Autotagging to Improve Text Search for 3D Models

Corey Goldfeder
Department of Computer Science
Columbia University
New York, NY
coreyg@cs.columbia.edu

Peter Allen
Department of Computer Science
Columbia University
New York, NY
allen@cs.columbia.edu

ABSTRACT

Text search on libraries of 3D models has traditionally worked poorly, as text annotations on 3D models are often unreliable or incomplete. We attempt to improve the recall of text search by automatically assigning appropriate tags to models. Our algorithm finds relevant tags by appealing to a large corpus of partially labeled example models, which does not have to be preclassified or otherwise prepared. For this purpose we use a copy of Google 3DWarehouse, a library of user contributed models which is publicly available on the Internet. Given a model to tag, we find geometrically similar models in the corpus, based on distances in a reduced dimensional space derived from Zernike descriptors. The labels of these neighbors are used as tag candidates for the model with probabilities proportional to the degree of geometric similarity. We show experimentally that text based search for 3D models using our computed tags can approach the quality of geometry based search. Finally, we describe our 3D model search engine that uses this algorithm.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*Indexing Methods*

General Terms

Algorithms, Experimentation

1. INTRODUCTION

Building a search engine for 3D models represents a significant user interface challenge. Existing 3D search engines require users to submit complex queries, such as drawing a sketch [1, 4] or providing an initial 3D model as a starting point. In contrast, the simplest and most natural interface, keyword search on the descriptive text associated with each model, is limited by a dependence on the accuracy of these descriptions, and more fundamentally by the requirement that they even exist.

In the past, text annotations on 3D models have been largely dismissed by shape search researchers as being unreliable and incomplete, and therefore of limited use in retrieval systems. Min, Kazhdan, and Funkhouser experimentally confirmed that searching on text alone is a poor retrieval strategy for 3D models drawn from the web [6]. In this work we use geometric similarity to propagate text tags between similar models and automatically generate salient keywords. Our goal is to improve the precision and recall of keyword based 3D model search to the point that it is comparable to searching on geometric shape descriptors. Compared to similar recent work in tagging 2D images [2], our 3D *autotagging* algorithm has the advantage of not requiring an explicit training stage.

2. AUTOTAGGING

Given an unlabeled 3D model ω, we wish to assign to ω a set of text tags from the set of all possible tags $\Lambda = \{\lambda^1, \lambda^2 \dots \lambda^n\}$. Specifically, for each tag λ^i we wish to assign a confidence value $P(\lambda^i, \omega)$ which we interpret as the probability that λ^i is a relevant tag for ω. We informally define relevancy to mean that a conscientious annotator would apply tag λ^i to model ω.

To tag ω, we make use of a corpus of known models $\Omega = \{\omega_1, \omega_2 \dots \omega_n\}$, where each model ω_x in Ω has associated initial tag probabilities $P(\lambda^i, \omega_x)$ for each λ^i in Λ, most of which will be zero. We start with a geometric shape similarity metric and find the neighbors of ω within some distance threshold $\tau(\omega)$. We use Zernike descriptors [8] but in principal any reasonable shape distance should do. Note that τ is allowed to be a function of the model, which allows for adaptively defining the threshold based on the density of models in a given portion of the descriptor space. We take

$$P(\omega \approx \omega_x) = (1 - D(\omega_x, \omega_y))^2 \qquad (1)$$

to be an estimate of the probability that ω_x and ω_y represent the same type of object and therefore should have similar text tags. Then given our untagged model ω, a possible text tag λ^i, and a neighbor ω_x from the corpus, the probability that our query model should have the tag is

$$P(\lambda^i, \omega) = P(\omega \approx \omega_x) \wedge P(\lambda^i, \omega_x). \qquad (2)$$

Intuitively this means that the probability that λ^i is appropriate for ω is the probability that it is appropriate for ω_x and that ω and ω_x are similar enough to share tags. $P(\lambda^i, \omega_x)$ can be thought of as measuring how much we trust the original annotation on ω_x. When considered over the full

