Early Experiences with a 3D Model Search Engine

Patrick Min John A. Halderman Michael Kazhdan Thomas A. Funkhouser*

Princeton University

Abstract

New acquisition and modeling tools make it easier to create 3D models, and affordable and powerful graphics hardware makes it easier to use them. As a result, the number of 3D models available on the web is increasing rapidly. However, it is still not as easy to find 3D models as it is to find, for example, text documents and images. What is needed is a "3D model search engine," a specialized search engine that targets 3D models. We created a prototype 3D model search engine to investigate the design and implementation issues. Our search engine can be partitioned into three main components: (1) acquisition: 3D models have to be collected from the web, (2) analysis: they have to be analyzed for later matching, and (3) query processing and matching: an online system has to match user queries to the collected 3D models. Our site currently indexes over 36,000 models, of which about 31,000 are freely available. In addition to a text search interface, it offers several 3D and 2D shape-based query interfaces. Since it went online one year ago (in November 2001), it has processed over 148,000 searches from 37,800 hosts in 103 different countries. Currently 20–25% of the about 1,000 visitors per week are returning users. This paper reports on our initial experiences designing, building, and running the 3D model search engine.

CR Categories: H.3.5 [Online Information Services]: Web-based Services; I.3.8 [Computer Graphics]: Applications

Keywords: Specialized search engine, 3D model database, shape matching, shape query interfaces

1 Introduction

Pushed by the fast increase in the performance of affordable graphics hardware, 3D graphics has found its way into many mainstream applications. New powerful modeling software and new acquisition techniques, such as 3D scanners, further boost the number of available 3D models. Many of these models are easily accessible on the web.

However, unlike text documents, 3D models are not easily located. There is no search engine for 3D content, and the few available repository sites each offer a limited number of models, accessible only by browsing a directory structure or running a text keyword search. Attempting to find a 3D model using just text keywords suffers from the same problems as any text search: a text description may be too limited, incorrect, ambiguous, or in a different language. Furthermore, 3D models contain shape and appearance information, which is hard to query using just text.

*e-mail: min,jhalderm,mkazhdan,funk@cs.princeton.edu

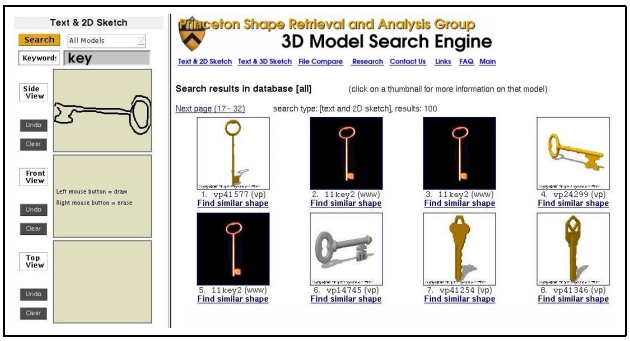

Figure 1: A 3D Model Search Engine, with an example query and some resulting 3D models

We have developed a search engine for 3D models that supports shape-based queries, as well as textual ones. In many cases, a shape query is able to describe a property of a 3D model that is hard to specify using just text. For example, Figure 1 shows a combined text and 2D shape query, in which the keyword "key" is used to select the class of models we are interested in (keys), and a simple 2D outline sketch is used to pinpoint the particular kind of key we want (old-fashioned keys with an open handle and a simple bit). Here the combination of a shape and text query results in an effective search.

In this paper, we describe the design and implementation choices made when building our search engine and report on initial experiences running it. This hopefully will benefit future developers of similar 3D search systems. We partitioned the engine into three main components: (1) acquisition: 3D models have to be collected from the web, (2) analysis: they have to be analyzed for later matching, and (3) query processing and matching: an online system has to match user queries to the collected 3D models. We report results on the performance of these three components, as well as site usage statistics.

2 Previous Work

A search engine for 3D models belongs to the category of *specialized search engines*. These engines collect a relatively small subset of domain-specific data from the web, and make it more accessible through a suitable user interface. Examples are CiteSeer, a search engine for scientific articles [Bollacker et al. 1998], Deadliner, for conference and workshop announcements [Kruger et al. 2000], and HomePageSearch,

for homepages of computer scientists [Hoff and Mundhenk 2001]. Because it is impractical to download every webpage and determine for each whether it contains relevant data, specialized search engines typically use some form of *focused crawling* [Chakrabarti et al. 1999]. Focused crawlers use heuristics to estimate if a link is likely to point to a relevant page and follow them in likelihood order.

An important advantage of a specialized search engine is that it can provide a domain-specific query interface. For example, the web site of the State Hermitage Museum in St. Petersburg uses the "Query By Image Content" method to allow users to search paintings by drawing simple colored sketches [Faloutsos et al. 1994]. Rowe *et al.* describe a system where the user can draw a 2D outline of a ceramic vessel as a shape query into a database of 3D models of such vessels [Rowe et al. 2001].

Many web sites allow users to find 3D models, but they typically provide only limited query options. Online repositories, such as [3D Café] and [Avalon], offer models for free, while several companies sell 3D models through online catalogs (for example [Cacheforce] and [Viewpoint]). These sites only provide a directory browser and sometimes a text search option as a query interface. Other sites, such as [CADlib] and [MeshNose], index multiple 3D model collections, but they also just support directory browsing and text search interfaces. The National Design Repository, an online repository of CAD models, allows searches by text keyword, and by file type and size, or by browsing through directories [Drexel University].

Several other web sites allow searching based on 3D shape. For example, at the "ShapeSifter" site of Heriot-Watt University, the user can select from a long list of shape features, such as surface area, bounding box diagonal length, and convex hull volume, and perform a search with conditions on these features [Corney et al. 2002]. The search is in a CAD test database with 102 L-shaped blocks and several transformed versions of about 20 other models. In the online demo of the commercial system "Alexandria," the user can set weights of individual model attributes (for example "geometry", "angular geometry", "distributions", and "colour") to be used in matching, and search in a database of 4,500 commercial models [Paquet and Rioux 2000]. In the experimental system "Ogden IV," the user can pick the matching method to be used for a shape search (matching grid-based or rotation invariant feature descriptors at several different grid resolutions) and search in a database of 1,500 VRML models, although they are not available for download [Suzuki 2001]. At the experimental site "3D Shape Retrieval Engine" of Utrecht University, the user can also pick one of three matching methods (Gaussian curvature, Normal variations, Midpoints) and one of three test databases: (1) a database of 133 web models collected from the web by Osada *et al.* [2001], (2) a database of 684 models (containing 366 airplanes) the authors collected from the web, and (3) the same database of 102 L-shaped blocks used in the ShapeSifter site [Tangelder and Veltkamp 2003; Utrecht University]. Elad *et al.* present a novel query method in a system (which is not online) that matches a 3D model to a database of 3D models using moments-based matching. In a set of resulting closest matching models, the user can mark specific models as "good" or "bad," after which weights in the matching function are adjusted appropriately and the matching method is repeated [Elad et al. 2001]. A "3D Object Retrieval System" from National Taiwan University supports a few query methods similar to ours: the user can draw one or two 2D sketches, upload a 3D model file (in the Wavefront OBJ format) for matching, or select a result model from a previous search as a query [Chen and Ouhyoung 2002]. Two databases (containing models manually downloaded from the web) can be searched, one with 445 models using a matching method introduced by Hilaga *et al.* [Hilaga et al. 2001], and one with 10,000 models using a matching method based on matching projected 2D images of a model.

