3D Shape Retrieval using Uncertain Semantic Query
A Preliminary Study

Hattoibe Aboubacar[1,2], Vincent Barra[1,2] and Gaëlle Loosli[1,2]
[1]*Clermont Université, Université Blaise Pascal, LIMOS, BP 10448, F-63000, Clermont-Ferrand, France*
[2]*CNRS, UMR 6158, Limos, F-63171 Aubiere, France*

Keywords: 3D Shape Retrieval, Uncertainty Coding, Semantic Query, SimpleMKL, SVDD.

Abstract: The recent technological progress contributes to a huge increase of 3D models available in digital forms. Numerous applications were developed to deal with this amount of information, especially for 3D shape retrieval. One of the main issues is to break the semantic gap between shapes desired by users and shapes returned by retrieval methods. In this paper, we propose an algorithm to address this issue. First the user gives a semantic request. Second, a fuzzy 3D-shape generator sketches out suitable 3D-shapes. Those shapes are filtered by the user or a learning machine to select the ones that match the semantic query. Then, we use a state-of-the-art retrieval method to return real-world 3D shapes that match this semantic query. This algorithm is used to retrieve object in SHREC'07 database. The results are good and promising.

1 INTRODUCTION

With the technological progress in computer graphics and object modeling, the number of available 3D data has grown exponentially. It makes it essential to develop effective methods and elegant modeling, analysis and processing tools to allow a better understanding of the phenomena involved and the interpretation of data.

The proposed work is closely linked to the 3D-shape retrieval problem, even though it doesn't propose a novel algorithm for this task (examples of retrieval methods can be found in (Tangelder and Veltkamp, 2008) and a brief description is given in section 3). Indeed, the target problem deals with the request that is given to any retrieval machine. Usually, this request is a 3D-shape and we propose here to start from a (textual) semantic request and help the user to obtain a 3D-shape that will be given to the retrieval machine.

In the next section, the black box between the request and the generated 3D-shape is detailed. Section 3 reviews the related work. Then, we detail the different tools used in our process in section 4. In section 5 we introduce some experiments. At last, some points are explained and discussed in section 6.

2 FROM THE SEMANTIC REQUEST TO A SUITABLE SYNTHETIC 3D-SHAPE

As mentioned above, the objective is to provide a tool that goes from a semantic request to a 3D-shape suitable to be an input for retrieval.

Semantic Request. This requires to define what will be considered as a semantic request : it is restricted to the association of a noun (*the class*) and an adjective (*the concept*). For this study, the nouns are limited to a set of known classes (although future work is planned to extend this) and the concept is free.

Suitable 3D-shape. Let's now define what a *suitable 3D-shape* is. Since we will use a retrieval machine based on Reeb graphs that are extracted from the 3D-shape, we consider that texture or details are meaningless and that the general shape and position are sufficient to define the semantic request. A suitable 3D shape will be a shape composed of very simple geometrical parts (like balls, cylinders, parallelograms) that looks like what the user meant.

Process. It is obviously not easy or even possible to define all concepts *a priori*. The choice here is to ask the user, as in retrieval feedback processes, to give

Aboubacar H., Barra V. and Loosli G..
3D Shape Retrieval using Uncertain Semantic Query - A Preliminary Study.
DOI: 10.5220/0004819106000607
In *Proceedings of the 3rd International Conference on Pattern Recognition Applications and Methods* (ICPRAM-2014), pages 600-607
ISBN: 978-989-758-018-5