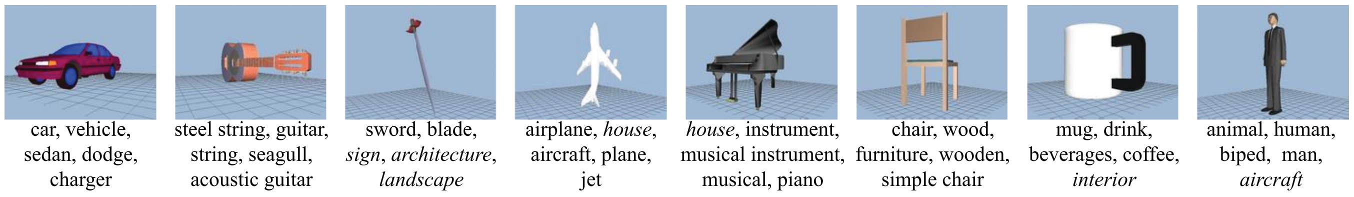

car, vehicle, sedan, dodge, charger
steel string, guitar, string, seagull, acoustic guitar
sword, blade, *sign*, *architecture*, *landscape*
airplane, *house*, aircraft, plane, jet
house, instrument, musical instrument, musical, piano
chair, wood, furniture, wooden, simple chair
mug, drink, beverages, coffee, *interior*
animal, human, biped, man, *aircraft*

Figure 1: Eight models from the PSB and their 5 best autotags. Tags we deemed to be incorrect are shown in italics. ("Seagull" is considered salient because it is a brand of guitar.)

set of neighbors N this generalizes to

$$P(\lambda^i,\omega) = \bigcup_{n=1}^{|N|} P(\omega \approx \omega_n) \wedge P(\lambda^i,\omega_n) \tag{3}$$

or equivalently by the sieve principle to

$$P(\lambda^i,\omega) = \sum_{n=1}^{|N|}(-1)^{n-1}\sum_{\substack{S\subset\{1,\ldots|N|\}\\ |S|=n}}\prod_{s\in S}(1-D(\omega,\omega_s))^2 P(\lambda^i,\omega_s). \tag{4}$$

2.1 Implementation

Our corpus consists of 192,343 models downloaded from Google 3DWarehouse. Each model has a title, a set of keywords, and a text description, although for many models one or more is blank. A good deal of the text is composed of nonsense words or blatantly incorrect labels. We found that the title and keyword fields were usually more reliable than the description, and so we assumed $P(\lambda^i,\omega) = 0.7$ for tags drawn from the title and keywords and $P(\lambda^i,\omega) = 0.5$ for tags drawn from the words of the description. Tags were stemmed using WordNet [3] and words that appeared on a list of stop words were ignored.

For the geometric similarity distance we used the Euclidean distances between Zernike descriptors [8], computed on a voxel grid of 128 voxels per side with a binary thickening kernel 4 voxels in diameter. For scaling, we used 7 point Gaussian numerical integration to find the center of mass of a uniform mass distribution on the surface of the object. Further integrations found the mean distance and standard deviation from surface points to the center of mass. We scaled so that the mean distance and 3 standard deviations fit within the unit sphere and clipped anything that lay outside. Scaling in this fashion is robust to moderate changes in shape and to outliers. We voxelized our models using a fast software voxelizer which we wrote, and computed the descriptors using a tuned version of Novotni and Klein's publicly available reference implementation. Following their recommendation, we used 20 levels of moments, which resulted in 121 dimensional descriptors. We performed a PCA over the descriptors of the 3DWarehouse data and kept only the top 57 dimensions. This preserved 99.9% of the original variance and led to much faster neighbor search, as described in Section 4.

3. EXPERIMENTAL VALIDATION

To validate the quality of our automatically produced tags, we used the Princeton Shape Benchmark (PSB) [11]. We computed Zernike descriptors for every model in the PSB, matched them against the models in our 3DWarehouse corpus, and tagged them using our algorithm. In computing tags we treated the PSB as if it consisted of completely untagged models. For $\tau(\omega)$ we used an adaptive threshold, which we defined as the radius of the hypersphere containing the first 15 nearest neighbors. Fig. 1 shows the results for eight models, where we have examined the autotags and italicized those we deemed to be incorrect.

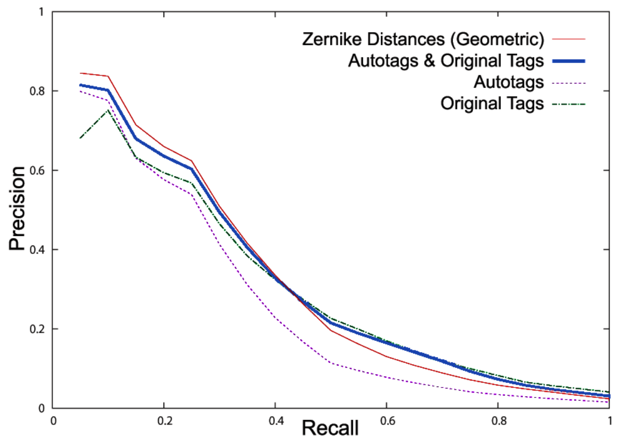


Figure 2: Precision/recall over the PSB for tag distances on autotags, Zernike descriptor distances, tag distances on the original tags of the PSB, and tag distances on a combination of the original tags and autotags.