Unfortunately, all the online sites are experimental in nature, their query methods are usually not very user-friendly, and most of the model databases are either inaccessible or relatively small. When addressing these problems, several questions arise: How do we efficiently find the 3D models that are available on the web? Which query interfaces are easy to use, yet are descriptive enough to find what you are looking for? How do we efficiently and effectively match these queries to a 3D model database?

We have developed an online 3D model search engine to investigate these issues. To find the available 3D models on the web we developed a focused crawler for 3D models. Our system provides a wide variety of query interfaces based on text, 3D shape, and 2D shape, which were designed to be easy to use, and hide parameters of the underlying matching methods. We use recently introduced 2D and 3D shape matching methods which are both efficient and effective (as demonstrated in [Funkhouser et al. 2003] and [Kazhdan et al. 2003]). The contribution of this paper is an investigation of the design and implementation trade-offs in building such a 3D model search engine based on these methods.

3 System Design and Implementation

The search engine consists of three main components: (1) acquisition, (2) analysis and indexing, and (3) query processing and matching. Figure 2 shows a high-level schematic overview of the system. The following subsections describe each component in more detail.

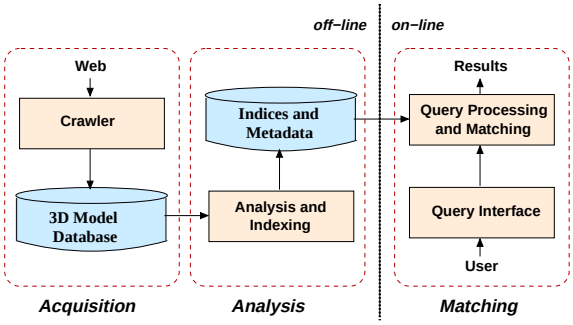

Figure 2: System Overview

3.1 Acquisition

The first task is to find the high-quality 3D models available on the web. This problem is similar to that of other specialized search engines: finding domain specific data that is contained in a small subset of the web.

One may think that simply searching for links pointing to files with a 3D model file extension (e.g. `.dxf`) is sufficient. This strategy works well when searching for files with a unique extension, such as VRML files, which have

the extension `.wrl` or `.wrl.gz`. Also, being a file format designed for the web, VRML files are usually linked to directly rather than contained in compressed archive files. In a first attempt at acquiring models, we only searched for VRML models and used query results from sites that allow text queries into link text only (i.e. the text in the A HREF tag). For many other 3D model file types, this straightforward approach is unsuitable: for example, the extension `.obj` is both used for Wavefront 3D model files as well as Windows object files. Also, many model files are compressed into archives, which have extensions such as `.zip` and `.tar.gz`. Searching for and downloading all archives is impractical, as most of them do not contain 3D models. As a result, a more general searching strategy is required.

Other specialized search engines address the problem of finding a very small subset of domain-specific data by *focusing* their search (also called *focused crawling*) [Chakrabarti et al. 1999]. A priority is assigned to links on each downloaded web page, and these links are followed in priority order. So the problem becomes one of assigning priorities to links such that the crawl is directed to sites with 3D models. We cannot use a page-ranking method like the one Google uses (where a page's priority is determined by the number of links pointing *to* it, and the priorities of these referring pages [Brin and Page 1998]) because most models are pointed to by a single link. Instead, we assign priorities to web *sites*, not to individual pages, and determine a site's priority by the number of 3D models already found on that site. As a result, our crawler concentrates on model "repositories" (sites containing large numbers of 3D models), rather than sites that contain only a handful of models among many other files. This is beneficial to users because models from repositories are usually of consistently high quality and are more likely to be accessible for longer periods of time. In initial tests, we used the triangle count of a model to help determine page priority, but this became a bottleneck in the system because the models could not be converted and analyzed fast enough to keep up with the crawler. Thus, we settled on model frequency alone as a compromise between focusing effectiveness and speed.

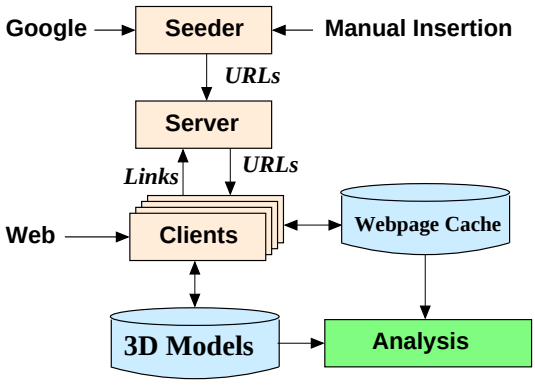

Figure 3: Acquisition stage: the crawler

Our crawler uses this focusing method and works as follows (see Figure 3 for an overview). A "crawler server" is seeded with the first 100 search results from four Google text queries, each of the following form: `<specific term> AND <general term>`. The `<general term>` was `(model* OR mesh* OR object*) AND (3-d OR 3d)`. The `<specific term>` depended on the 3D file type, and was one of `("3d studio" OR 3ds)`, `(vrml OR wrl)`, `(wavefront OR obj)`, and `(lightwave OR lwo OR lw)`. Although we do download Autocad DXF files, we did not query for this format because the Google results included too many 2D files. The 400 URLs returned from these queries were used as the starting points for our crawler. To these, we added another 159 hand-picked URLs to model repository and other 3D graphics related sites.

The crawler ranks each site with a priority from 0–5, with 0 being the highest priority, and maintains a separate queue of pages to be downloaded from each site. Pages from higher priority sites are retrieved first, with the limitation that only one simultaneous connection can be open to a given site (to limit the load on any particular server). The initial sites were assigned a priority 1, since they were likely to be relevant. Links to a new site are assigned the "parent" site priority plus 1. If a link points to a model or archive, it goes to the head of the site's page queue, and the site's priority is temporarily set to the maximum priority. To find other pages, the crawler server farms out web retrieval requests to a number of clients. Once a client returns pages retrieved from a particular site, the site's priority is recomputed. The new priority depends on the number of pages retrieved from the site so far and what percentage of those were 3D models. For example, a percentage larger than 10% will always put a site in priority class 1.

Smaller percentages will put it in lower priority classes, but only if enough pages have been downloaded from that site. In other words, a site will not be demoted because of a small percentage computed after only a few pages have been downloaded. More formally, if for a site the number of pages returned so far is p and the number of those that were models m, then its priority class c will be:

$$\begin{cases} \lfloor -\log(\frac{m}{p}) \rfloor + 1 & \text{if} \quad \frac{m}{p} > 10^{-4} \\ 5 & \text{if} \quad \frac{m}{p} \leq 10^{-4} \end{cases}$$

but *only* if $d = \log(\frac{p}{k}) > c$ (i.e. if enough pages have been downloaded), otherwise the priority remains unchanged. The constant k determines the cutoff values for the number of downloaded pages. For example, assume k is set to 5, and for a certain site $p = 1,000$, and $m = 25$. Then $c = \lfloor -\log(\frac{25}{1,000}) \rfloor + 1 = \lfloor -(-1.6) \rfloor + 1 = 2$, and $d = \log(\frac{1,000}{5}) = 2.3 > c$, so the site's priority class will be set to 2. In our crawler we set k to 5, though we still need to test how this constant impacts crawler performance.

3.2 Analysis and Matching

Our search engine supports queries based on text, 3D shape, and 2D shape. For each of these input modes, the relevant information is extracted from the 3D database models during an *offline* stage, so that they can be efficiently compared to *online* queries later. This means that relevant text associated with each 3D model has to be identified and that both 3D and 2D "shape descriptors" have to be computed. These shape descriptors are designed such that they are a concise representation of overall shape that can be indexed.

