of the voxel, the identified vertices being included in the respective set of intersection points for that voxel.

[0026] In some implementations, each convex hull for the voxel includes a set of triangles, each of the set of triangles separating an interior volume of the proxy mesh from an exterior volume of the proxy mesh. In such implementations,
aggregating the respective convex hulls for the subset of the grid of voxels includes adding each of the set of triangles to the proxy mesh.

[0027] In some implementations, aggregating the respective convex hulls for the grid of voxels includes merging the convex hulls of each of the respective convex hulls for the subset of the grid of voxels such that duplicate vertices and edges are deleted.

[0028] In some implementations, each voxel of the subset includes a respective interior and a respective boundary and each convex hull for the voxel includes a set of triangles. In such implementations, aggregating the respective convex hulls for the subset of the grid of voxels includes, for each voxel of the subset of the grid of voxels, identifying a triangle of the set of triangles of the convex hull for that voxel not embedded in the boundary of that voxel; and adding the identified triangle to the proxy mesh.

[0029] A technical advantage of disclosed implementations is that, in contrast to the conventional approaches to generating proxy meshes, the improved technique preserves all, or substantially all, features of an input scene including narrow wire-like structures (e.g., cables, pipes) without resorting to an excessive resolution. Moreover, proxy meshes generated according to the improved technique have a higher visual quality than those generated using the conventional approaches.

[0030] FIG. 1 is a diagram that illustrates an example electronic environment 100 in which the above-described technical solution may be implemented. The computer 120 is configured to train and operate a prediction engine configured to estimate surface normal and reflectance from image data.

[0031] The computer 120 includes a network interface 122, one or more processing units 124, memory 126, and a display interface 128. The network interface 122 includes, for example, Ethernet adaptors, Token Ring adaptors, and the like, for converting electronic and/or optical signals received from the network to electronic form for use by the computer 120. The set of processing units 124 include one or more processing chips and/or assemblies. The memory 126 includes both volatile memory (e.g., RAM) and non-volatile memory, such as one or more ROMs, disk drives, solid state drives, and the like. The set of processing units 124 and the memory 126 together form control circuitry, which is configured and arranged to carry out various methods and functions as described herein. The display interface
128 is configured to provide data to a display device 190 for rendering and display to a user.

[0032] In some implementations, one or more of the components of the computer 120 can be, or can include processors (e.g., processing units 124) configured to process instructions stored in the memory 126. Examples of such instructions as depicted in FIG. 1 include a mesh acquisition manager 130, a voxel manager 140, an intersection engine manager 150, a convex hull manager 160, and an aggregation manager 170. Further, as illustrated in FIG. 1, the memory 126 is configured to store various data, which is described with respect to the respective managers that use such data.

[0033] The mesh acquisition manager 130 is configured to receive mesh data 132. In some implementations, the mesh acquisition manager 130 receives the mesh data 132 over the network interface 122, i.e., over a network. In some implementations, the mesh acquisition manager 130 receives the mesh data 132 from local storage (e.g., a disk drive, flash drive, SSD, or the like).

[0034] The mesh data 132 represents the three-dimensional mesh for a given object. For example, in some implementations the object is a complex system such as a data center having many large and small components. As shown in FIG. 1, the mesh data include vertex data 134, edge data 136, and face data 138.

[0035] In some implementations, each of the vertices of the vertex data 134 includes an ordered triplet representing a point in three-dimensional space. In some implementations, each component of the ordered triplet is quantized, i.e., represented by a bit string of a specified length. Further, the points may be represented in more than three dimensions, or two dimensions. In some implementations, the vertex data 134 also includes information such as texture coordinates and normal vectors. In some implementations, the vertex data 134 also includes a vertex identifier for each vertex. In some implementations, the edge data 136 includes an edge identifier of an edge and vertex identifiers of a pair of vertices through which the edge terminates. In some implementations, the face data 138 includes a face identifier of a triangular face. In some implementations, the face data 138 also includes edge identifiers of edges of each face and/or vertex identifiers of vertices at the corners of the face.

[0036] The voxel manager 140 is configured to generate voxel data 142 representing a grid of voxels overlaid over the mesh represented by the mesh data 132. In some implementations, the voxel manager 140 generates a grid of voxels in
response to input by a user. In some implementations, the input from the user includes a voxel dimension, a grid dimension or number of voxels, and a corner vertex of the grid. In some implementations, the input from the user includes a voxel dimension; in such implementations, the voxel manager 140 is configured to position the grid of voxels with respect to the input mesh automatically. In some implementations, the voxel manager is configured to determine the size of the grid of voxels given the dimensions of the input mesh.

[0037] The voxel data 142 represents a grid of voxels overlaid over an input mesh represented by the mesh data 132. As shown in FIG. 1, the voxel data 142 includes voxel grid data 144 and voxel size data 146. In some implementations, the voxel grid data 144 represents a size of the grid of voxels. In some implementations, the size of the grid is expressed in terms of a dimension of the grid. In some implementations, the size of the grid is expressed in terms of a number of voxels in the grid. In some implementations, the voxel grid data 144 represents a grid other than a cubic form, e.g., a regular polyhedron such as a tetrahedron, an icosahedron, or the like.

