September 02, 2016

3D Object Search System

Wallace P. Scott

3D OBJECT SEARCH SYSTEM

TECHNICAL FIELD

[0001] The subject matter of the present disclosure relates to searching a large repository of 3D models (*e.g.*, objects) to find models similar to a query model. More specification, the subject matter of the present disclosure relates to semi-structuring of the 3D model data and implementing of approximate similarity algorithms in the search. Such a search may detect potential intellectual property infringements, both physical and virtual.

BACKGROUND

[0002] The boundaries between physical and virtual commerce are becoming increasingly blurred as online marketplaces, 3D printing and scanning, and file sharing open up new opportunities to businesses and consumers. Such technologies have many intellectual property owners worried that their objects now may be pirated with ease, thus choking off royalties or premiums needed to recoup development efforts of those objects. Providers of the technologies and channels through which pirating occurs are also hurt by this situation, due to damaged reputation, increased liability, and wasted effort spent surveilling for illicit activity. The damage stemming from 3D object piracy is perhaps most immediate to the consumer, whose safety may be endangered by the degradation of critical operating features of objects and by the absence of control over its physical production.

[0003] Traditional duplicate-detection tools, such as file checksums and Digital Rights Management, are not sufficient to guard against 3D object piracy. Because 3D objects are generated from easily modified files, objects are often tweaked slightly and resold, still violating copyright but avoiding detection. In a way, this issue is the inverse of traditional counterfeiting—rather than trying to create and pass off a near-identical copy of an established brand's product, the attacker is trying to claim that the object is his original creation by modifying it from its authentic form.

SUMMARY

[0004] Systems and methods described here provide comparative analysis results between 3D objects that may exhibit superficial and/or abstract similarities. The objective is to efficiently sort through large datasets (*e.g.*, a library of 3D models) to detect not just duplicates to a query object but similar objects that may have been derived from the query object, and to provide quantitative descriptors of object similarities. Implementations include a novel method of consistently structuring 3D model data that represents a given model as a feature tree with varying levels of 'abstraction' in order to avoid skewed comparison results due to superficial model modifications. Because tree comparison algorithms are computationally expensive, this full structuring of 3D model data can be represented as a semi-structured histogram or signature accompanied metadata that relates the data in each bin to its level in the feature tree hierarchy in order to deliver quicker search and comparison results. These results can be achieved with a high degree of efficiency from very large datasets (such as an online marketplace or repository for CAD files) by using an approximate nearest or k-nearest neighbor algorithm (such as Locality-Sensitive Hashing) and an approximate Earth Mover's Distance algorithm (*e.g.*, Signature EMD or Wavelet EMD). Implementations draw high-level model similarity and low-level feature similarity conclusions with minimal computational expense (linear time runtime) and with any desired level of probability. Such conclusions, in combination with feature tree metadata, provide the necessary tools to perform extensive and accurate follow-up analysis on whether intellectual property infringement has occurred in one or more 3D objects.

[0005] 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

[0006] FIG. 1 is a block diagram that illustrates an example of a 3D model similarity search system, in accordance with disclosed implementations.

[0007] FIG. 2 is an example diagram that illustrates a 3D model of a lightbulb represented as its constituent primitive surfaces that make up the hierarchical feature tree, using a surface age metric, according to an implementation.

[0008] FIG. 3 is an example diagram that illustrates the influence of a given surface's bounding box volume and Gaussian curvature on its surface age, according to an implementation.

[0009] FIG. 4 is an example diagram that illustrates a 3D model represented as a hierarchical feature tree and a semi-structured histogram, using the surface age metric, according to an implementation.

[0010] FIG. 5 is a flow diagram of an example process for searching a large, unstructured set of 3D models for overall similarity and specific feature similarity to a query model, according to an implementation.

[0011] FIG. 6 illustrates an example process for generating a feature tree, according to an implementation.

DETAILED DESCRIPTION

[0012] FIG. 1 illustrates an example block diagram of a 3D model similarity search system, in accordance with disclosed implementations. The 3D model similarity search system 100 can be embodied, for example, on one or more computing devices, such as server 110, and client 180. Server 110 may be a mainframe, a server, a group of servers, a rack system, networked personal computers, *etc*., that include one or more processors formed in a substrate, which are configured to execute instructions stored in a memory. The instructions, when executed by the one or more processors, make the server 110 a specially programmed machine, The server 110 may be a combination of two or more computing devices. For example, two or more computing devices may be physically or logically distinct from each other but in communication with each other via a communications network (not shown in FIG. 1), operating as server 110. The server 110 may be accessible to other computing devices via a network 150, such as the Internet.