3.1 Discriminative Power

Our first experiments were designed to test how *discriminative* our tags are, in the sense that models that belong to the same class in the PSB were given similar tags, and models in different classes were given dissimilar tags. We used the Vector Space Model [9] to define a "tag distance" between models. In the Vector Space Model, every possible tag $\lambda_i \in \Lambda$ is assumed to be an independent dimension, and the tags for a model ω form a vector in Λ-space. The length of the vector along each dimension λ^i is given by the "tag frequency, inverse document frequency" method (tf-idf) [10]. The distance between the tags of two models is 1 - the cosine of the angle between the tag vectors, or 1.0 (the maximum possible distance[1]) if either model is untagged.

Using this tag distance, we computed the distance matrix for the models of the PSB. Fig. 2 shows the precision/recall graphs of our autotag distances as compared to Zernike descriptor distances. It is important to remember that the autotag results are for text search, while the Zernikes re-

[1]Since tf-idf weights are always nonnegative the cosine must lie within (0,1).

quire an input 3D model. Although the Zernike descriptors are more discriminative, our algorithm still captures much of the power of the underlying shape descriptor and makes it accessible via keyword search.

As a control, we compared the quality of our computed tags to the original tags that came with the PSB models, using the method of [6]. Like them, we used seven sources of text for each model, including the model's filename, original URL, text from the referring webpage, and synonyms from WordNet. We formed tag vectors as we did for the autotags and calculated the tag distances. Fig. 2 shows that the initial precision of our tags is significantly superior to that of the original tags. For most models the original tags and the autotags are not identical, and so we can combine both sets of tags into a single tag vector. As Fig. 2 also shows, the combination of original and computed tags outperforms either tag source alone. In fact, the results are quite close to the precision/recall of the Zernike descriptors. We feel that this result is strong evidence against the notion that text based search can never compete with other forms of 3D search such as sketches and 3D query models.

3.2 Search Quality

Our first experiments confirmed that our tags are reasonably consistent within a class. However, nothing was said about the saliency of the tags; tagging all "houses" with the keyword "car" is consistent but not very useful. For our second set of experiments we tested tag quality by simulating example keyword searches for models in the PSB. We evaluated the searches for our autotags and the original tags. The queries were chosen to map directly onto classes in the PSB classification, so that we could evaluate the precision and recall of the results.

Given a search query λ, we returned the models ω_x that were tagged with λ, ordered by descending $P(\lambda, \omega_x)$. For the original tags, we weighted all tags equally, since we have no probability information for them.[2] Figure 3 shows the precision/recall for the queries "airplane," "head" and "sword" where we have capped the recall at the point where there are no more models tagged with the query string, and so any further retrieval would be random. Note that the precision of the autotags is equal or greater to that of the original tags nearly always. Perhaps more importantly, the autotags can successfully recall 60% to 75% of the relevant results for each query, while the recall for queries on the original tags capped out at 5%, 10% and 45%. The greater recall of the autotags demonstrates that our algorithm can assign usable, salient tags to 3D models, extending the reach of text search to models that were previously unreachable.

4. SEARCH ENGINE

We have implemented a shape search engine that uses autotagging. Our search engine has access to copies of 3DWarehouse and the PSB and can find models by geometric similarity, original tags, or autotags.

In Section 2.1 we described how we used PCA to reduce the 121 dimensional Zernike descriptors to 57 dimensions. PCA packs as much variance as possible into the first few dimensions, which allowed us to build a very fast k-nearest-neighbors implementation. This is the core of our search engine, since we need to find neighbors in Zernike descriptor space in order to do autotagging.

[2]In Figure 3, the apparently increasing precision for "sword" on the original tags is due to the random retrieval order for models with the same tag weight.