[0038] In some implementations, the voxel size data 146 represents a dimension of a single voxel, e.g., a length of an edge of a cubic voxel. In such an implementation, the grid data 144 represents a number of voxels. In some implementations, the voxel size data 146 represents a number of voxels in the grid of voxels. In such an implementation, the voxel grid data 144 represents a grid dimension.

[0039] The intersection manager 150 is configured to determine the intersections of the voxel grid represented by the voxel data 142 with the input mesh represented by the mesh data 132 to produce intersection data 152. For example, the intersection manager 150 is configured to determine whether any edges or faces of the grid of voxels intersect with any edges or faces of the input mesh to produce voxel boundary intersection data 154. Further, the intersection manager 150 is configured to determine vertices of the input mesh that are located in an interior of a voxel to produce voxel interior intersection data 156. In some implementations, the intersections are determined as follows:

- For each edge of the mesh find all voxel faces that the edge intersects (if any) are found and a point is created for each intersection. This can be done efficiently because it may be assumed that voxels are regular; in this case, one
may easily limit the number of intersection tests of voxel face/edge based on the coordinates of the edge end points.

- For each triangle of the mesh all voxel edges that intersect the voxel are found. As above, the number of intersection tests can be limited because the regular structure of the voxel grid. In some implementations, only voxel edges around the triangle are tested.

[0040] The computation of the intersections of the input mesh with the grid of voxels is equivalent to a voxelization of the input mesh. It is noted that each intersection with a voxel grid edge is shared between four neighboring voxels when the grid has a cubic geometry. In such a case, intersections with voxel grid faces are shared between two neighboring voxels.

[0041] The convex hull manager 160 is configured to determine a convex hull of intersection points represented by the voxel boundary intersection data 154 and the voxel interior intersection data 156 to produce convex hull data 162. A convex hull of a set of points is defined as the smallest convex shape that encloses the points. The convex hull manager 160 is configured to determine the convex hull of intersections in and on a voxel using standard techniques.

[0042] The convex hull data 162 represents the convex hull of the intersections represented by the intersection data 152. In some implementations, the convex hull data 162 includes triangle data 164, which includes triplets of points defining triangles of the convex hull. In some implementations, the convex hull data 162 takes the form of a triplet of points representing a first triangle, and then pairs of points representing adjacent triangles. In some implementations, the convex hull data 162 contains an array of points with each row of the array representing an element of the convex hull.

[0043] The aggregation manager 170 is configured to aggregate the convex hull data 162 for each voxel of the grid of voxels to produce proxy mesh data 176. As shown in FIG. 1, the aggregation manager 170 includes a deduplication manager 172 and a boundary embedding manager 174.

[0044] The aggregation manager 170 forms the proxy mesh represented by proxy mesh data 176 by independently extracting a mesh surface for each voxel from the convex hull data 162. Only faces that are connected to an “exterior” of the volume are used from the convex hull for a voxel. To accomplish this, the boundary
embedding manager 174 is configured to extract faces not aligned with voxel boundaries; i.e., convex hull triangles not embedded in the faces of a voxel. The aggregation manager 170 then adds such triangles into a voxel mesh for that voxel.

[0045] The aggregation manager 170 is also configured to merge all of the voxel meshes to form the proxy mesh represented by proxy mesh data 176. To accomplish this, the deduplication manager 172 removes all duplicate vertices and edges on the shared boundaries of connected voxels.

[0046] The proxy mesh data 176 represents the proxy mesh 176 resulting from an aggregation of the voxel meshes. In some implementations, the proxy mesh data 176 takes the form of a list of triplets of three-dimensional points. In some implementations, the proxy mesh data 176 also includes representations of the vertices and edges removed during aggregation.

[0047] It is noted that, because edge and face points are shared between neighboring voxels, the above algorithm guarantees that the resulting mesh is going to be closed and well connected across voxels. Non-manifoldness of a proxy mesh is guaranteed only for input meshes that are non-manifold. Manifoldness implies that each edge is connected to two triangles and each vertex is connected to a single surface.

[0048] In some implementations, the processing units 124, via the display interface 190, transmits the proxy mesh data 176 to the display device 190. The display device 190 then renders the proxy mesh data 176 and displays the proxy mesh represented by the proxy mesh data 176. A user may view a graphical rendering of the proxy mesh 176 on the display device 190. The display device 190 would take significantly less time rendering the proxy mesh data 176 for display than the mesh represented by mesh data 132 from which the proxy mesh data 176 is derived.

[0049] In some implementations, the display device 190 is a part of a virtual reality or augmented reality (VR/AR) system. In that case, the VR/AR system may use the rendered proxy mesh data 176 to define occlusions and/or obstacles for physics simulation. For example, treating collisions of an object represented by mesh data 132 may require fast processing of the mesh data 132. In contrast, using the proxy mesh data 176 instead, a VR/AR system may generate fast and accurate physics simulation for collision detections.

