Abstract: A digital library for non-textual, multimedia documents can be defined by its functionality: markup, indexing, and retrieval. For textual documents, the techniques and algorithms to perform these tasks are well studied. For non-textual documents, these tasks are open research questions: How to markup a position on a digitized statue? What is the index of a building? How to search and query for a CAD model? If no additional, textual information is available, current approaches cluster, sort and classify non-textual documents using machine learning techniques, which have a cold start problem: they either need a manually labeled, sufficiently large training set or the (automatic) clustering / classification result may not respect semantic similarity. We solve this problem using procedural modeling techniques, which can generate arbitrary training sets without the need of any "real" data.

The retrieval process itself can be performed with any method. In this article we describe the histogram of inverted distances in detail and compare it to salient local visual features method. Both techniques are evaluated using the Princeton Shape Benchmark (Shilane et al., 2004). Furthermore, we improve the retrieval results by diffusion processes.

1 INTRODUCTION

Techniques for digitizing 3D shapes are currently becoming available to a wide audience. With more and more data, questions of storage and archival arise – especially for generalized documents (Fellner et al., 2007). They should be treated analogue to ordinary text documents, so that they can be inserted into a digital library, which can be defined by its main functions: indexing and retrieval (Havemann et al., 2012).

This paper presents and combines three important ideas for content-based retrieval:

1. a new 3D model retrieval meta approach based on generative modeling techniques to eliminate the cold start problem (Ullrich and Fellner, 2011);
2. the method histogram of inverted distances by (Grabner et al., 2014) to measure the similarity of two 3D models;
3. an approach to improve retrieval results using diffusion processes (Donoser and Bischof, 2013).

All these ideas and techniques are described in detail and are combined to one retrieval system for 3D models.

A generative model describes a family of shapes, a so-called shape space. We use this shape space to randomly draw a number of samples. These training samples are passed to a machine learning based classifier. Without needing any "real" data, our method uses solely generative 3D models in the training phase. Consequently, it eliminates the cold start problem. The generative models themselves are represented as JavaScript code, which takes a number of parameters and returns a 3D model.

The used classifier is called histogram of inverted distances. It is a geometry-based method that operates on volume models. In a preprocessing step, the method converts the training samples to a voxel representation, aligns their principal axes to the canonical, Euclidean basis and calculates the inverted distance transformation of the volume model. The volume model is then split into a regular grid of cubes, which we call cells. For each cell we determine the histogram of inverted distances and learn a non-parametric density function (Szeliski, 2010), (Bishop, 2007). The object class is represented by its generative model and its learned density function. In the recognition phase the similarity is estimated using the learned density function of the object class to test.
The similarity of the whole model is given by the product of all cell similarities. This method is able to find similar objects. To illustrate its applicability, we compare it to the established salient local visual features method (Ohbuchi et al., 2008).

Using diffusion processes the retrieval results can improved even further. The similarity between two models can be calculated. If each model is compared to each other, the resulting square matrix of differences/similarities is called the affinity matrix. Retrieving 3D models similar to the \(i\)-th model using the affinity matrix only, is done by extracting the \(i\)-th row and sorting this row by its values.

### 2 RELATED WORK

Our approach combines techniques of shape description and generative modeling with content-based retrieval and machine learning.

The main idea of generative modeling is the description of shapes using algorithms. An overview on generative modeling techniques can be found in the survey by Watson and Wonka (2008), in the overview by Vanegas et al. (2010), as well as in the tutorial notes “The Rules Behind – Tutorial on Generative Modeling” (Krispel et al., 2014).

Concerning content-based retrieval, many methods for 3D models have been proposed recently. Tangelder et al. (2008) and Bustos et al. (2007) have both surveyed literature on content-based retrieval methods. All methods can be classified according to three categories: feature-based, graph-based and geometry-based methods. Feature-based methods operate on global (e.g. volume, area) or local (e.g. curvature) properties. Graph-based methods calculate a graph, such as a skeleton, based on the 3D model and perform matching based on graphs. Geometry-based methods operate directly on the models’ geometric representation. For detailed explanation and representative algorithms we refer the reader to the surveys mentioned above.