[0013] The server 110 may include modules or engines stored in memory, *e.g.*, non-transitory memory such as RAM, flash, disk, optical drives, cache, main memory, etc. The modules may include a model abstraction engine 120. The model abstraction engine 120 may take as input a computer aided design (CAD) model and generate an abstraction of the model that can be used to find similar models in a repository. The abstraction may be a histogram based on surface age or a signature based on surface age,

as will be explained in more detail below. The modules may also include a model comparison engine 130 configured to compare a query model to models in a repository to find models in the repository with a sufficient match. In some implementations, the server 110 may include model repository abstractions 142. Model repository abstractions 142 may be abstractions of 3D models stored in 3D model repository 160, which may be stored on server 110 or may be located remotely from server 110. The model repository abstractions 142 may be calculated ahead of time (*e.g.*, prior to a query) or may be calculated at the time of the query. In some implementations, an entry in the model repository abstractions 142 may be generated each time a model in the 3D model repository 160 changes. For example, the server 110 may include an engine similar to a web crawler that looks for changes to 3D models in the 3D model repository 160 and generates a new model repository abstraction 142 for models that have changed. As another example, the server 110 may be notified when a model has changed and may generate a new model repository abstraction 142.

[0014] The server 110 may also include a model comparison engine 130. The model comparison engine may receive a query object, or in other words a CAD model for comparison against objects in the 3D model repository 160. The model comparison engine 130 may use model abstraction engine 120 to generate an abstraction of the query object and then compare the abstraction of the query object to the model repository abstractions 142.

[0015] The system 100 may also include a remote client 180. The remote client 180 may be another server, a personal computing device, or a smartphone. The client 180 may be in communication with server 120, *e.g.*, via network 150, and may be used to submit a query object, *e.g.*, a 3D model file, to the server 110.

[0016] **Feature Extraction and Model Abstraction**

[0017] Each Computer Aided Design workflow begins with primitive geometric objects that gradually acquire greater detail and often ends with widely differentiated objects or assemblies. No single step in the modeling process can be said to impart an object's 'identity;' rather, it emerges over a series of modeling steps and is expressed through characteristic properties of each of these steps. While a simple object similarity search (*e.g.*, a conclusion drawn from a random sampling of surface coordinates) is

sufficient in many applications, such a method would be vulnerable to 'rolling back' a model's history by deleting features and perhaps subsequently introducing new features. In order to weigh similarities between objects while avoiding false negatives due to superficial feature differences, the objects being compared are compared at varying levels of abstraction (*i.e.*, from primitive shapes to fully detailed models). In order to accomplish this, the feature similarity query system may build a feature tree, where each node in the tree represents one or more surfaces of a volume in the CAD model that have a similar surface age. FIG. 6 illustrates an example process for generating a feature tree.

[0018] As a first step in building the feature tree, the feature similarity query system may perform feature extraction across one or more rounds of model abstraction. The input to the feature extraction and model abstraction is a clean CAD model, *i.e.*, one that contains a set of only manifold surfaces and 'watertight' volumes, not 'polygon soup' or point cloud models. The system discretizes the model into its constituent volumes (step 605) if more than one volume is present (*e.g.*, if the model is an assembly of separate parts). Each surface patch of the constituent volume was introduced somewhere along the model's abstraction spectrum, and larger surfaces with lower curvature tend to be more 'fundamental' to an object's identity than smaller, more tightly curved surfaces. FIG. 2 is an example diagram that illustrates a 3D model of a lightbulb represented as its constituent primitive surfaces that make up the hierarchical feature tree that can be analyzed using a surface age metric. In the example of FIG. 2, the light bulb 100 is first separated into a plurality of surfaces 205 to 235 as predefined by the input model's CAD data. As indicated above, the system may assume a clean CAD, but an additional precautionary step may be taken to ensure the CAD data is represented in its simplest, cleanest possible form.

[0019] In order to construct a feature tree that represents an object and that allows the system to produce increasingly abstract representations of an object by successively trimming off the tree's lowest-level branches, implementations define a metric, referred to as surface age, that enables the system to determine each surface's proper place in a node of the feature tree (step 615). A different way of stating this problem is that the system determines which 'child' nodes (lower-level surface patches) belong to 'parent' nodes (more abstract surface patches) using the surface age.