[0050] In some implementations, the computer 120 is part of a graphics modeling system configured to display a model of a complex, three-dimensional
object. In some implementations, such a graphics modeling system enables a user to view the model from any perspective via rotation, translation, and scaling operations. Such operations may be performed significantly more quickly using a proxy mesh representation of the three-dimensional object. The improvements to the proxy mesh allow for similar rapid processing of a proxy mesh with more fine details.

[0051] The components (e.g., modules, processing units 124) of the user device 120 can be configured to operate based on one or more platforms (e.g., one or more similar or different platforms) that can include one or more types of hardware, software, firmware, operating systems, runtime libraries, and/or so forth. In some implementations, the components of the computer 120 can be configured to operate within a cluster of devices (e.g., a server farm). In such an implementation, the functionality and processing of the components of the computer 120 can be distributed to several devices of the cluster of devices.

[0052] The components of the computer 120 can be, or can include, any type of hardware and/or software configured to process attributes. In some implementations, one or more portions of the components shown in the components of the computer 120 in FIG. 1 can be, or can include, a hardware-based module (e.g., a digital signal processor (DSP), a field programmable gate array (FPGA), a memory), a firmware module, and/or a software-based module (e.g., a module of computer code, a set of computer-readable instructions that can be executed at a computer). For example, in some implementations, one or more portions of the components of the computer 120 can be, or can include, a software module configured for execution by at least one processor (not shown). In some implementations, the functionality of the components can be included in different modules and/or different components than those shown in FIG. 1, including combining functionality illustrated as two components into a single component.

[0053] Although not shown, in some implementations, the components of the computer 120 (or portions thereof) can be configured to operate within, for example, a data center (e.g., a cloud computing environment), a computer system, one or more server/host devices, and/or so forth. In some implementations, the components of the computer 120 (or portions thereof) can be configured to operate within a network. Thus, the components of the computer 120 (or portions thereof) can be configured to function within various types of network environments that can include one or more devices and/or one or more server devices. For example, the network can
be, or can include, a local area network (LAN), a wide area network (WAN), and/or so forth. The network can be, or can include, a wireless network and/or wireless network implemented using, for example, gateway devices, bridges, switches, and/or so forth. The network can include one or more segments and/or can have portions based on various protocols such as Internet Protocol (IP) and/or a proprietary protocol. The network can include at least a portion of the Internet.

[0054] In some implementations, one or more of the components of the computer 120 can be, or can include, processors configured to process instructions stored in a memory. For example, an mesh acquisition manager 130 (and/or a portion thereof), a voxel manager 140 (and/or a portion thereof), an intersection manager 150 (and/or a portion thereof), a convex hull manager 160 (and/or a portion thereof), and an aggregation manager 170 can be a combination of a processor and a memory configured to execute instructions related to a process to implement one or more functions.

[0055] In some implementations, the memory 126 can be any type of memory such as a random-access memory, a disk drive memory, flash memory, and/or so forth. In some implementations, the memory 126 can be implemented as more than one memory component (e.g., more than one RAM component or disk drive memory) associated with the components of the computer 120. In some implementations, the memory 126 can be a database memory. In some implementations, the memory 126 can be, or can include, a non-local memory. For example, the memory 126 can be, or can include, a memory shared by multiple devices (not shown). In some implementations, the memory 126 can be associated with a server device (not shown) within a network and configured to serve the components of the computer 120. As illustrated in FIG. 1, the memory 126 is configured to store various data, including mesh data 132, voxel data 142, intersection data 152, convex hull data 162, and proxy mesh data 176.

[0056] FIG. 2A is a diagram that illustrates example locations of intersection points from an intersection scenario 200 involving a cylindrical object 202 intersecting a voxel 204 with edges, faces, and interior. There are intersections of the vertices of the mesh representation of the object 202 with the voxel boundary 206 and interior 208.

[0057] FIG. 2B is a diagram that illustrates an example scenario 220 in which a convex hull 222 for the voxel 204 is derived from the intersection points 206. The
convex hull 222 for the voxel 204 is represented as a set of triangles between the intersection points. The convex hull for each voxel is generated independently.

[0058] FIG. 2C is a diagram that illustrates an example aggregation 250 of convex hulls from different voxels in a voxel grid 254. Because the surface of the object 202 is a simple, smooth cylinder, the aggregated convex hull (i.e., the aggregation of the voxel hulls) 252 has vertices that form a relatively smooth surface, despite being formed from intersections of the mesh with both the boundaries and the interiors of the voxels. It is noted, however, that triangles of the convex hull 222 embedded in a face of the voxel 204 are not added to the proxy mesh.

[0059] It is noted that, in aggregating the convex hulls for each voxel, that there will be duplicate intersection points. Such duplicates occur on shared voxel faces and edges in a grid. For example, duplicate point 256 lies on an edge shared by faces of adjacent voxels. Such duplicate points may be identified using, e.g., a hash function.