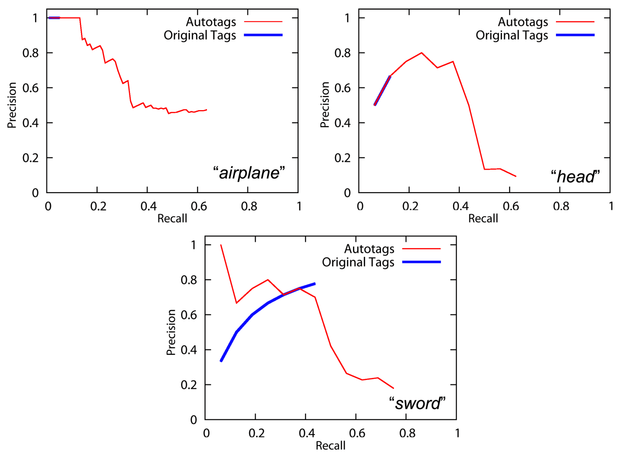


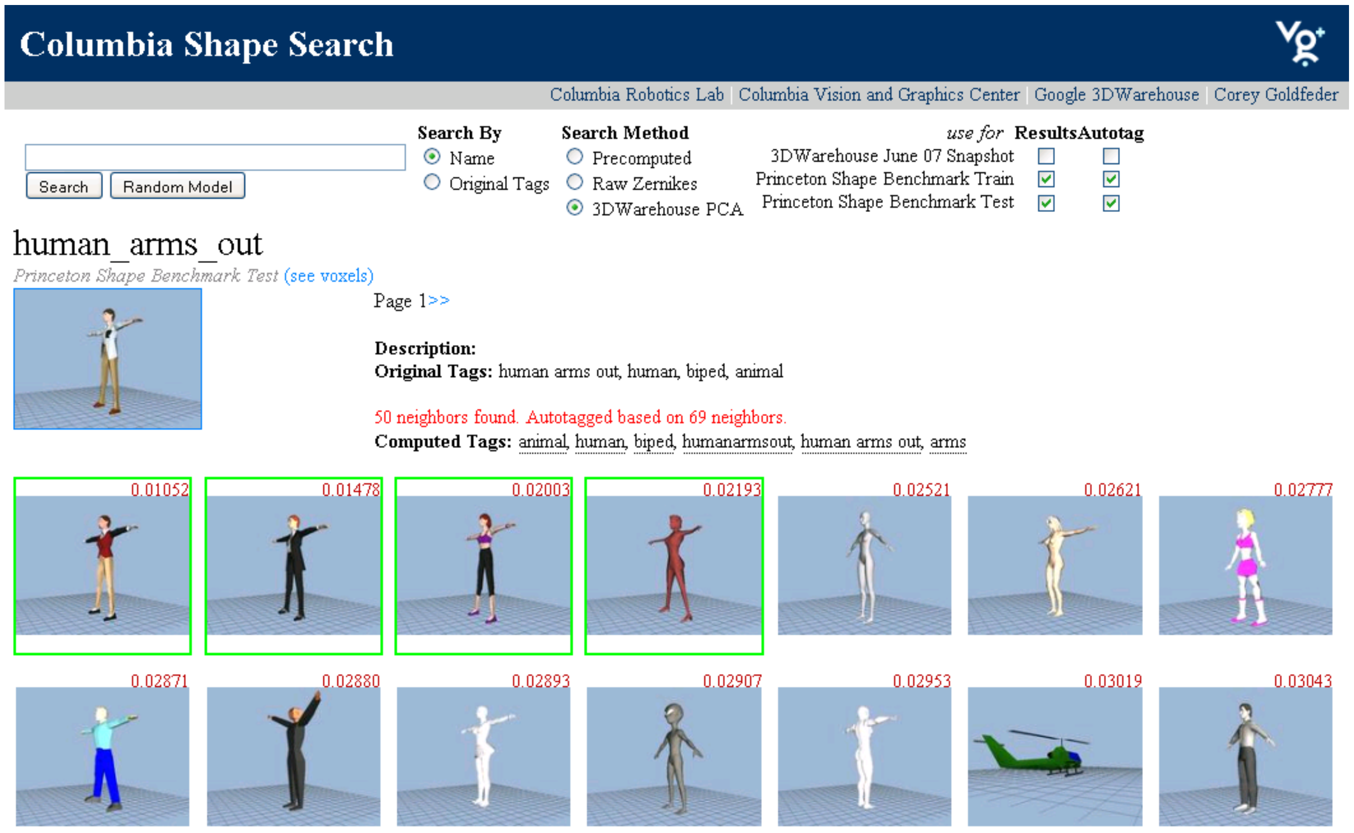
Figure 3: Simulated searches for "airplane," "head," and "sword".

Our approach is based on [7]. To find all of the neighbors of p within radius r they first prune the space to a hypercube with sides of $2r$, centered on p. To support this operation, they maintain n separate lists of the points, each sorted along one dimension. Pruning to a hypercube then reduces to rejecting any points with a distance greater than r in any one dimension, and then finding the intersection of n lists. The points which remain are brute force searched, and those which lie outside of the radius r hypersphere are rejected. If k neighbors are not found, the algorithm can be run again with a larger value of r.