[0020] The definition for this metric can be expressed as:

$$Age = \frac{(\boldsymbol{\alpha} * V) + [(1 - \boldsymbol{\alpha}) * \boldsymbol{\beta} * S]}{\kappa^2 + 1}$$

where $\boldsymbol{\alpha}$ is the skewness or 'flatness' of the bounding box of the surface (*i.e.*, its deviation from a perfect cube), defined as:

$$\boldsymbol{\alpha} = \begin{cases} \frac{bc}{a^2}, & b, c \neq a \\ 1, & a = b = c \end{cases}$$

where *a*, *b*, and *c* are the lengths of the bounding box sides, listed from largest to smallest values, respectively,

$\boldsymbol{\beta}$ is a normalization factor for comparing the surface area of the bounding box of the surface to the volume of the bounding box of the surface, defined as:

$$\boldsymbol{\beta} = 2((1/a) + (1/b) + (1/c))$$

V is the bounding box volume of the surface (*i.e.*, the smallest box that can fully contain a surface and is aligned with the surface's principal axes),

S is the surface area of the bounding box of the surface,

and κ is the Gaussian curvature of the surface.

[0021] The greater the value of the age of a surface, the higher it is likely to reside in a node in an object's feature tree (*i.e.*, to have more levels of child nodes beneath it). As discussed above, each node in the feature tree may represent one or more neighboring surfaces with similar age. Neither the bounding box volume nor the curvature of a surface is sufficient by itself to describe that surface's place in a feature tree. A large, flat surface is likely to be part of a high-level feature, but it may have a flat bounding box with very small volume. Therefore, in order to produce a set of widely different ages by which to sort out a model's constituent surfaces, the system divides the size of the surface (expressed by a combination of its volume and surface area) by the squared Gaussian curvature to produce values that tend to be either large or very small. This is the most condensed expression for producing an arrangement of surface identifiers that assists the system in locating the most 'low-level' surfaces among the surfaces of the volume. FIG. 3 is an example diagram that illustrates the influence of a given surface's bounding box volume and Gaussian curvature on its surface age. In the

example of FIG. 3, surface 300 has an age much greater than 1. The system may determine this by the size of the bounding box and the Gaussian curvature. Surface 305 has an age of approximately 1, and surface 310 has a surface age much less than 1.

[0022] Once all the ages for a volume's surfaces have been derived, the system may define those with the smallest values, defined for example using a statistical grouping method, as the input surfaces to a node completion routine. The node completion routine is as follows:

1. Starting with a first surface and its age, find each neighboring surface, where a neighboring surface shares a common boundary with the first surface.
2. If the age of the neighboring surface is within a given similarity range of the starting or first surface (*i.e.*, $\boldsymbol{\gamma} > \left|age_{starting} - age_{neighbor}\right|$, where $\boldsymbol{\gamma}$ is the similarity threshold), these surfaces are assigned to the same node in the feature tree.
3. Repeat this search for all the neighbor surface's neighbors, *i.e.*, identifying neighbors with an age within the same $\boldsymbol{\gamma}$ value until no new surfaces are added to the node. In other words, the system may stop looking at neighbors when all neighboring surfaces are much larger or much smaller surfaces. This allows the system to 'leapfrog' surfaces belonging to potential children (*e.g.*, surfaces in a child node of the current node and capture the entire node currently being analyzed.

[0023] The surfaces with the given similarity are tentatively defined as the lowest-level nodes of the volume's (and therefore, the object's) feature tree, but it cannot yet be assumed with certainty that all these surfaces are lowest-level children. Accordingly, the system may perform a parent-child node testing routine on each node within the lowest level. The set of surfaces that are placed in the lowest-level node may be considered collectively as a child node. The system may perform the parent-child node testing routine as follows:

1. Starting with the child node, the ages of its surfaces, and the bounding box of the full node, or in other words the set of surfaces in the child node, find neighboring surfaces that either 1) share a common boundary or 2) have a

bounding box that significantly overlaps that of the node's (*i.e.*, $\boldsymbol{\varepsilon} < v_{node,child} \cap v_{surf,neighbor}$, where $\boldsymbol{\varepsilon}$ is the bounding box overlap threshold). The surfaces that either share a common boundary or have a bounding box that significantly overlaps are connected surfaces.