[0060] FIG. 2D is a diagram that illustrates an example proxy mesh 270 resulting from the aggregation of convex hulls. The proxy mesh 270 is generated by removing duplicate vertices and edges from the voxel hulls. In this case, the proxy mesh 270 is essentially a facsimile of the object 202. The duplicate vertices belong to triangles that are to be aggregated. Accordingly, once a duplicate vertex is removed, in some implementations, triangles including that vertex are merged with triangles including its duplicate.

[0061] FIG. 3A is a diagram that illustrates example locations of intersection points from an intersection scenario 300 involving an object 302 having a narrow component, with edges, faces, and interior of a voxel 304. There are intersections 306 of the vertices and edges of the mesh representation of the object 302 with the voxel boundary 306. Moreover, there are intersections 308 with the interior of the voxel 304; these are vertices of the mesh representation of the object 302 that are in the interior of the voxel 304.

[0062] FIG. 3B is a diagram that illustrates an example scenario 320 in which a convex hull 310 for the voxel 304 is derived from the intersection points 306 and 308. Although not shown in FIG. 3B, the convex hull 310 is represented as a set of triangles between the intersection points. The convex hull for each voxel is generated independently.

[0063] FIG. 3C is a diagram that illustrates an example proxy mesh 352
resulting from an aggregation of convex hulls from the voxel 304 and neighboring voxels. It is noted that the object 304 has a thin component, i.e., narrower than the length of a voxel edge or having a maximum cross-section smaller than a voxel face. Because the point grid captures all intersection points, it can effectively represent even very thin components. The convex hull 310 smooths out concave regions of the mesh but the final shape of the proxy mesh 352 still represents the object 302 accurately.

[0064] It is also noted that duplicates of the boundary intersections were removed as described above as part of the aggregation process. The triangles of the convex hulls including those duplicate points are merged together to form the proxy mesh 352.

[0065] FIGs 4A, 4B, and 4C are diagrams that illustrate proxy meshes of an object performed by conventional approaches and the improved technique, at different voxel grid resolutions. FIG. 4A is a diagram representing a complex, three-dimensional object 400 represented by an input mesh. FIG. 4B illustrates a comparison between a proxy mesh 410 generated using a $20 \times 20 \times 20$ voxel grid via a conventional technique (using a feature-sensitive Hermite grid) and a proxy grid 412 generated using a $20 \times 20 \times 20$ voxel grid via the improved techniques involving generating convex hulls of intersection points. FIG. 4C illustrates a comparison between a proxy mesh 420 generated using a $100 \times 100 \times 100$ voxel grid via the conventional technique and a proxy grid 422 generated using a $100 \times 100 \times 100$ voxel grid via the improved techniques involving generating convex hulls of intersection points.

[0066] It is clear from FIGs. 4A, 4B, and 4C that the improved techniques result in proxy meshes that are faithful renderings of the original objects, even with a modest grid at which a conventional technique produces a disjointed proxy mesh that has very little resemblance to the original object. Voxel hull remeshing can preserve convex components such as propeller blades even with a very low number of voxels while the conventional Hermite grid-based approach struggles to maintain the edges of the blades even when the voxel resolution is substantially increased.

[0067] FIG. 5 is a flow chart depicting an example method 500 of generating a proxy mesh from an input mesh representation of a complex object according to the above-described improved techniques. The method 500 may be performed by
software constructs described in connection with FIG. 1, which reside in memory 126 of the computer 120 and are run by the set of processing units 124.

[0068] At 502, the mesh acquisition manager 130 receives input mesh data (e.g., mesh data 132) representing a mesh for a complex three-dimensional object, the mesh including a plurality of vertices (e.g., vertex data 134), a plurality of edges (e.g., edge data 136), and a plurality of faces (e.g., face data 138), the complex three-dimensional object including at least one feature having a dimension smaller than a smallest dimension of a grid of voxels (e.g., voxel data 142) overlaid on the mesh.

[0069] At 504, the intersection manager 150, for each voxel of the grid of voxels, determines a respective set of intersection points (e.g., intersection data 152) from at least one intersection between that voxel and the mesh.

[0070] At 506, the convex hull manager 160, for each voxel of the grid of voxels, generates a convex hull (e.g., convex hull data) of the respective set of intersection points for that voxel.

[0071] At 508, the aggregation manager 170 aggregates the respective convex hulls for the grid of voxels to produce a proxy mesh (e.g., proxy mesh data 176) of the complex three-dimensional object.

[0072] FIG. 6 illustrates an example of a generic computer device 600 and a generic mobile computer device 650, which may be used with the techniques described here. Computer device 600 is one example configuration of computer 120 of FIG. 1 and FIG. 2.