In addition, metadata such as polygon counts and thumbnail images are computed so we can produce more informative results pages. The full sequence of analysis operations is shown in Figure 4, with each data type approximately in its own quadrant.

3.2.1 Text

The most basic query interface we support uses text keywords. These keywords are matched against a representa-

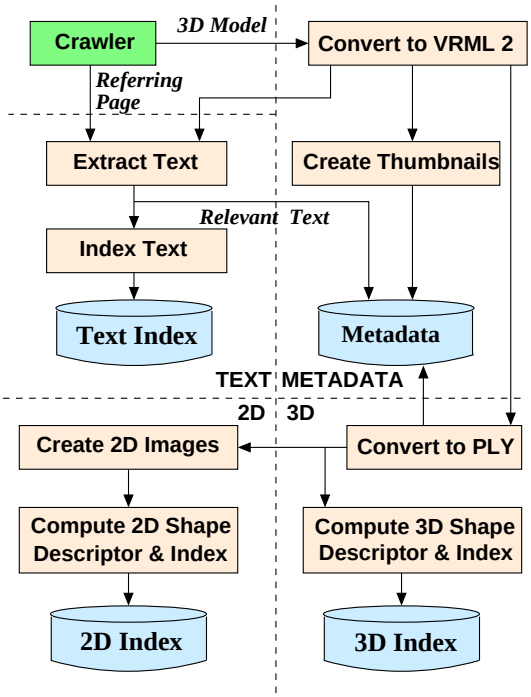

Figure 4: Preprocessing: analysis stage

tive text file of each 3D model. The text in this file includes the model filename, the anchor text, nearby text parsed from the referring page, and ASCII labels parsed from inside the model file. The latter category includes part names (e.g. "DEF" nodes in VRML), texture file names, and other informational fields (e.g. the "WorldInfo" node in VRML). We found that including comments is counter-productive, as files often contain commented-out geometry, which floods the documents with indiscriminating keywords.

Following common practices from the text retrieval literature, each text file is preprocessed by removing common words (*stop words*) that do not carry much discriminating information, such as "and," "or," and "my." We use the SMART system's stop list of 524 common words [Salton 1971], as well as stop words specific to our domain (e.g. "jpg," "www," "transform"). Next, synonyms and hypernyms (category descriptors) are added to the document using WordNet [Miller 1995]. This enables queries like "vehicle" to return objects like trucks and cars, or "television" to return a TV. Unfortunately, this last step increases the number of bad matches. For example, WordNet returns "domestic animal" as a hypernym for the word "head" (i.e. a head of cattle), and as a result the text query "animal" returns several models of heads. The resulting text file is stemmed (normalized by removing inflectional changes) using the Porter stemmer: for example "wheels" is changed to "wheel" [Porter 1980]. Figure 5 shows an example model (a Ferrari car from a commercial model database [De Espona]) and part of its representative text file.

In order to match text documents to user-specified keywords or to other documents, we use the *TF-IDF/Rocchio* method, a popular weighting and classification scheme for text documents [Joachims 1997; Rocchio 1971; Salton 1988]. This method assigns a vector of term weights to each document based on a term's frequency in the document (Term Frequency (TF), higher is better), and its frequency over

Filename: `f40`
Wordnet synonyms:
`car auto automobile machine motorcar motor vehicle automotive railcar railway railroad wheel cable compartment`
Webpage context: `ferrari car vehicle`

The first 20 model file identifiers: `camera01 timer pos interp rot spot01 spot02 spot04 spot05 spot03 dummy01 target suelo f40 f40pirel maps pirelli2 f40tire f40whlc f40horse`

Figure 5: Example model of a Ferrari car and part of its representative text file

all documents (Inverse Document Frequency (IDF), lower is better). The similarity score between two vectors is simply the cosine of the angle between them.

The stemming, indexing and matching is done using a program called `rainbow`, which is part of the Bow toolkit, a freely available C library for statistical text analysis [McCallum 1996]. It allows the text matcher to run as a server with the text index in memory, greatly accelerating the matching process.

3.2.2 3D Shape

Our system provides three ways to specify a 3D shape query: the user can (1) upload an existing 3D model file or (2) select a 3D model on a results page (called a *result model query*). We also allow the user to (3) create a 3D shape query from scratch, using a simple 3D "sketching" tool. So far, we have experimented with Teddy [Igarashi et al. 1999], a gesture-based 3D sketching tool. We chose Teddy because it is the simplest 3D modeling tool we could find: it is fairly easy to learn, and allows one to create coarse shapes quickly, both useful properties for our application.

Each of these query interfaces supplies a 3D model, whose shape has to be compared to the shapes of the 3D models in our database. We do this by converting them to 3D *shape descriptors* and comparing those. The 3D shape descriptor we use is described in [Funkhouser et al. 2003; Kazhdan et al. 2003]. It is especially suitable for our application because of the following properties: it is robust under model degeneracies, concise and efficient to compute (the compute time is 2 seconds on average, on a 2.2 GHz Xeon CPU, for a model with 3,500 triangles), and invariant to similarity transformations (with normalization for scale and translation). Furthermore, matching two descriptors can be done simply by computing the Euclidian distance between them, and thus large databases of descriptors can be indexed efficiently (matching a single descriptor to a database of about 36,000 descriptors takes less than 0.4 seconds). Finally, the method outperforms other current 3D shape matching methods in precision/recall tests on a large test database (1890 models classified into 85 classes). For full details, see [Funkhouser et al. 2003] and [Kazhdan et al. 2003].

3.2.3 2D Shape

To provide another, simpler shape query method, we also support a 2D sketch interface. We found that a simple 3D sketching tool like Teddy is still too hard to use for average users. The 2D interface is similar to the pen-drawing

interface in Paintbrush-style programs: pixels can be drawn by dragging the mouse. Since this way of drawing is similar to drawing using pencil and paper, it is more likely to be familiar to the user.

The user is asked to draw up to three simple 2D *outline* sketches: these sketches are matched to multiple outline images of the 3D models in the database as seen from different directions. The matching method compares 2D descriptors computed from the images, which are precomputed for the outline images of the database models. These 2D image descriptors are concise and efficient to compute, invariant to similarity transformations (with normalization for scale and translation) and reflections, and robust under small user inaccuracies. Comparing and indexing the 2D descriptors is done using methods similar to the ones used for the 3D descriptors. For more details, see [Funkhouser et al. 2003; Min et al. 2002].

The n ($1 \leq n \leq 3$) user sketches are matched to m outline images of a database model by matching all combinations of n out of m images, with the restriction that no two user sketches can match a single database image. The matching score of multiple user sketches is then the smallest sum of the individual scores, for all combinations.

We chose to use a small number $m = 7$ of outline images for each 3D model for the following reasons. In a user study, we found that people tend to prefer the simpler, orthogonal views (top, side and front views) over "diagonal" views [Funkhouser et al. 2003]. Also, most 3D models in the database appear to be axis-aligned. This implies we should at least include the three axis-aligned views (looking from the center of the sides of a cube towards the centroid). Our matching method is invariant to reflection, so views from the opposite sides are not needed. We added four views (looking from the corners of a cube towards the centroid) to cover non-orthogonal user input and models which are not axis-aligned. Sampling from more directions increases the likelihood that a certain user sketch matches a model outline more precisely, but also increases the number of false positives.