For the training phase, the above mentioned methods need a given sample set. This introduces a cold start problem: a sufficiently large data set has to be tagged and classified manually (for training purposes) or an unsupervised learning approach is used, which may result in a classification that does not correspond to the objects’ semantics or the classification used in the respective field of application. Ullrich and Fellner (2011) circumvent this problem by fitting generative models to the test data, so only the generative models must be known in advance. We use the same technique to span a shape space and to take a sample set by random. This randomized subset is the input of the training phase which uses histograms.

Shape histograms have previously been used by Ankerst et al. (1999) to classify molecules; however their approach uses one global histogram per molecule. Kriegel et al. (2003) also split their voxelized models into a regular grid of cells and calculate features vectors per cell. In contrast to our approach, they do not use the histogram of inverted distances. The inverted distance transformation is used by Dutagaci et al. (2005), and Kazhdan et al. (2003), but they use discrete fourier transformation based descriptors, radial cosine transformation based descriptors or spherical harmonic representation based descriptors for detection.

### 3 SHAPE DESCRIPTION

A possibility to describe a shape is realized by the generative modeling paradigm (Ozkar and Kotsoopoulos, 2008), (Ullrich et al., 2010). The key idea is to encode a shape with a sequence of shape-generating operations, and not just with a list of low-level geometric primitives. In its practical consequence, every shape needs to be represented by a program, i.e., encoded in some form of programming language, shape grammar (Müller et al., 2006), modeling language (Havemann, 2005) or modeling script (Autodesk, 2007).

Based on this idea each class of objects is represented by one algorithm \(M\). Furthermore, each described object is a set of high-level parameters \(x\), which reproduces the object, if an interpreter evaluates \(M(x)\). As this kind of modeling resembles programming rather than “designing”, it is obvious to use software engineering techniques such as versioning and annotations. In this way, model \(M\) may contain a human-readable description of the object class it represents. In order to train a machine learning approach, the parameter domain \(D(M)\) is sampled randomly using a uniform distribution. These random models \(M(x_i)\) are used in the training phase.

The generative models are implemented in JavaScript and interpreted on-demand. The scripts must implement the functions `shapeName`, `shapeDomain` and `shape`. The first function returns the name of a model, for example “car”; the second function returns an array that contains the lower and the upper bounds for every model parameter; the `shape` function takes an array containing numerical values and returns concrete 3D geometry represented by an indexed face set.

The generative models illustrated in Figure 1 represent the object classes “sedan car” (left) and “com-
commercial airplane” (right). The car model takes six parameters, whereas the plane model takes five parameters. Both models generate and return a 3D model with a fixed topology and varying geometry.

4 CONTENT-BASED RETRIEVAL

Histogram of Inverted Distances: For the training, all training models are converted to a voxel representation, scaled to a common size and aligned using Principal Component Analysis (PCA) as illustrated in Figure 2.

Afterwards, for each aligned training model \( T_i \) the inverted distance models are computed. The volume of the distance transformed training samples is defined by

\[
v(T_i, x, y, z) = \max \left\{ \frac{\text{cut} - d(T_i, x, y, z)}{\text{cut}}, 0 \right\},
\]

with the Euclidean distance \( d(T_i, x, y, z) \) of point \( (x, y, z) \) to the model’s surface and a threshold value \( \text{cut} \). Due to the voxelization the inverted distances can be calculated even for non-manifold and non-watertight models. Any 3D geometry representation (point clouds, NURBS, etc.) can be used as input data. Figure 3 shows the cross section of an aligned voxelized training model (left) as well as its corresponding inverted distance transformed version (right).