[0073] As shown in FIG. 6, computing device 600 is intended to represent various forms of digital computers, such as laptops, desktops, workstations, personal digital assistants, servers, blade servers, mainframes, and other appropriate computers. Computing device 650 is intended to represent various forms of mobile devices, such as personal digital assistants, cellular telephones, smart phones, and other similar computing devices. The components shown here, their connections and relationships, and their functions, are meant to be exemplary only, and are not meant to limit implementations of the inventions described and/or claimed in this document.

[0074] Computing device 600 includes a processor 602, memory 604, a storage device 606, a high-speed interface 608 connecting to memory 604 and high-speed expansion ports 610, and a low speed interface 612 connecting to low speed bus 614 and storage device 606. Each of the components 602, 604, 606, 608, 610, and 612, are interconnected using various busses, and may be mounted on a common
motherboard or in other manners as appropriate. The processor 602 can process instructions for execution within the computing device 600, including instructions stored in the memory 604 or on the storage device 606 to display graphical information for a GUI on an external input/output device, such as display 616 coupled to high speed interface 608. In other implementations, multiple processors and/or multiple buses may be used, as appropriate, along with multiple memories and types of memory. Also, multiple computing devices 600 may be connected, with each device providing portions of the necessary operations (e.g., as a server bank, a group of blade servers, or a multi-processor system).

[0075] The memory 604 stores information within the computing device 600. In one implementation, the memory 604 is a volatile memory unit or units. In another implementation, the memory 604 is a non-volatile memory unit or units. The memory 604 may also be another form of computer-readable medium, such as a magnetic or optical disk.

[0076] The storage device 606 is capable of providing mass storage for the computing device 600. In one implementation, the storage device 606 may be or contain a computer-readable medium, such as a floppy disk device, a hard disk device, an optical disk device, or a tape device, a flash memory or other similar solid state memory device, or an array of devices, including devices in a storage area network or other configurations. A computer program product can be tangibly embodied in an information carrier. The computer program product may also contain instructions that, when executed, perform one or more methods, such as those described above. The information carrier is a computer- or machine-readable medium, such as the memory 604, the storage device 606, or memory on processor 602.

[0077] The high speed controller 608 manages bandwidth-intensive operations for the computing device 500, while the low speed controller 612 manages lower bandwidth-intensive operations. Such allocation of functions is exemplary only. In one implementation, the high-speed controller 608 is coupled to memory 604, display 616 (e.g., through a graphics processor or accelerator), and to high-speed expansion ports 610, which may accept various expansion cards (not shown). In the implementation, low-speed controller 612 is coupled to storage device 506 and low-speed expansion port 614. The low-speed expansion port, which may include various communication ports (e.g., USB, Bluetooth, Ethernet, wireless Ethernet) may be
coupled to one or more input/output devices, such as a keyboard, a pointing device, a scanner, or a networking device such as a switch or router, e.g., through a network adapter.

[0078] The computing device 600 may be implemented in a number of different forms, as shown in the figure. For example, it may be implemented as a standard server 620, or multiple times in a group of such servers. It may also be implemented as part of a rack server system 624. In addition, it may be implemented in a personal computer such as a laptop computer 622. Alternatively, components from computing device 600 may be combined with other components in a mobile device (not shown), such as device 650. Each of such devices may contain one or more of computing device 600, 650, and an entire system may be made up of multiple computing devices 600, 650 communicating with each other.

[0079] Computing device 650 includes a processor 652, memory 664, an input/output device such as a display 654, a communication interface 666, and a transceiver 668, among other components. The device 650 may also be provided with a storage device, such as a microdrive or other device, to provide additional storage. Each of the components 650, 652, 664, 654, 666, and 668, are interconnected using various buses, and several of the components may be mounted on a common motherboard or in other manners as appropriate.

[0080] The processor 652 can execute instructions within the computing device 450, including instructions stored in the memory 664. The processor may be implemented as a chipset of chips that include separate and multiple analog and digital processors. The processor may provide, for example, for coordination of the other components of the device 650, such as control of user interfaces, applications run by device 650, and wireless communication by device 650.

[0081] Processor 652 may communicate with a user through control interface 658 and display interface 656 coupled to a display 654. The display 654 may be, for example, a TFT LCD (Thin-Film-Transistor Liquid Crystal Display) or an OLED (Organic Light Emitting Diode) display, or other appropriate display technology. The display interface 656 may comprise appropriate circuitry for driving the display 654 to present graphical and other information to a user. The control interface 658 may receive commands from a user and convert them for submission to the processor 652. In addition, an external interface 662 may be provided in communication with processor 652, so as to enable near area communication of device 650 with other
devices. External interface 662 may provide, for example, for wired communication in some implementations, or for wireless communication in other implementations, and multiple interfaces may also be used.