3.2.4 Combined Queries

Text queries may be combined with 2D or 3D sketches. An example "text & 2D sketch" query (for old-fashioned keys) was shown in Figure 1 on the first page of this paper. Combined queries can be more effective than text or shape queries alone: shape properties of a model may be hard to describe using text but relatively easy to sketch. For example, when searching for tables with a single leg, one could submit a text query "table" plus a simple 2D outline sketch of such a table as seen from the side. This query and some results are shown in Figure 6. A text query "table" by itself would have returned many different tables, and the fact that a table has a single leg is usually not described in its associated text. Similarly, submitting just the sketch as a query adds irrelevant results such as molecules and chairs.

To match a combined query, we simply perform both queries separately, and combine their results by mean-normalizing and averaging their matching scores.

3.2.5 Metadata

In order to create informative results pages, some extra information has to be extracted from the 3D models. Specifically, we create thumbnail images of the model as seen from three directions and provide polygon counts.

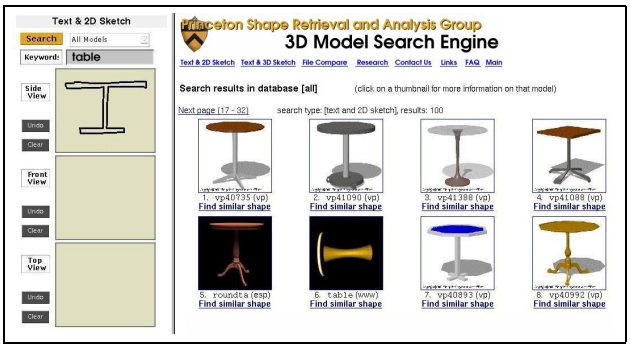

Figure 6: Example combined "text & 2D sketch" query for round tables with a single leg

The thumbnail images are created from VRML files using ParallelGraphics' Cortona VRML plugin [ParallelGraphics]. Non-VRML files are first converted to VRML 2.0 using PolyTrans (a commercial 3D model converter) to simplify the thumbnail generation stage and later conversions [Okino Computer Graphics]. From the VRML 2.0 files eight thumbnail images are created. The first thumbnail shows the model as seen from a camera viewing one of the largest two sides of the model's bounding box. A smaller version of the first thumbnail is used on the main results page. The viewpoints for the 2nd and 3rd thumbnails are another bounding box side view, and a view orthogonal to the first two. If the VRML file has at least one `Viewpoint` node, then the camera of the first `Viewpoint` node is used for one of the thumbnails (because this is the viewpoint that the model creator intended to be the default). The remaining thumbnails show the model in several different rendering styles (wireframe, hidden-line) and in close-up.

Next, the VRML 2.0 files are triangulated and converted to the PLY format using our own VRML reader and freeware PLY software [PLY]. Since PLY is a simple, geometry only format, this conversion simplifies the later 2D and 3D shape analysis stages. During conversion, the number of triangles, quads, and larger polygons is counted and stored, as well as the number of triangles after triangulation (Note that, because our input models are VRML, the `Box`, `Sphere`, `Cone`, `Cylinder`, `Extrusion` and `ElevationGrid` nodes are also triangulated). Model files containing 3D text only (i.e. VRML Text nodes, representing text displayed as 2D surfaces in 3D) were ignored (this was the case for 0.5% of all downloaded models).

3.3 Query Processing

This section describes the steps a query goes through as it progresses through our system. The full process (with paths for all possible queries) is shown in Figure 7. One path, for a 3D sketch query, has been highlighted.

For each query interface, Table 1 lists the language it was implemented in, and what data gets sent from the user to the web server. A "browse results" request (i.e. requesting a next or previous page with match results) fetches the next or previous 16 results from a cached set of all results from a query.

The web server is only responsible for data collection and building a results page. We could have run the matching process on the web server as well. However, since it is computationally intensive, we implemented a matching server on a separate PC designed to handle higher loads: it has two

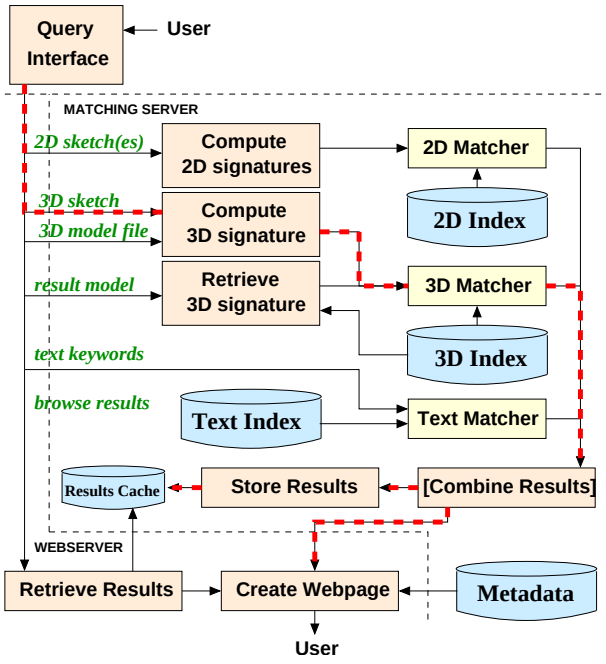

Figure 7: Query processing and matching stage. The path of a 3D sketch query has been highlighted.

query type	implemented as	query data
2D sketch	Java applet	pixel coordinates of "set" pixels
3D sketch	Java applet	Wavefront OBJ file
3D model file	Javascript	actual modelfile
result model	HTML link	unique model id
text keywords	Java applet	actual keywords

Table 1: Implementation method, and query data sent for each query interface

1.5 GHz Xeon CPUs and 512 MB memory. The web server sends matching requests via TCP to the matching server. Query files such as 3D sketch object files are transferred via NFS.

Matching of the incoming queries is handled as follows:

- Text queries are passed directly to the Bow toolkit classifier `rainbow` [McCallum 1996]
- 2D sketches, which initially are represented as a list of foreground pixels, are saved as PGM files by the web server (images with zero pixels are ignored). The match server passes these to a separate 2D match server which computes their 2D shape descriptors and does the matching
- 3D sketches and uploaded 3D model files are first converted to just geometry in the PLY format using an in-house converter which reads VRML 2.0 (and a small subset of the Wavefront OBJ, Autocad DXF, Persistance of Vision POV, DEC OFF, and Unigrafix UG formats). The PLY file is passed to a separate 3D match server, which computes its 3D shape descriptor, and matches it to the descriptors in the database
- For "result model" queries (queries where a model on a results page is selected as a 3D shape query) only a unique model id needs to be passed to the matching server, which then retrieves its descriptor from the database and matches it
- For combined (text & 2D/3D sketch) queries, the individual queries are performed separately, and their matching results are combined by mean-normalizing and averaging the scores

The 2D, 3D, and text matchers each run as server processes that compare incoming descriptors/text against an in-memory index and return model id's and matching scores to the matching server. This information is returned to the calling script on the web server, which then builds the results web page and returns it to the user. Figure 8 shows an example of a results page for a text query "dog." Clicking on a "find similar shape" link below a thumbnail image submits that particular model as a 3D shape query (i.e. a "result model" query). Match results are cached to enable fast browsing of multiple results pages using "Previous page" / "Next page" links at the top of the results page.