2. If the ages of the connected surfaces are outside of a given similarity range from the age of the (child node's average surface age, at least part of the parent node has been found. The similarity range used to determine a parent node is different from the similarity threshold $\boldsymbol{\gamma}$ in that this threshold indicator is much greater. Put another way, the age of the neighbor node should be much larger than the child node's average surface age.
3. Using this parent node surface, run the node completion routine.
4. Using this parent node, find all the child surfaces of that node by searching for all connected surfaces with much smaller ages or those whose bounding boxes are entirely enclosed by the parent node bounding box surface.
    a. Run node completion routine on child surfaces to find complete child nodes.

[0024] Any surface-to-node assignment conflicts (whether during the node completion or parent-child test routines) are resolved by reverting back to the results of the prior assignment and continuing on with the routine. If a parent node has not been identified for a child node after searching through the input set, the search is expanded to the remainder of the volume (which guarantees the assignment of a parent node, assuming clean CAD geometry as input). A parent node is guaranteed by this method to be found, if the starting node is not the highest-level node itself of the feature tree (in which case, the routine is already complete).

[0025] At this point, the entire first level of the feature tree has been found. Characteristic information of these features are recorded (step 625) before the features are deleted from the model. In order to clean up the CAD model to provide the proper input to the routines described above, the system may use a surface fitting and cleanup routine (step 635). The system may use conventional or later developed surface fitting and

cleanup routines. The objective of this routine should be to minimize deviation from the original surfaces while minimizing the number of control vertices needed to construct the new surfaces. Once a new, clean, 'abstracted' CAD model has been generated, the system may repeat the Feature Extraction and Model Abstraction routine (step 640, No) building a new tree with the surfaces generated by the fitting and cleanup routine. This system may repeat this process until no significant age differentiation exists between the surfaces composing a volume, and the highest-level node of the volume's feature tree can be said to have been reached. This is considered a flat tree condition (step 640, Yes). For models containing more than one volume, the same process is repeated using volume ages (analogous to surface ages) (step 645, Yes) until the highest-level node of the model's feature tree can be said to have been reached (645, No).

[0026] Depending on the precise implementation of the following model comparison procedures, each input model's tree structure may be represented either as a histogram or a histogram signature that expresses its set of surface ages (FIG. 4). In FIG. 4, a histogram 405 $\{h_{\mathbf{i}}\}$ is defined as a 1-dimensional set of vectors **i** mapped to the set of non-negative real numbers that measure the surface age. The system may also use a histogram signature, which includes compressing the histogram data by eliminating empty bins and grouping or clustering in all data 'nearby' a mean value. A histogram signature $\{s_j = (\mathbf{m}_j, w_j)\}$ is defined as a 1-dimensional set of feature clusters, each represented by its mean $\mathbf{m}_j$ and by the fraction w_j of surface ages that belong to that cluster j. The system may mark each entry to the histogram 405 or signature with a node-assignment metadata tag, which carries information about the node in the feature tree to which the surface age belongs. The metadata tag may aid the system in determining node assignment in the feature tree because the surface ages are correlated, but not exact pointers, to feature tree nodes.

[0027] The system may not use the feature tree directly for feature classification in the following section. Instead, the feature tree, by capturing higher levels of abstraction in the histogram and signatures, enables the system to detect skewing of the surface age data of high-level nodes away from their originals simply by introducing superficial model modifications, like feature addition, deletion, or morphing. It is also useful so that post-processing procedures following the model comparison routines (*e.g.*, feature

similarity visualization) can examine node-assignment metadata in conjunction with high-level model similarity data or low-level feature similarity data between models. This information, *i.e.*, the signature or the histogram, serves as the input to the feature classification routines described in the following section. The system may store the feature tree, and/or the histogram and signature that results from the Feature Abstraction and Model Abstraction, for example as model repository abstractions 142 of FIG. 1. Thus, the system maybe able to access this information for comparison with a query model.