[0082] The memory 664 stores information within the computing device 650. The memory 664 can be implemented as one or more of a computer-readable medium or media, a volatile memory unit or units, or a non-volatile memory unit or units. Expansion memory 674 may also be provided and connected to device 650 through expansion interface 672, which may include, for example, a SIMM (Single In Line Memory Module) card interface. Such expansion memory 674 may provide extra storage space for device 650, or may also store applications or other information for device 650. Specifically, expansion memory 674 may include instructions to carry out or supplement the processes described above, and may include secure information also. Thus, for example, expansion memory 674 may be provided as a security module for device 650, and may be programmed with instructions that permit secure use of device 650. In addition, secure applications may be provided via the SIMM cards, along with additional information, such as placing identifying information on the SIMM card in a non-hackable manner.

[0083] The memory may include, for example, flash memory and/or NVRAM memory, as discussed below. In one implementation, a computer program product is tangibly embodied in an information carrier. The computer program product contains instructions that, when executed, perform one or more methods, such as those described above. The information carrier is a computer- or machine-readable medium, such as the memory 664, expansion memory 674, or memory on processor 652, that may be received, for example, over transceiver 668 or external interface 662.

[0084] Device 650 may communicate wirelessly through communication interface 666, which may include digital signal processing circuitry where necessary. Communication interface 666 may provide for communications under various modes or protocols, such as GSM voice calls, SMS, EMS, or MMS messaging, CDMA, TDMA, PDC, WCDMA, CDMA2000, or GPRS, among others. Such communication may occur, for example, through radio-frequency transceiver 668. In addition, short-range communication may occur, such as using a Bluetooth, WiFi, or other such transceiver (not shown). In addition, GPS (Global Positioning System) receiver module 670 may provide additional navigation- and location-related wireless data to device 650, which may be used as appropriate by applications running on device 650.
Device 650 may also communicate audibly using audio codec 660, which may receive spoken information from a user and convert it to usable digital information. Audio codec 660 may likewise generate audible sound for a user, such as through a speaker, e.g., in a handset of device 650. Such sound may include sound from voice telephone calls, may include recorded sound (e.g., voice messages, music files, etc.) and may also include sound generated by applications operating on device 650.

The computing device 650 may be implemented in a number of different forms, as shown in the figure. For example, it may be implemented as a cellular telephone 680. It may also be implemented as part of a smart phone 682, personal digital assistant, or other similar mobile device.

Various implementations of the systems and techniques described here can be realized in digital electronic circuitry, integrated circuitry, specially designed ASICs (application specific integrated circuits), computer hardware, firmware, software, and/or combinations thereof. These various implementations can include implementation in one or more computer programs that are executable and/or interpretable on a programmable system including at least one programmable processor, which may be special or general purpose, coupled to receive data and instructions from, and to transmit data and instructions to, a storage system, at least one input device, and at least one output device.

These computer programs (also known as programs, software, software applications or code) include machine instructions for a programmable processor and can be implemented in a high-level procedural and/or object-oriented programming language, and/or in assembly/machine language. As used herein, the terms “machine-readable medium” “computer-readable medium” refers to any computer program product, apparatus and/or device (e.g., magnetic discs, optical disks, memory, Programmable Logic Devices (PLDs)) used to provide machine instructions and/or data to a programmable processor, including a machine-readable medium that receives machine instructions as a machine-readable signal. The term “machine-readable signal” refers to any signal used to provide machine instructions and/or data to a programmable processor.

To provide for interaction with a user, the systems and techniques described here can be implemented on a computer having a display device (e.g., a CRT (cathode ray tube) or LCD (liquid crystal display) monitor) for displaying
information to the user and a keyboard and a pointing device (e.g., a mouse or a trackball) by which the user can provide input to the computer. Other kinds of devices can be used to provide for interaction with a user as well; for example, feedback provided to the user can be any form of sensory feedback (e.g., visual feedback, auditory feedback, or tactile feedback); and input from the user can be received in any form, including acoustic, speech, or tactile input.

[0090] The systems and techniques described here can be implemented in a computing system that includes a back end component (e.g., as a data server), or that includes a middleware component (e.g., an application server), or that includes a front end component (e.g., a client computer having a graphical user interface or a Web browser through which a user can interact with an implementation of the systems and techniques described here), or any combination of such back end, middleware, or front end components. The components of the system can be interconnected by any form or medium of digital data communication (e.g., a communication network). Examples of communication networks include a local area network (“LAN”), a wide area network (“WAN”), and the Internet.

[0091] The computing system can include clients and servers. A client and server are generally remote from each other and typically interact through a communication network. The relationship of client and server arises by virtue of computer programs running on the respective computers and having a client-server relationship to each other.

[0092] A number of implementations have been described. Nevertheless, it will be understood that various modifications may be made without departing from the spirit and scope of the specification.

[0093] It will also be understood that when an element is referred to as being on, connected to, electrically connected to, coupled to, or electrically coupled to another element, it may be directly on, connected or coupled to the other element, or one or more intervening elements may be present. In contrast, when an element is referred to as being directly on, directly connected to or directly coupled to another element, there are no intervening elements present. Although the terms directly on, directly connected to, or directly coupled to may not be used throughout the detailed description, elements that are shown as being directly on, directly connected or directly coupled can be referred to as such. The claims of the application may be
amended to recite exemplary relationships described in the specification or shown in the figures.