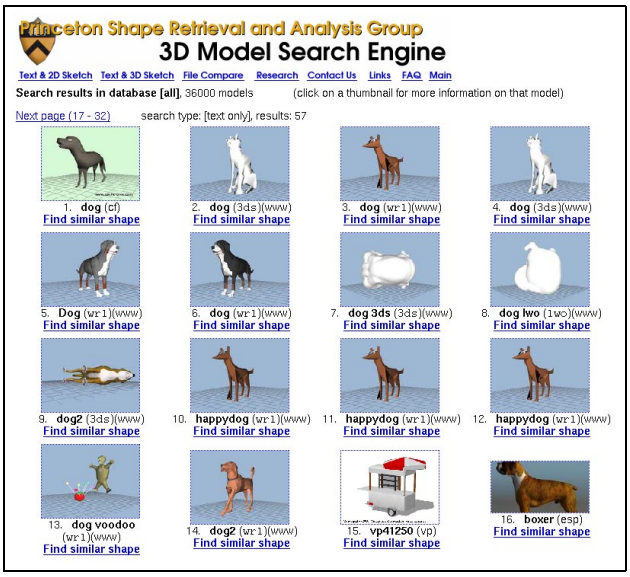

Figure 8: Example results page for a text query "dog"

3.4 Result Reporting

In deciding what to show the user on a results page, several factors have to be considered. The user should have enough information about each model to be able to determine if it is interesting or not, without using too much bandwidth. A single results page should show as many models as possible without appearing cluttered. The most information would be provided if each model were shown in its own small 3D viewer (e.g. a VRML plugin), but this would require a high-performance PC at the user's end and a high bandwidth connection (to transfer the potentially large model files). The least possible amount of information is just each model's filename, which in most cases is insufficient.

We chose to provide model information at two levels of detail: the first is a page with 16 results, shown as a 4 by 4 matrix of labeled thumbnail images (of 128×96 pixels each), as shown in Figure 8. This results page occupies about 625×825 pixels of screen space (with the query interface the

total width is about 900 pixels). The next level of detail is shown after the user clicks on a thumbnail image on a results page. This brings up a window with more information about that model: eight 160×120 thumbnail images from different viewpoints and in different rendering styles (which can each be enlarged to 640×480), links to the model and its referring page, polygon count information, and the text associated with the model. Figure 9 shows an example of such a window for a 3D model of a dog (result number 5 in Figure 8).

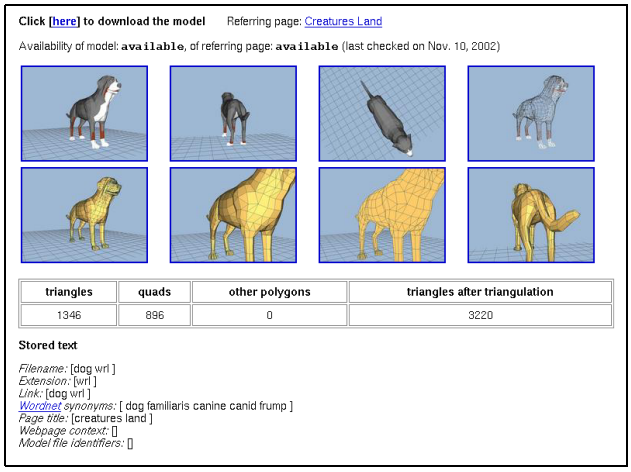

Figure 9: Example information window for a model of a dog (result number 5 in Figure 8)

4 Results

This section reports results for the acquisition, analysis, and query processing stages.

4.1 Acquisition

We performed two crawls to test different acquisition strategies. The first ran in October 2001 and targeted VRML files only. This crawl began with the results of search engine queries for web pages linking to VRML files and crawled outward from these in a breadth-first fashion. The second crawl ran in August 2002 and targeted five popular 3D file formats, possibly contained in compressed archive files. It used the focused crawling technique described in Section 3.1 to home in on relevant model files from a smaller set of starting pages.

4.1.1 First Crawl

An initial crawl in October 2001 searched for VRML models only. In this crawl, we simply used the results of the AltaVista and HotBot search engines to seed our search. These sites allow the user to search in the URL text of links. VRML model files have a unique extension and are usually linked to directly rather than contained in compressed archives. For these reasons, simply searching for `wrl` yielded a good set of starting pages for our crawler.

The crawl ran for 48 hours and resulted in 22,243 VRML files, retrieved from 2,185 different sites. VRML files can include other VRML files (using `Inline` nodes for example, similar to the C `#include` directive), which also were retrieved. This resulted in an additional 63,676 VRML files (these files typically represent subparts of the larger main model).

file type	number of models	linked to directly	in compressed archive
VRML	15,167 (76%)	11,339 (95%)	3,828
3D Studio	2,483 (12%)	608	1,875
Lightwave	1,153 (5.7%)	14	1,139
Autocad	737 (3.7%)	0	737
Wavefront	544 (2.7%)	12	532
Total	20,084	11,973	8,111

Table 2: Number of 3D model files retrieved of each type in the second crawl

4.1.2 Second Crawl

A second crawl in August 2002 targeted VRML, 3D Studio, Autocad, Wavefront, and Lightwave objects, both in plain links as well as in compressed archive files. Unlike VRML, the other formats were not designed to be used on the web and often are contained within compressed archives, so they typically cannot be located by the technique used in our first crawler. Instead, the second crawler searched for them using our focused crawling method, which favors repository sites.

The second crawl ran for 2 days and 16 hours on one server and one client PC. This crawl took more time than the first one (which took 2 days) because much more data was downloaded (about 30 GB vs. 4.6 GB, mostly due to the many large compressed archive files). The server sent 946,100 URL retrieval requests to the client (about 4.1 per second on average). 804,413 (85%) were HTML files, 81,975 (8.7%) could not be retrieved because of network or server errors, 13,217 (1.4%) were 3D model files (i.e. models linked to directly, containing 1,244 duplicate links, leaving 11,973 unique model files), 5,539 (0.6%) were archive files containing 3D models, and 40,868 (4.3%) were archive files with no 3D model files in them. Of the archives containing 3D models, 79% had the extension `.zip`, 20% had `.gz` or `.tar.gz`, and a handful had `.lha`, `.tar`, or `.Z`.

After decompression, there were 20,084 model files retrieved from 455 different sites. The file formats are distributed as shown in Table 2.

To investigate how the incoming model rate changed over time during the second crawl, we plot the number of models per 10,000 incoming URLs in Figure 10. The decay in the "model yield" in this graph shows that, starting from this set of 559 seed URLs, we retrieved most of the available models after examining about 950,000 URLs, which is why we stopped the crawling process at this point.

Comparing the results of both crawls we see that the second crawl found many (mostly non-VRML) models inside compressed archives that could not have been found using a search method based on identifying file extensions in links. Fewer VRML files were found in the second crawl because we no longer took advantage of the AltaVista and HotBot indices to find links to VRML files like we did in the first crawl. The different crawling strategies are reflected in the fact that the second crawl retrieved a similar number of models from far fewer sites (455 compared to 2,185), and the relatively small overlap between the two sets of models (19% of the models from the second crawl were also found by the first). Overall, the second crawl may be characterized as narrower and deeper than the first.

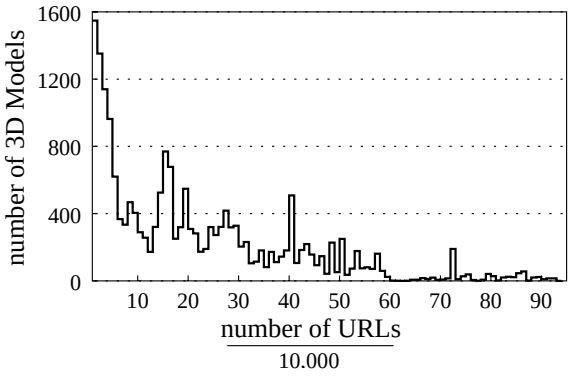

Figure 10: Number of incoming 3D models per 10,000 retrieved URLs in the second crawl

Surprisingly, the distributions of the number of triangles per model of the first and the second crawl are nearly identical (see Figure 11 for a graph of both). We expected the second crawl to retrieve more complex models than the first, because it targeted repository sites. We are still investigating whether the similar distribution is caused by the fact that the first crawl retrieved a relatively large number of models from repository sites, or that repository sites do not necessarily carry larger models than other model sites.