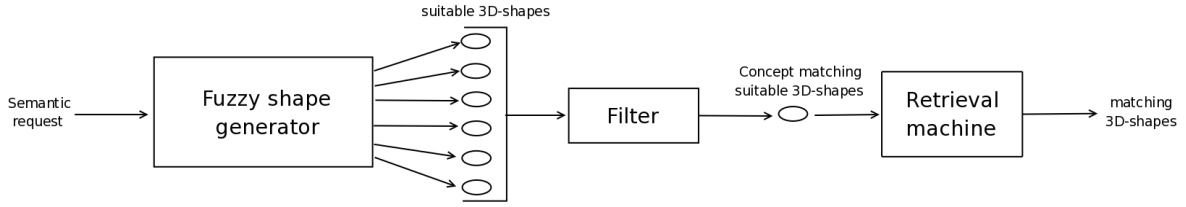

Figure 1: Overview - we propose a process to go from a semantic request to a suitable 3D-shape that can be used in a retrieval machine in order to retrive 3D-shapes matching the semantic request. We present a way to generate suitable 3D-shapes and examples of machine filter.

precision on his request by selecting matching shapes when a concept is unknown. The idea is to generate several instances of the classes, based on several uncertainties in the definition of the class (see section 4.1). This is the first contribution of this paper. Then, a filter is applied, in order to keep only instances of the desired concept. This is the second contribution. This filter takes two forms: the human one (manual selection of the generated shapes that match the concept) and the trained machine one (trained using previously selected shapes). The basic process is that the human filtering is required for any unknown concept and used to train the machine one. If a reliable filter has been trained for a concept, the human intervention is not required anymore.

Figure 1 gives an overview of our process.

Tools and Open Questions. To implement our process, we need to evaluate several points:

- the trained machine filter : since we'll get only good instances, we focus on one-class learning methods (here we present results for SVDD and SimpleMKL with One-class SVM) : see section 4.2 for details on the methods.
- the reliability of the machine filter : the question here is to evaluate the number of good instances that are necessary to train the filter (see experiment in section 6).
- how good are the generated 3D-shape to retrieve concept matching real 3D-shapes using a retrieval machine : see section 5.

3 RELATED WORK

Breaking the semantic gap between users and machines has been the subject of a huge amount of works in CAD and computer graphics in general. A brief review of state-of-art methods in reducing the semantic gap is given in (Han et al., 2008).

Recently, (Eitz et al., 2012) made a great contribution towards human's way of seeing images. It helped them to develop a powerful method to retrieve shape from sketch. It is an interesting way of dealing with uncertainty as sketches are a personal simplification of the way we see objects. We aim at developing a tool that allows user to sketch out the 3D shape they want like in the approach of (Eitz et al., 2012), moreover we want to be able to retrieve not only objects of the same class but shapes in a specified posture.

In this paper, we are mainly inspired by shape content-based methods. The recent progress made in this field is the introduction of relevance feedback methods. The idea behind relevance feedback is to approach the user's query by allowing him to specify it several times. In practice, the user gives his request. Then first objects that best match the query are displayed. Hence the user can state the ones he wants and those which are wrong. These choices may be reused to be more accurate on the retrieval process. Therefore based on the iterative feedback of the user better results can be obtained. (Giorgi et al., 2010) proposed a relevance feedback method based on the assumption that similarity may emerge from the inhibition of differences. In this approach, objects that are wrong according to the user are not integrated in the new retrieval process. (Zhang and Jin, 2010) used a learning technique to learn through the different responses of the user. This technique used a Gaussian kernel coupled with a method of gradient descent to improve the learning step. The results were very encouraging as these of (Giorgi et al., 2010).

Moreover, CAD-oriented works gradually and indirectly integrated uncertainty in the modeling of 3D shapes. (Funkhouser et al., 2004) was one of the first to propose the construction of models based on other existing models. Several algorithms for handling 3D shapes have been developed in this work including smart scissors which allow the user to easily separate parts of 3D shapes and paste one part to another using an efficient paste tool. The results are really remarkable and let the user play with his imagination. However, despite of a shape search module, the user must do a great job to find all forms of possible parts of objects. More recently (Chaudhuri et al., 2011) provides an effective response to this problem. The different parts of 3D shapes are automatically segmented and

classified as primitives in a relevant way according to position, size and other parameters. In addition, as the user manually creates a model, the method provides the user with primitives sorted by relevance semantics, from most relevant to least relevant, step by step. This work is subject to an automatic extension in (Kalogerakis et al., 2012), where 3D shapes are automatically synthesized using existing shapes and a probabilistic learning model. Experimental creation of planes, cars and boats have been made and the results are mind-blowing. Furthermore, the method allows different kinds of imaginable objects for each class to be computed and there is a great time saving as thousand of possible realistic models are created in a few minutes. It is a powerful tool, but the user still has to look at several models to find what he wants.