[0094] While certain features of the described implementations have been illustrated as described herein, many modifications, substitutions, changes and equivalents will now occur to those skilled in the art. It is, therefore, to be understood that the appended claims are intended to cover all such modifications and changes as fall within the scope of the implementations. It should be understood that they have been presented by way of example only, not limitation, and various changes in form and details may be made. Any portion of the apparatus and/or methods described herein may be combined in any combination, except mutually exclusive combinations. The implementations described herein can include various combinations and/or sub-combinations of the functions, components and/or features of the different implementations described.

[0095] In addition, the logic flows depicted in the figures do not require the particular order shown, or sequential order, to achieve desirable results. In addition, other steps may be provided, or steps may be eliminated, from the described flows, and other components may be added to, or removed from, the described systems. Accordingly, other implementations are within the scope of the following claims.
WHAT IS CLAIMED IS:

1. A method comprising:
   receiving input mesh data representing a mesh for a three-dimensional object, the mesh including a plurality of vertices, a plurality of edges, and a plurality of faces;
   for each of a subset of a grid of voxels,
   determining a respective set of intersection points from at least one intersection between that voxel and the mesh; and
   generating a convex hull of the respective set of intersection points for that voxel; and
   aggregating the respective convex hulls for the subset of the grid of voxels to produce a proxy mesh of the three-dimensional object.

2. The method as in claim 1, wherein each voxel of the subset includes a respective interior and a respective boundary, and
   wherein determining the respective set of intersection points includes:
   identifying points on the boundary of that voxel through which an edge of the plurality of edges or a vertex of the plurality of vertices of the mesh intersect the boundary of that voxel.

3. The method as in claim 1, wherein each voxel of the subset includes a respective interior and a respective boundary, and
   wherein determining the respective set of intersection points includes:
   identifying vertices of the plurality of vertices of the mesh located in the interior of that voxel, the identified vertices being included in the respective set of intersection points for that voxel.

4. The method as in claim 1, wherein each convex hull for the voxel includes a set of triangles, each of the set of triangles separating an interior volume of the proxy mesh from an exterior volume of the proxy mesh, and
   wherein aggregating the respective convex hulls for the subset of the grid of voxels includes:
   adding each of the set of triangles to the proxy mesh.
5. The method as in claim 1, wherein aggregating the respective convex hulls for the subset of the grid of voxels includes:
   merging the convex hulls of each of the respective convex hulls for the grid of voxels such that duplicate vertices and edges are deleted.

6. The method as in claim 1, wherein each voxel of the subset includes a respective interior and a respective boundary,
   wherein each convex hull for the voxel includes a set of triangles, and
   wherein aggregating the respective convex hulls for the subset of the grid of voxels includes:
   for each voxel of the subset of the grid of voxels, identifying a triangle of the set of triangles of the convex hull for that voxel not embedded in the boundary of that voxel; and
   adding the identified triangle to the proxy mesh.

7. A computer program product comprising a nontransitive storage medium, the computer program product including code that, when executed by processing circuitry, causes the processing circuitry to perform a method, the method comprising:
   receiving input mesh data representing a mesh for a three-dimensional object, the mesh including a plurality of vertices, a plurality of edges, and a plurality of faces;
   for each of a subset of a grid of voxels,
     determining a respective set of intersection points from at least one intersection between that voxel and the mesh; and
   generating a convex hull of the respective set of intersection points for that voxel; and
   aggregating the respective convex hulls for the subset of the grid of voxels to produce a proxy mesh of the three-dimensional object.

8. The computer program product as in claim 7, wherein each voxel of the subset includes a respective interior and a respective boundary, and
   wherein determining the respective set of intersection points includes:
identifying points on the boundary of that voxel through which
an edge of the plurality of edges or a vertex of the plurality of vertices
of the mesh intersect the boundary of that voxel.

9. The computer program product as in claim 7, wherein each voxel of the subset
includes a respective interior and a respective boundary, and
wherein determining the respective set of intersection points includes:
identifying vertices of the plurality of vertices of the mesh
located in the interior of that voxel, the identified vertices being
included in the respective set of intersection points for that voxel.

10. The computer program product as in claim 7, wherein each convex hull for the
voxel includes a set of triangles, each of the set of triangles separating an
interior volume of the proxy mesh from an exterior volume of the proxy mesh,
and
wherein aggregating the respective convex hulls for the subset of the
grid of voxels includes:
adding each of the set of triangles to the proxy mesh.

11. The computer program product as in claim 7, wherein aggregating the
respective convex hulls for the subset of the grid of voxels includes:
merging the convex hulls of each of the respective convex hulls for the
grid of voxels such that duplicate vertices and edges are deleted.