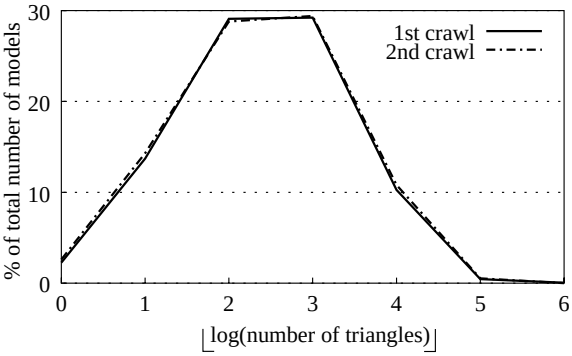

Figure 11: Distribution of the number of triangles per model for the first and the second crawl

4.2 Model Database Size

As of December 2002, our site indexes over 36,000 models. Of the 42,000 models downloaded in the two crawls, about 31,000 remain after conversion and processing (as described in the next section). An additional 5,000 commercial models were provided by [Cacheforce; De Espona; Viewpoint].

4.3 Analysis

In this section we present statistics gathered during 3D model file conversions, such as percentage of successful conversions, average polygon counts per model file type, and timings of the various processing steps.

4.3.1 Model Conversion

Table 3 shows the number of successful conversions (using Okino's PolyTrans [Okino Computer Graphics]), and the average number of triangles for the models retrieved in the second crawl (not counting models with zero triangles). 12,082 of the VRML files were already in the 2.0 format. Assuming that the number of triangles is an indication of the complexity of a model, then these numbers show that VRML files are on average much simpler than models in the other formats.

file type	number of models	successfully converted	average # triangles
VRML	15,167	14,782	3,182
3D Studio	2,483	2,473	10,860
Lightwave	1,153	1,141	10,717
Autocad	737	719	8,611
Wavefront	544	293	10,086
Total	20,084	19,408	5,691

Table 3: Number of successful conversions, and average number of triangles for each 3D model file type of the second crawl

		first crawl	second crawl
1	model files	22,243	19,408
2	duplicates	n/a	3,763 (19%)
3	no geometry	4,466 (20%)	2,397 (12%)
4	remaining models	17,777 (80%)	13,248 (68%)
Total number of indexed web models: 31,025			

Table 4: The number of (1) VRML 2.0 model files acquired and created in both crawls, (2) models found in the second crawl that were also found by the first, (3) models lost because of conversion errors in our converter, and models without geometry, and (4) remaining useable models. The last line shows the total number of web models in our database.

Table 4 shows for both crawls the initial number of VRML 2.0 files, and how many were unusable because of missing geometry or conversion errors. The conversion errors were caused both by errors in the model files as well as bugs and missing features in our VRML parser. For the second crawl, 99.7% of the models lost in this stage were originally in the VRML format.

4.3.2 Descriptors, Indices and Metadata

This section reports the time taken for the preprocessing steps for the VRML model database acquired in the first crawl. See Figure 4 in section 3.2 for an overview of these steps. The model conversion to VRML was done on a 195 MHz MIPS R10000 CPU running Irix 6.5. All other processing was done on Intel CPUs running Red Hat Linux 7. The thumbnail generation ran on a 733 MHz Pentium III, capturing the 2D outline images for the 2D descriptor index on a 1.5 GHz Xeon, and the remaining processing steps on a 2.2 GHz Xeon.

Table 5 lists the running times for each of the preprocessing steps. Creating the actual indices from the extracted text was done using the `rainbow` toolkit [McCallum 1996] and took 11 seconds. The conversion to VRML 2.0 involved the conversion of about 15,500 VRML 1.0 and Inventor files (this includes files referenced using `WWWInline`). Triangle counts were computed during the "conversion to PLY" step.

Average storage requirements for the VRML models from the first crawl are 500 Kb for the model file itself (including texture files), 27 Kb for the text data, 400 Kb for the thumbnail images, 260 Kb for the PLY file, and 17 Kb for the 2D and 3D shape signatures.

preprocessing step	approx. time (in hours:mins)
extract text	1:20
convert to VRML 2.0	3:20
create thumbnails	60:00
convert to PLY	6:00
compute 3D index	9:30
create 2D images	18:00
compute 2D index	1:40

Table 5: Time taken for the preprocessing steps of the 17,777 models acquired in the first crawl

query type	processing and matching time (in sec)
text	0.22
2D sketch	0.61
text & 2D sketch	0.59
3D sketch	3.2
text & 3D sketch	3.2
result model	0.36
file upload	5.0

Table 6: Average time for processing and matching for each query type

4.4 Query Processing Performance

Here we report the response times of our site for the various query types.

The response time a user experiences is the sum of the time it takes for the following operations: (1) connecting to our web server and sending query data, (2) executing a CGI script on the web server (which connects to and has to wait for the matching server), (3) processing and matching of the query on the matching server, (4) returning the results to the user, and (5) rendering the results web page on the user's machine. The time taken in steps (1) and (4) depends on the available bandwidth between the user's machine and our web server, step (5) depends on the performance of the user's machine. Step (2) adds an estimated overhead of about 1–2 seconds. We can report accurate timings for step (3): processing and matching of the query on the matching server.

Table 6 shows for each query type the average time used for processing and matching on the matching server, averaged over about 16,000 searches performed since our model database was expanded to 36,000 models. These numbers show that the response time is mostly determined by the amount of query data that needs to be processed. Text and "result model" queries take the least time, followed by queries that involve 2D sketch(es) (for which 2D image(s) are converted to image descriptors), and queries that require the conversion of 3D models (3D sketch and file upload). In the latter case, the 3D model file size dominates the total processing time. The conversion times for a few example model sizes are: 60 KB, 2,000 triangles: 4 seconds, 800 KB, 6,500 triangles: 6 seconds, 2 MB, 61,000 triangles: 13 seconds (2 MB is currently the filesize limit on the "file upload" feature).

The time between when a query arrives at the web server and when its results are ready is about 0.4 seconds on average. To investigate how this average time increases under increasing load, we ran two experiments in which we sent queries to our web server at a higher rate. These queries were taken from a set of 4,000 representative queries submitted to our search engine, logged during a one week period in July 2002. In the first test we sent on average one query every 4 seconds for a period of two hours from a single host. In the second test we sent twice the number of queries from 4 different hosts (which is a more than 120 times higher rate than the currently typical rate of about 400 queries per day). The average processing time per query increased to 0.88 and 1.46 seconds per query, respectively (the maximum times were 5 and 7 seconds), which shows that our current system will be able to accomodate much higher loads without incurring a significant performance penalty (i.e. one where the turnaround time becomes unacceptable for the user).

	# searches (%)	% info window	% ref. page	% model download
text	99,949 (67)	31.8	9.6	12.2
result model	22,696 (15)	37.9	12.3	15.2
2D sketch	16,130 (10.8)	18.1	4.3	2.9
text & 2D sketch	7,764 (5.2)	25.8	7.2	7.7
3D sketch	1,559 (1.0)	19.2	3.3	3.5
text & 3D sketch	615 (0.4)	30.7	8.9	10.0
file upload	100 (0.06)	25.0	7.0	8.0

Table 7: Relative use of each query interface, and user interest in the resulting models

4.5 Site Usage Statistics

In this section we show how the site has been used since it first came online in November 2001.