[0028] **Feature Classification**

[0029] FIG. 5 is a flow diagram of an example process 500 for searching a large, unstructured set of 3D models for overall similarity and specific feature similarity to a query model, according to an implementation. Process 500 may be performed by a 3D model search system, such as system 100 of FIG. 1. Process 500 may include query repository pre-processing (505), as described above with regard to FIG. 6 (*i.e.*, feature extraction and model abstraction with regard to each model in a 3D model repository). Step 505 may be performed ahead of receiving a query model. Process 500 may also include query model pre-processing (510). Query model pre-processing includes performing feature extraction and model abstraction on the query model, again as described above with regard to FIG. 6.

[0030] While comparing two models using their tree structures may be the most thorough method to evaluate similarities between them, tree comparison algorithms are notoriously process-time-intensive (with many run times increasing at rates of $O(n^2)$, $O(n^4)$, *etc.*, as the input size n increases linearly). In fact, for all but the most strictly constrained tree comparisons, an optimal approach to this type of problem is mathematically intractable. Therefore, the system disregards a thorough tree comparison procedure in favor of two successive, approximate similarity procedures. The first similarity procedure may narrow down the set of models in the repository (*e.g.*, in model repository abstractions 142) likely to exhibit some meaningful level of similarity to the query model (515). The second similarity procedure may serve as a detailed follow-up comparison between the narrowed-down set and the query model (520). In some implementations the similarity procedures may be Locality-Sensitive Hashing (LSH) and

Earth Mover's Distance (EMD). While LSH and EMD are used for ease of explanation below, implementations may use other similar similarity procedures

[0031] The system may use a procedure like Locality-Sensitive Hashing (LSH) to quickly find similar entries to a query item in large, high-dimensional datasets. LSH has been applied to Internet search engine indexing, where response speed and accuracy are critical. LSH delivers probabilistic results, meaning that while it cannot guarantee an exact nearest or k-nearest neighbor(s) answer, the level of certainty of its answer can be pushed to any arbitrary probability (though at increased computational expense). The premise behind the family of LSH algorithms is that since any two points that lie close to each other in Euclidean space will have small l_2-norm (or, distance) values when viewed from any given direction, one can sample these values from arbitrary directions with increasing certainty of 'closeness' as the number of samplings increase. These algorithms generally run in linear time, or O(n).

[0032] The E^2LSH function, belonging to a popular implementation of LSH, is defined as follows:

$$h^{x,b}(\vec{v}) = \left\lfloor \frac{\vec{x} \cdot \vec{v} + b}{w} \right\rfloor$$

where $\vec{x}$ is a random vector selected from a Gaussian distribution, b is a random variable uniformly distributed between 0 and w to aid in error analysis, and $\vec{v}$ is a query vector in a high-dimensional space.

[0033] Given a failure probability threshold (or degree of desired approximate certainty), the number of iterations of this function that the system performs is defined as follows:

$$L = \frac{\lfloor \log \delta \rfloor}{\log(1 - P^k)}$$

where L is the number of iterations, δ is the failure probability, P is the hash table size (a large prime number), and k is the number of dimensions of the vector space.

[0034] By providing the query histogram or signature as a high-dimensional vector, and the search space of models as a set of high-dimensional vectors, the system then define the desired level of confidence and 'closeness' threshold of nearest neighbors before running the LSH routine. The output of this routine is a vastly reduced

dissimilarity space on which to perform more extensive analysis, produced in linear (rather than exponential or worse) time (O(n)).

[0035] Once the nearest neighbors of the query model have been found, the next approximate similarity procedure can be driven by the Earth Mover's Distance (EMD) or similar algorithm. A formal definition of the Earth Mover's Distance algorithm as a linear programming problem is as follows:

Solve for a Flow $\mathbf{F} = [f_{ij}]$, with f_{ij} the flow between histogram bins p_i and q_j, that minimizes the overall cost:

$$WORK(P,Q,F) = \sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij}f_{ij},$$

subject to:

$$f_{ij} \geq 0 \qquad 1 \leq i \leq m, 1 \leq j \leq n$$

$$\sum_{j=1}^{n} f_{ij} \leq w_{p_i} \qquad 1 \leq i \leq m$$

$$\sum_{i=1}^{m} f_{ij} \leq w_{q_j} \qquad 1 \leq j \leq n$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij} = \min\left(\sum_{i=1}^{m} w_{p_i}, \sum_{j=1}^{n} w_{q_j}\right)$$

Once the optimal flow F has been found, the EMD metric is defined as the work normalized by the total flow:

$$EMD(P,Q) = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij}f_{ij}}{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij}}$$

[0036] EMD has been demonstrated to produce a true dissimilarity metric that is more robust than histogram matching techniques (such as Minkowski-form distance, χ^2 statistics, *etc.*) and has been applied extensively in the field of computer vision. It is particularly advantageous over other methods in its ability to detect matches within subsets of data, instead of only a 'high-level' dissimilarity metric. However, EMD also suffers from high computational complexity (O($n^3 \log n$), partly because it is an

optimization algorithm and because most histogram bins are usually empty (thus resulting in 'wasted' time during the comparison procedure).