12. The computer program product as in claim 7, wherein each voxel of the subset
includes a respective interior and a respective boundary,
wherein each convex hull for the voxel includes a set of triangles, and
wherein aggregating the respective convex hulls for the subset of the
grid of voxels includes:
for each voxel of the subset of the grid of voxels,
identifying a triangle of the set of triangles of the convex hull
for that voxel not embedded in the boundary of that voxel, and
adding the identified triangle to the proxy mesh.
13. An apparatus, the apparatus comprising:
   memory; and
   controlling circuitry coupled to the memory, the controlling circuitry
   being configured to:
   receive input mesh data representing a mesh for a three-
   dimensional object, the mesh including a plurality of vertices, a
   plurality of edges, and a plurality of faces;
   for each of a subset of a grid of voxels,
   determine a respective set of intersection points from at
   least one intersection between that voxel and the mesh; and
   generate a convex hull of the respective set of
   intersection points for that voxel; and
   aggregate the respective convex hulls for the subset of the grid
   of voxels to produce a proxy mesh of the three-dimensional object.

14. The apparatus as in claim 13, wherein each voxel of the subset includes a
   respective interior and a respective boundary, and
   wherein the controlling circuitry configured to determine the respective
   set of intersection points is further configured to:
   identify points on the boundary of that voxel through which an
   edge of the plurality of edges or a vertex of the plurality of vertices of
   the mesh intersect the boundary of that voxel.

15. The apparatus as in claim 13, wherein each voxel of the subset includes a
   respective interior and a respective boundary, and
   wherein the controlling circuitry configured to determine the respective
   set of intersection points is further configured to:
   identify vertices of the plurality of vertices of the mesh located
   in the interior of that voxel, the identified vertices being included in the
   respective set of intersection points for that voxel.

16. The apparatus as in claim 13, wherein each convex hull for the voxel includes
   a set of triangles, each of the set of triangles separating an interior volume of
   the proxy mesh from an exterior volume of the proxy mesh, and
wherein the controlling circuitry configured to aggregate the respective convex hulls for the subset of the grid of voxels is further configured to:

add each of the set of triangles to the proxy mesh.

17. The apparatus as in claim 13, wherein the controlling circuitry configured to aggregate the respective convex hulls for the subset of the grid of voxels is further configured to:

merge the convex hulls of each of the respective convex hulls for the grid of voxels such that duplicate vertices and edges are deleted.

18. The apparatus as in claim 13, wherein each voxel of the subset includes a respective interior and a respective boundary,

wherein each convex hull for the voxel includes a set of triangles, and wherein the controlling circuitry configured to aggregate the respective convex hulls for the subset of the grid of voxels is further configured to:

for each voxel of the subset of the grid of voxels, identify a triangle of the set of triangles of the convex hull for that voxel not embedded in the boundary of that voxel; and

add the identified triangle to the proxy mesh.

19. The apparatus as in claim 13, wherein the controlling circuitry is further configured to:

display the proxy mesh on a display device.
Computer 120

Network Interface 122

Processing Units 124

Memory 126

Mesh Acquisition Manager 130
- Mesh Data 132
- Vertex Data 134
- Edge Data 136
- Face Data 138

Voxel Manager 140
- Voxel Data 142
- Voxel Grid Data 144
- Voxel Size Data 146

Intersection Manager 150
- Intersection Data 152
  - Voxel Boundary Intersection Data 154
  - Voxel Interior Intersection Data 156

Convex Hull Manager 160
- Convex Hull Data 162
  - Triangle Data 164

Aggregation Manager 170
- Deduplication Manager 172
- Boundary Embedding Manager 174

Proxy Mesh Data 176

Display Interface 128

FIG. 1

Display 190
Receive input mesh data representing a mesh for a complex three-dimensional object, the mesh including a plurality of vertices, a plurality of edges, and a plurality of faces, the complex three-dimensional object including at least one feature having a dimension smaller than a smallest dimension of a grid of voxels overlaid on the mesh.

For each voxel of the grid of voxels:

Determine a respective set of intersection points from at least one intersection between that voxel and the mesh.

Generate a convex hull of the respective set of intersection points for that voxel.

Aggregate the respective convex hulls for the grid of voxels to produce a proxy mesh of the complex three-dimensional object.
INTERNATIONAL SEARCH REPORT

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06T17/20
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, WPI Data, INSPEC

C. DOCUMENTS CONSIDERED TO BE RELEVANT

<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
</table>

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 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 reasons (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

"X" 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

"Y" 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

"Z" 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

"A" document member of the same patent family

Date of the actual completion of the international search

28 June 2022

Date of mailing of the international search report

15/07/2022

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

Authorized officer

González Arias, P
<table>
<thead>
<tr>
<th>Patent document cited in search report</th>
<th>Publication date</th>
<th>Patent family member(s)</th>
<th>Publication date</th>
</tr>
</thead>
<tbody>
<tr>
<td>US 2019221034 A1</td>
<td>18-07-2019</td>
<td>NONE</td>
<td></td>
</tr>
<tr>
<td>US 2005146522 A1</td>
<td>07-07-2005</td>
<td>NONE</td>
<td></td>
</tr>
</tbody>
</table>