4.5.1 Queries

During a one year period starting on November 18, 2001, 158,159 queries have been processed. 148,974 of these resulted in an actual search being performed (8,164 queries were empty, i.e. searches with no text or shape input, and 1,021 file upload queries failed). 67% of these were text-only searches, 33% were shape-based, possibly combined with text. Table 7 shows the number of searches performed for each query interface type, and what percentage of those resulted in, at least once, (1) the opening of an information window, (2) the visiting of a referring page, and (3) the downloading of an actual 3D model.

The text-only query interface was the most popular. This may be explained by the following reasons: (1) it is the most familiar kind of interface on the web, (2) it is easy to supply a text query, (3) the total time from specifying the query to seeing the results is very short, and (4) users are unfamiliar with shape queries. The 99,949 text queries contained 18,941 unique queries. The 10 most popular text queries were car (2,912), human (1,360), woman (1,065), tree (1,003), house (918), cars (839), man (834), chair (773), ship (629), and building (602). 8,469 text queries (or 8.5%) resulted in zero matches: these included queries that were mistyped or in a foreign language, but also queries for objects not in our database, e.g. "audi," "cellphone," and "spiderman." These also included queries for objects that *are* present in our database but are annotated incorrectly (for example, the database contains buddha models but their filename is misspelled). In these cases, a shape-based query could have been more effective.

The relative number of times extra information is requested about a model could be an indication of how interesting the results are to the user. If this is true, then the "result model" query yields the most relevant results: after

15.2% of the "result model" searches the user downloaded at least one model, more than for any other query method. This suggests that users are able to use the "result model" query to home in on a desired model.

Few people used the other 3D shape-based interfaces. About 1.4% of all the queries involved the 3D sketching interface: to use it effectively, users have to take a small tutorial, and have some 3D manipulation skills. We think this was too much of a barrier for the average user. Even fewer people used the "upload 3D model file" feature. Most of the time it is misunderstood, and used to enter text keywords. This could be because we provide insufficient information on the site (e.g. we could have provided a short explanation about 3D models and modeling).

Overall, we think that ease of use and familiarity of a query interface determine its popularity: text is most popular, followed by "result model" (less familiar, but intuitive and easy to use), 2D sketch (familiar, but harder to use), and finally the 3D shape based interfaces. In future work, we hope to improve the user-friendliness of the 2D and 3D sketch interfaces.

4.5.2 Visitors

To investigate whether people find the search engine useful, we measured the number of unique hosts using our site per week, and what percentage of them return after a first visit. Figure 12 shows the number of unique hosts per week using our site since the first hit arrived on November 18, 2001. Note that hits from `*.princeton.edu` were not counted. During each visit on average 3.2 searches were performed. The sharp increase in the week starting on March 10th was due to the improved ranking of the site at Yahoo and Google. The sharp peak in the week of November 3 occurred because our site was mentioned on Slashdot, a popular discussion board for technical news. As a result, during the first six hours after the post over 3,800 queries were processed.

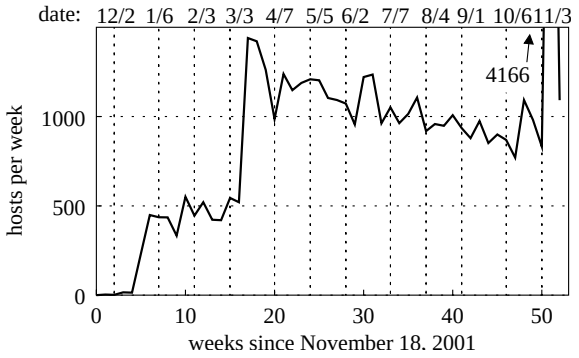

Figure 12: Number of unique hosts using the search engine per week (excluding local hits)

Figure 13 shows the percentage of returning hosts per week. A visitor is counted as returning if a subsequent visit occurred on a later date than a previous visit. It shows a clear upward trend, and currently approaches 25%, suggesting that an increasing number of people are using the site as a useful resource.

We were also interested in how widespread the usage of our search engine is. For this purpose we store the hostname of each visitor's PC and count the number of unique hostnames from each top-level domain. We received queries from 117 different top-level domains, 103 of which were from

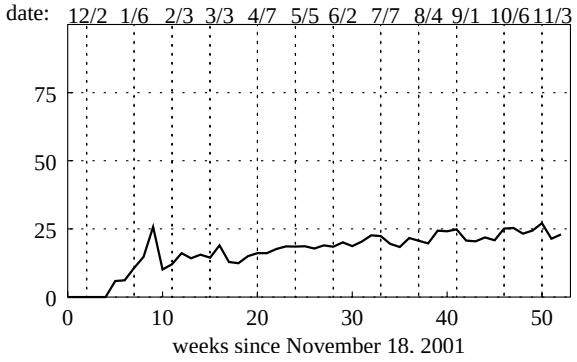

Figure 13: Percentage of returning hosts per week

	domain	hosts		domain	hosts
1	`net`	9,045	11	`es` (Spain)	592
2	`com`	6,899	12	`br` (Brazil)	506
3	`fr` (France)	1,666	13	`jp` (Japan)	364
4	`edu`	1,333	14	`mx` (Mexico)	301
5	`ca` (Canada)	830	15	`be` (Belgium)	301
6	`ge` (Germany)	818	16	`pl` (Poland)	259
7	`au` (Australia)	694	17	`at` (Austria)	258
8	`it` (Italy)	671	18	`ch` (Switzerland)	216
9	`nl` (Netherlands)	670	19	`ar` (Argentina)	207
10	`uk` (UK)	641	20	`se` (Sweden)	197

Table 8: Number of different hosts from the 20 most frequent top-level domains

specific countries. Table 8 shows the number of unique hosts visiting from the 20 most frequent top-level domains. Note that these numbers only give a rough indication of the relative use of our site in each country: the `.net` and `.com` domains are international, for 25% of the searches a hostname lookup failed, and dynamic IP addresses may cause the same host to be counted more than once. The numbers do indicate that the usage of our site is widespread, and highest among industrialized nations.

5 Limitations and Future Work

There are many avenues for future work.

First of all, it is not clear how many 3D models there are on the web, which prevents us from accurately evaluating the crawler's performance. We intend to investigate methods to compute a good estimate of the number of available 3D models on the web, and their distribution by application area (e.g. avatars, CAD, virtual environments, etcetera).

There are still ways to improve the query interfaces. The "Teddy" 3D sketching tool we currently use for 3D shape queries is hard to use for novice users. Restricting the freedom of user input can make it easier to build simple 3D objects. For example, by providing only a few basic shapes as building blocks, that then can be connected to form a larger, more complicated object. We are currently investigating a similar 2D query interface in which the user can draw a 2D shape as a collection of parts (e.g. ellipses). These parts will still be matched to images of the 3D database models as seen from different directions, but by also allowing for some variation in the relative location of parts, we may be able to match articulated models.

The focused crawler needs additional testing to evaluate the effectiveness of the set of seed queries and the parame-

ter settings for the site priority assignment. Also, we have not investigated other crawler issues such as recrawling the database to keep it up-to-date (by removing "dead links," currently occupying about 12–16% of the database) and the proper handling of duplicates. In the domain of 3D models, a duplicate can mean anything from a byte-identical copy of a model file, to one that has a very similar shape (e.g. a lower triangle count version of the same model).

Finally, we would like to increase the site's usefulness by including more search options (i.e. an "advanced search"), such as searching models of a specific filetype or with a minimum number of triangles.