In the next step several voxels are combined to one cell \( C_i \) and within each cell the histogram of inverted distances with \( k \) bins is calculated. Having normalized the histogram, it serves as a \( k \)-dimensional feature vector \( h_i \in [0, 1]^k \) of that cell. Based on the feature vectors \( h_i \) we estimate a non-parametric density function for each cell position \((a, b, c)\) using Gaussian kernel density estimation (Bishop, 2007). The density...
function for a cell at position \((a, b, c)\) is
\[
P(h'_{a,b,c}) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{(2\pi\sigma^2)^{3/2}} \exp\left(-\frac{||h'_{a,b,c} - h_i||^2}{2\sigma^2}\right),
\]
where \(h'\) is the feature vector of a test model and \(\sigma\) represents the standard deviation of the Gaussian kernel. Usually the standard deviation can be estimated easily using appropriate estimation methods (Jones et al., 1996).

In the recognition phase, the model to test is processed the same way each training sample has been processed: it is voxelized, scaled, PCA-aligned, distance transformed, combined to cells, and converted into a \(k\)-dimensional feature vectors (for each cell) using normalized histograms.

Let \(X\) be a test model and \(h'_{a,b,c}\) denote the feature vectors of the test model, then the joint probability of model \(X\) belonging to the learned class is \(\prod_{(a,b,c)\in (1...p)} P(h'_{a,b,c})\). We call this algorithm the histogram of inverted distances (HID) algorithm.

**Salient Local Visual Features Method:** To demonstrate that the generative training approach presented in the previous Section can be combined with different retrieval techniques, we implemented the salient local visual features method introduced by Ohbuchi et al. (Ohbuchi et al., 2008). The method is depicted schematically in Figure 4.

Like in the histogram of inverted distances, the visual word histogram is learned using kernel density estimation. However, in this case Manhattan distance \(L_1\) is used as the kernel function. The similarity \(S\) for the 3D model described by the visual word histogram \(l'\) is given by:
\[
S(l') = \frac{1}{n} \sum_{i=1}^{n} L_1\left(\frac{l' - l_i}{\sigma}\right),
\]
where \(n\) denotes the number of training samples and \(\sigma\) represents the smoothing factor.

The visual codebook quantizes visual features into visual words. The visual codebook is learned unsupervised in a preprocessing step using \(k\)-means++ clustering (Arthur and Vassilvitskii, 2007). The set of visual features that have to be clustered is selected randomly from all views of the 3D models.

## 5 Evaluation

We evaluated both retrieval methods with the Princeton shape benchmark (Shilane et al., 2004) using the two classes illustrated in Figure 1: “commercial airplane” and “sedan car”. The complete benchmark consists of 907 test samples including 10 sedan cars and 11 commercial airplanes. Both retrieval methods have been trained generatively without any “real-world” data or any previously defined or marked test samples. The benchmark has been executed on a computer with an Intel 7 950 CPU and 12 GB RAM. All timings mentioned in this Section have been taken on this reference system.

**Sedan Car.** The “sedan car” class was modeled using JavaScript taking six parameters. All other dimensions (e.g. length of the top) are derived combining multiple parameters. The car consists of one side (colored green in Figure 1 (left)). This side is extruded twice using tapered extrusions with different angles. The first extrusion creates the blue parts and the second one creates the red parts. By mirroring the side and the two extrusions around its center plane the whole car is created. The tires are modeled using cylinders. The extrusion as well as the creation of a cylinder are extracted JavaScript functions prepared for reuse.

**Commercial Airplane.** The procedural model of the “commercial airplane” has five parameters to control the dimensions of the airplane (see Figure 1 (right)). Dependent parameters – such as the length of the wings – are combinations of free parameters. The fuselage of the airplane is created using a rotational surface which is then slightly deformed to model the cockpit and the aft fuselage. The jet engines are simple cylinders. As this generative model can already
reuse procedures needed for the sedan car, its code is shorter than the car’s description.