We mapped this algorithm to a PostgreSQL database implementation. Instead of n lists, we maintain a table of n columns, where each row represents a single n-dimensional point. We also maintain an index on each column, which is algorithmically equivalent to maintaining a sorted list on each dimension. With this schema, we can perform the entire [7] pruning algorithm as a single SELECT statement with BETWEEN constraints on each dimension. Due to the PCA step most of the variance is in the first few dimensions, allowing PostgreSQL to prune dimensions with higher variance first. Columns with a higher variance are likely to have fewer neighbors within the search distance, and so most rows are pruned very early and do not need to be repeatedly considered for intersection. In practice, our implementation running on a 2.4 GHz Intel CPU can search 192,343 Zernike descriptors and return the 50 nearest neighbors of a query descriptor in approximately 5 seconds. Figure 4 shows the user interface of our search engine.

5. CONCLUSIONS

We have demonstrated an automatic tagging system that learns new tags for a 3D model by comparing it to a large set of tagged models and probabilistically propagating tags from neighbors. We have shown that the discriminative power of these tags is comparable to that of the underlying geometric similarity distance, and that searching for models based on our autotags can result in better precision and greater recall than searching on the original tags.


Columbia Shape Search

Columbia Robotics Lab | Columbia Vision and Graphics Center | Google 3DWarehouse | Corey Goldfeder

Search By
[✓ Name]
Original Tags

Search Method
Precomputed
Raw Zernikes
[✓ 3DWarehouse PCA]

use for	**Results**	**Autotag**
3DWarehouse June 07 Snapshot	☐	☐
Princeton Shape Benchmark Train	✓	✓
Princeton Shape Benchmark Test	✓	✓

human_arms_out

Princeton Shape Benchmark Test (see voxels)

Page 1>>

Description:
Original Tags: human arms out, human, biped, animal

50 neighbors found. Autotagged based on 69 neighbors.
Computed Tags: animal, human, biped, humanarmsout, human arms out, arms

Figure 4: Our web based search interface for keyword and geometry search.

Although we have focused in this paper on autotagging to improve shape *retrieval* in a digital library, there are several other domains where automatically annotating 3D models can be helpful. For example, when users submit models to a public digital library such as Google 3DWarehouse, they are often asked to supply tags for the models. If we can autotag models immediately, we can suggest tags that already exist on other models, which could improve the consistency of annotations in the library.

Our results are highly dependent on the quality of the corpus we use, in terms of both tag quality and coverage of the space of 3D models. In choosing 3DWarehouse as our corpus we have emphasized coverage over tag quality. We have experimented with autotagging using a smaller hand-classified corpus [5], and in future work we will examine the tradeoff between corpus accuracy and size.

6. ACKNOWLEDGMENTS

The authors wish to thank Google for providing a complete copy of 3DWarehouse in .obj format.

7. REFERENCES

[1] D. Y. Chen, M. Ouhyoung, X. P. Tian, and Y. T. Shen. On Visual Similarity Based 3D Model Retrieval. *Eurographics* 2003.

[2] R. Datta, W. Ge, J. Li, and J. Z. Wang. Toward Bridging the Annotation-Retrieval Gap In Image Search by a Generative Modeling Approach. *Multimedia*, 2006.

[3] C. Fellbaum. *Wordnet: An Electronic Lexical Database*. MIT Press, 1998.

[4] T. Funkhouser, P. Min, M. Kazhdan, J. Chen, A. Halderman, D. Dobkin, and D. Jacobs. A Search Engine for 3D Models. *ACM Transactions on Graphics*, 22(1), 2003.

[5] C. Goldfeder, H. Feng, and P. Allen. Training Set Expansion via Autotags. *Shape Modeling International: SHREC Shape Retrieval Contest*, 2008.

[6] P. Min, M. Kazhdan, and T. Funkhouser. A Comparison of Text and Shape Matching for Retrieval of Online 3D Models. *European Conference on Digital Libraries*, 2004.

[7] S. A. Nene and S. K. Nayar. A Simple Algorithm for Nearest Neighbor Search in High Dimensions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(9), 1997.

[8] M. Novotni and R. Klein. 3d Zernike Descriptors for Content Based Shape Retrieval. *Solid Modeling and Applications*, 2003.

[9] G. Salton. Mathematics and Information Retrieval. *Journal of Documentation*, 35(1), 1979.

[10] G. Salton and C. Buckley. Term-weighting Approaches in Automatic Text Retrieval. *Information Processing and Management*, 24(5), 1988.

[11] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser. The Princeton Shape Benchmark. *Shape Modeling Applications*, 2004.