6 Conclusions

Because of the fast growing number of 3D models that are available on the web, there is a need for a search engine specific to 3D data. Several factors contribute to this increase: new model acquisition methods and modeling software create an increasing supply of 3D models, and fast and cheap graphics hardware creates an increasing demand for them.

We built a prototype web-based 3D model search engine. It functions as a framework to evaluate methods in focused crawling for 3D data, shape matching methods, and shape query interfaces.

In this paper we discuss system design and implementation issues, and report usage statistics and crawler and web site performance. We find that there are enough 3D models available on the web to justify building a 3D model search engine. Judging from our usage statistics, it also appears that there is a demand for such a search engine: in one year we processed over 148,000 queries from 37,800 different hosts in 103 countries, resulting in more than 26,000 model downloads. Furthermore, currently 20–25% of the about 1,000 visitors per week are returning users.

Try it out yourself at `http://shape.cs.princeton.edu`

Acknowledgements

We would like to thank Bernard Chazelle, Joyce Chen, David Dobkin, Adam Finkelstein, David Jacobs, Emil Praun, and Szymon Rusinkiewicz, who provided useful comments and insights during the development of the search engine and the writing of the paper.

Cacheforce, Viewpoint Corporation, and Jose Maria De Espona donated commercial databases of polygonal models for experiments [Cacheforce; De Espona; Viewpoint]. The National Science Foundation provided partial funding for this project under grants CCR-0093343, 11S-0121446, CCR-99-88173, and DGE-9972930. The Army Research Organization provided partial funding under grant DAAD19-99-1-0205. Thomas Funkhouser is partially supported by an Alfred P. Sloan Fellowship.

References

3D Café. 3D Café free 3D models meshes. `http://www.3dcafe.com`.

Avalon. Avalon 3D archive. `http://avalon.viewpoint.com`.

Bollacker, K., Lawrence, S., and Giles, C. L. 1998. CiteSeer: An autonomous web agent for automatic retrieval and identification of interesting publications. In *Proceedings of the Second International Conference on Autonomous Agents*, ACM Press, New York, K. P. Sycara and M. Wooldridge, Eds., 116–123.

Brin, S., and Page, L. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems 30*, 1–7, 107–117.

Cacheforce. Cacheforce 3D model libraries. `http://www.cacheforce.com`.

CADlib. CADlib web based CAD parts library. `http://www.cadlib.co.uk`.

Chakrabarti, S., van den Berg, M., and Dom, B. 1999. Focused crawling: a new approach to topic-specific web resource discovery. In *Proc. 8th World Wide Web Conference*, Elsevier, Toronto, Canada, 545–562.

Chen, D.-Y., and Ouhyoung, M. 2002. A 3D object retrieval system based on multi-resolution reeb graph. In *Proc. Computer Graphics Workshop*, 16–20.

Corney, J., Rea, H., Clark, D., Pritchard, J., Breaks, M., and MacLeod, R. 2002. Coarse filters for shape matching. *IEEE Computer Graphics & Applications 22*, 3 (May/June), 65–74.

De Espona. De Espona Infografica 3D model collection. `http://www.deespona.com`.

Drexel University. National design repository. `http://www.designrepository.org`.

Elad, M., Tal, A., and Ar, S. 2001. Content based retrieval of VRML objects - an iterative and interactive approach. In *Proc. EG Multimedia*, 97–108.

Faloutsos, C., Barber, R., Flickner, M., Hafner, J., Niblack, W., Petkovic, D., and Equitz, W. 1994. Efficient and effective querying by image content. *Journal of Intelligent Information Systems 3*, 3/4, 231–262.

Funkhouser, T., Min, P., Kazhdan, M., Chen, J., Halderman, A., Dobkin, D., and Jacobs, D. 2003. A search engine for 3d models. *ACM Transactions on Graphics 22*, 1 (January).

Hilaga, M., Shinagawa, Y., Kohmura, T., and Kunii, T. L. 2001. Topology matching for fully automatic similarity estimation of 3D shapes. In *Proc. SIGGRAPH 2001*, ACM, 203–212.

Hoff, G., and Mundhenk, M. 2001. Creating a virtual library with HomePageSearch and Mops. In *Proc. 19th SIGDOC Conference*, ACM.

Igarashi, T., Matsuoka, S., and Tanaka, H. 1999. Teddy: A sketching interface for 3D freeform design. In *Proc. SIGGRAPH 1999*, ACM, 409–416.

Joachims, T. 1997. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In *Proc. Int. Conf. on Machine Learning*, 143–151.

Kazhdan, M., Chazelle, B., Dobkin, D., Funkhouser, T., and Rusinkiewicz, S. 2003. A reflective symmetry descriptor for 3D models. *Algorithmica, Special Issue on Shape Algorithmics. to appear.*

Kruger, A., Giles, C. L., Coetzee, F. M., Glover, E., Flake, G. W., Lawrence, S., and Omlin, C. 2000. DEADLINER: building a new niche search engine. In *Proc. Information and Knowledge Management.*

McCallum, A., 1996. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. `http://www.cs.cmu.edu/~mccallum/bow`.

MeshNose. MeshNose, the 3D objects search engine. `http://www.deepfx.com/meshnose`.

Miller, G. A. 1995. WordNet: A lexical database for English. *Communications of the ACM 38*, 11, 39–41.

Min, P., Chen, J., and Funkhouser, T., 2002. A 2D sketch interface for a 3D model search engine. SIGGRAPH Technical Sketches. p. 138.

OKINO COMPUTER GRAPHICS. Polytrans 3D model converter. http://www.okino.com/conv/conv.htm.

OSADA, R., FUNKHOUSER, T., CHAZELLE, B., AND DOBKIN, D. 2001. Matching 3D models with shape distributions. In *Proc. Shape Modeling International*, IEEE Press, Genova, Italy, 154–166.

PAQUET, E., AND RIOUX, M. 2000. Nefertiti: A tool for 3-D shape databases management. *SAE Transactions: Journal of Aerospace 108*, 387–393.

PARALLELGRAPHICS. Cortona VRML plugin. http://www.parallelgraphics.com.

PLY. 3D file format. http://www.cc.gatech.edu/projects/large_models/ply.html.

PORTER, M. 1980. An algorithm for suffix stripping. *Program 14*, 3, 130–137.

ROCCHIO, J. 1971. *The SMART Retrieval System: Experiments in Automatic Document Processing*. Prentice-Hall, ch. Relevance Feedback in Information Retrieval, 313–323.

ROWE, J., RAZDAN, A., COLLINS, D., AND PANCHANATHAN, S. 2001. A 3D digital library system: Capture, analysis, query, and display. In *Proc. 4th Int. Conf. on Asian Digital Libraries (ICADL)*.

SALTON, G. 1971. *The SMART retrieval system*. Prentice-Hall, Englewood Cliffs, NJ.

SALTON, G. 1988. *Automatic text processing: the transformation, analysis, and retrieval of information by computer*. Addison-Wesley, Reading, Massachusetts.

SUZUKI, M. T. 2001. A web-based retrieval system for 3D polygonal models. *Joint 9th IFSA World Congress and 20th NAFIPS International Conference (IFSA/NAFIPS2001)* (July), 2271–2276.

TANGELDER, J. W., AND VELTKAMP, R. C. 2003. Polyhedral model retrieval using weighted point sets. *International Journal of Image and Graphics 3*, 1, 1–21.

UTRECHT UNIVERSITY. 3D shape retrieval engine. http://www.cs.uu.nl/centers/give/imaging/3Drecog/3Dmatching.html.

VIEWPOINT. Viewpoint corporation. http://www.viewpoint.com.