Evaluation of the Histogram of Inverted Distances Method. The values for the retrieval parameters were evaluated empirically. All parameters and their corresponding values are displayed in Table 1. The generation and voxelization of 64 models for both model classes took 103 seconds. The learning phase took 292 seconds. The actual retrieval process for both classes took about 4.75 hours, respectively 9.41 seconds per test model in average.

Figure 5 shows the precision-recall graphs (Manning et al., 2008) for cars and airplanes using the histogram of inverted distances method (left) and using the salient local visual features method (right).

The histogram of inverted distances method is able to find similar objects to the given generative models. The retrieval results for the class “sedan car” are almost perfect. However, the “commercial airplane” class performs significantly worse. The main reason for this effect can be found in the generative model. It is not detailed enough to differentiate between the benchmark’s different airplane classes (commercial, fighter, biplane, …). Figure 6 shows the top 16 retrieval results for both classes using inverted distances.

Evaluation of the Salient Local Visual Features Method. The parameters for the salient local visual features method are presented in Table 2.

Rendering of the range images, extraction of the visual features and calculation of the visual word histogram took 15.5 seconds per 3D model on average. The calculation of the visual codebook, which consists of the extraction of 30000 visual features from a random subset of the benchmark and the clustering of the features into 1024 clusters, took 72 minutes. The generation of 64 training models for both model classes and the calculation of the visual word histograms took 33 minutes. The actual retrieval process took 0.55 seconds per class.

Figure 5 (right) shows the precision-recall graphs

Table 1: The parameters and their corresponding values listed in this table have been used for benchmarking the histogram of inverted distances method.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>64</td>
</tr>
<tr>
<td>$R$</td>
<td>256</td>
</tr>
<tr>
<td>$p$</td>
<td>16</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>$1/4$</td>
</tr>
<tr>
<td>$cut$</td>
<td>16</td>
</tr>
<tr>
<td>$k$</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 2: The parameters and their corresponding values listed in this table have been used for benchmarking the salient local visual features method in combination with the generative training approach.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>64</td>
</tr>
<tr>
<td>$R$</td>
<td>256</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>$7/10$</td>
</tr>
<tr>
<td>$w$</td>
<td>1024</td>
</tr>
</tbody>
</table>
for both shape classes using the salient local visual features method. The method is able to find similar objects to the given generative models. Again, the “commercial airplane” class performs significantly worse than the “sedan car” class due to its generative model, which is not detailed enough to differentiate between the airplane classes “commercial”, “fighter”, etc. as can be seen in Figure 7. It shows the top 16 retrieval results for the classes “sedan car” and “commercial airplane”.

6 IMPROVED RETRIEVAL VIA DIFFUSION PROCESS

In order to improve retrieval results we use diffusion processes. Using matching methods for 3D models, the similarity or affinity between two models can be calculated. Assuming that \( n \) is the number of 3D models, the so-called affinity matrix \((n \times n)\) of all 3D models can be calculated. Retrieving 3D models similar to the \( i \)-th model using the affinity matrix only, is done by extracting the \( i \)-th row and sorting it by its affinity values. However, doing so ignores the structure of the underlying data manifold (Donoser and Bischof, 2013). Diffusion processes re-evaluate the affinities of all models in the context of all other elements. This is done by diffusing the affinity values through the graph described by the affinity matrix. Donoser et al. (2013) have recently surveyed diffusion processes for retrieval.

One of the best known diffusion algorithm is the Google page rank algorithm (Page et al., 1998). For the page rank algorithm an affinity matrix \( A \) is needed, which contains all pairwise affinity values. If each row of \( A \) is divided by its sum, the result is a stochastic matrix \( P \), which can be interpreted as a transition matrix for randomly walking on a complete graph of size \( n \).

Assuming we want to retrieve 3D models similar to the \( j \)-th model, the \( n \)-dimensional probability vector \( f_0 \) must be initialized with Kronecker \( \delta_{ij} \). The probability vector, which contains the final affinities for all other 3D models can be calculated iteratively \( f_{r+1} = f_r \cdot P \). This update operation is performed until convergence.