[0037] One solution that the system may use to work around the issue of computational complexity is to approximate the input histogram $\{h_i\}$ using a signature $\{s_j = (\mathbf{m}_j, w_j)\}$ for each model; this effectively compresses the information expressed through the data and thus reduces the time required by the EMD algorithm to converge on a solution. Not only does this approach avoid degradation of similarity query results, but it has been shown to consistently improve on results achieved through the histogram-based approach.

[0038] Another solution the system may use delivers even greater time savings than other methods by approximating the EMD between two histograms with the wavelet transform of the difference histogram. This difference histogram can be expressed as a wavelet series, defined as:

$$f(x) = \sum_k f_k \phi(x-k) + \sum_\lambda f_\lambda \psi_\lambda(x)$$

where k represents shifts, ϕ is the scaling function, ψ is the wavelet, and $\lambda := (j,k)$ is the Lagrangian multiplier (which is used to identify extrema, or points of interest in determining the minimal EMD value), and where j represents the scale, or 'resolution,' of the wavelet functions.

[0039] Wavelet transforms are generally useful for compressing sparsely distributed signals in a set of data, which is the case with a histogram representing a model's set of surface ages. By approximating the EMD with weighted wavelet transform coefficients, comparisons between two models' histograms can be completed in linear time (O(n)).

[0040] Regardless of approximate EMD's specific implementation, its advantages over a nearest or k-nearest neighbor search include its robustness against partial alterations to the input data (*e.g.*, a 3D models' features being locally added, deleted, or morphed), as well as a higher degree of granularity than high-level model similarity (*e.g.*, comparing histogram or signature bins p_i and q_j). Particularly in light of the vastness of the datasets from which these detailed comparisons must be made, implementations of

approximate similarity procedure on semi-structured data delivers the optimal tradeoff between certainty of results accuracy and computational cost (FIG. 5).

[0041] Although this invention has been disclosed in the context of certain preferred embodiments and examples, it will be understood by those skilled in the art that the present invention extends beyond the specifically disclosed embodiments to other alternative embodiments and/or uses of the invention and obvious modifications and equivalents thereof. In addition, while a number of variations of the invention have been shown and described in detail, other modifications, which are within the scope of this invention, will be readily apparent to those of skill in the art based upon this disclosure. It is also contemplated that various combinations or sub-combinations of the specific features and aspects of the embodiments may be made and still fall within the scope of the invention. Accordingly, it should be understood that various features and aspects of the disclosed embodiments can be combined with or substituted for one another in order to form varying modes of the disclosed invention. Thus, it is intended that the scope of the present invention herein disclosed should not be limited by the particular disclosed embodiments described above, but should be determined only by a fair reading of the disclosure.

WHAT IS CLAIMED IS: Scott: 3D Object Search System

1. A method for searching a large, unstructured set of 3D models for overall similarity and specific feature similarity to a query model, the method comprising:

generating a feature tree for the query model;

generating a histogram from the feature tree based on surface ages for surfaces in the query model;

determining a set of nearest neighbors to the query model using the histogram and histograms for the set of 3D models; and

determining, from amongst the nearest neighbors, whether a model from the set of 3D models matches the query model using an approximate similarity procedure.

2. The method of claim 1, wherein generating the feature tree includes:

determining surfaces for a volume in the query model;

determining a respective surface age for each surface; and

generating nodes for the feature tree based on the respective surface ages, where neighboring surfaces with similar surface ages are grouped in a common node of the feature tree.

3. The method of claim 1, wherein generating the feature includes:

cleaning up the model by merging surfaces in nodes in lower levels of the feature tree include surfaces with lower surface ages having similar surface ages.

4. The method of claim 1, wherein the feature tree includes nodes representing increasingly abstract surfaces.

5. The method of claim 1, further comprising assigning each surface for the query model to a node in the feature tree.