To apply diffusion processes in the field of retrieval, \( n \times n \) affinity matrix containing all pairwise similarities is needed. The pairwise similarities are calculated by comparing the cell histograms of the models. As described by (Johnson and Hebert, 1999), the similarity between two cells is given by the correlation coefficient. The similarity between two models is given by the sum of the cells similarity. The construction of the affinity matrix for the whole Princeton Shape Benchmark requires 18142 model comparisons. One comparison of two models took 0.34 seconds on average, which results in \( \approx 310 \) hours for the whole benchmark. To speed up the calculation of the affinity matrix, the workload has been distributed to seven similar computers. Thereby, calculation time was reduced to two days. From the affinity matrix for the whole benchmark, the affinity matrix for the test set was extracted. It is depicted in Figure 8 (left) using a the black body radiation color map. The resulting affinity matrix is depicted in Figure 8 (right).

The effect of the diffusion process reinforces retrieval effects; i.e. the precision slightly decreases at lower recall values, but increases at recall values of above 0.4.

7 CONCLUSION

This paper presents a new approach to perform content-based retrieval of 3D shapes based on generative modeling techniques. The generative models are used to describe 3D model classes, respectively, 3D shape spaces. In the training phase, the shape spaces are sampled randomly. In this way, no “real” training data is needed a priori.

The big advantage of procedural modeling techniques is the included expert knowledge within an object description (Ullrich and Fellner, 2011); e.g. the knowledge of an expert about the inner structure and the semantics of an object class can be mapped to procedures (Ullrich et al., 2013). Within the Cultural Heritage (CH) project “Procedural Fitting Server (ProFitS)” we incorporate this technique to index a CH repository semantically using expert knowledge. The approach of a generative training set, which does not need any “real” data can be combined with various retrieval algorithms. We have evaluated two retrieval methods to illustrate this approach.

The first method is called the histogram of inverted distances method by (Grabner et al., 2014). The second method is called the salient local visual features method based on the SIFT algorithm. Both methods use feature vectors to learn a non-parametric density function for each 3D model class. In the recognition phase, the feature vector is calculated for the test object and the similarity is estimated using the learned non-parametric density function.

Furthermore, the retrieval results can be improved using diffusion processes to take the underlying structure of the data manifold into account, similar to the Google page rank algorithm.

Our contribution to 3D documents is a shape retrieval approach based on machine learning and gen-
Figure 6: The top 16 retrieval results for the class “sedan car” using the histogram of inverted distances method (left) are almost perfect. The ten car models of the benchmark are listed within the best 16 matches, whereas top 16 retrieval results for the class “commercial airplane” using the same method have many false positives. Although all returned models are airplanes, the Princeton Shape benchmark distinguishes between many different airplane classes (commercial, fighter, biplane, ...).

Figure 7: These top 16 retrieval results for the classes “sedan car” (left) and “commercial airplane” (right) have been generated using the salient local visual features method. As the Princeton Shape benchmark distinguishes between many different airplane classes, the results for “commercial airplane” have many false positives.

Figure 8: The affinity matrix before (left) and after (right) the application of the diffusion process. The values are mapped using the black body radiation scheme. A black color indicates low similarity, whereas a white color indicates high similarity.
enerative modeling. In this way, we provide a classification technique, which uses generative modeling to encode expert knowledge in a way suitable for automatic classification and indexing of 3D repositories. We have shown that it is possible to train a retrieval method using generative models only. As a benefit (not only for users of our method), this technique eliminates the cold start problem in the training phase. A generative description implemented in a few lines of code is sufficient to generate a reasonable training set.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the support of the European Commission within the project DU-RAARK founded by the program “ICT-2011-4.3: Digital Preservation” as well as the support of the Austrian Research Promotion Agency (FFG) for the project ProFitS.

REFERENCES