The process proposed in this paper is a complementary approach to that ones. Most of the time people does not think of one object but different possible objects. We think that trying to model the requested object from simple part before searching for real objects will help pass the hurdle. Our main intention is to explicitly deal with uncertainties in modeling, coding and processing. Thus, we will be able not only to design objects but ideas.

So far, there has not been yet a black box that captures the semantic request of a user and returns exactly what he wants. Our purpose is to provide tools to help taking a step towards this black box.

4 PROCESS DEPLOYEMENT

We introduce here all the different tools that are used, from 3D-shape modeling to the retrieval method, including a brief description of the one-class learning methods.

4.1 Modeling Process

We consider that most of 3D shapes can be robustly segmented. This brings the idea that a shape is a combination of several simple shapes joined according to some rules. Therefore, we model 3D shapes as a combination of simple shapes we call primitives. In order to introduce some uncertainties, these primitives can be altered by pattern maker called modifiers, before being pieced together according to random spatial relations. Doing this, we are able to design all possible postures of the same object. As a result, we create a fast and flexible 3D shapes generator which randomly produces uncertain shapes allowing the user to select the ones that correspond to his semantic query.

Here, we choose to model a simple manufactured class of objects: chair in section 4.1.1, and a more complex one: humanoid in section 4.1.2. To model these uncertain 3D shapes we use four primitives which are spheres, cylinders, parallelograms and ellipsoids. One has to note that our 3D models are quite simple and do not satisfy all real constraints inherent to each object. However, this is not annoying in our case because we want to learn semantic concepts and we assess that our semantic concepts only embed realistic models so our methods have to be robust with unrealistic ones.

4.1.1 Design of a Generic Chair

A chair is modeled as a combination of three primitives representing the back, the chair seat and the feet. We assume the feet are symmetrical and have the same characteristics, therefore only information on one foot is needed. Otherwise, we consider only two kind of primitives in our chair modeling process which are cylinder and parallelogram. Indeed, the back, the chair seat and the feet are each one independently from the other a cylinder or a parallelogram. In addition, characteristics of cylinder and parallelogram are randomly defined. Of course, we include in our implementation some constraints to ensure that our uncertain chairs are like realistic ones. These constraints ensure that primitives do not have abnormal size according to realistic characteristics of a chair. This is made using uniform distribution to draw the sizes under specified bounds. Furthermore, we make sure the back is linked with the chair seat and the feet are located below the chair seat. Then we randomly select angles to fix the orientation between the primitives.

Figure 2 shows some examples of designed chairs.

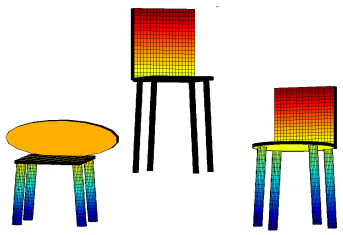

Figure 2: Examples of designed chairs.

4.1.2 Design of a Generic Humanoid

Humanoid are modeled using eleven body parts modeled as primitives: two arms, two forearms, two thighs, two legs, one chest, one neck and a head. We choose to model limbs and neck as cylinders, head as sphere and chest as ellipsoids. Quite naturally, the

motion of each limb is independent of other limbs. As for chairs, the characteristics of primitives are set up randomly to bring some size uncertainties. We start modeling a humanoid from the chest which is always vertical. Then we add neck and head which do not need to be oriented, so we just fix neck to chest and head to neck. We add limbs and orient each one independently which allows to obtain random postures. Furthermore, we ensure that corresponding limbs have the same characteristics as for realistic humanoid, and we enable independent orientation from one to another.

Figure 3 shows an example of designed humanoid.

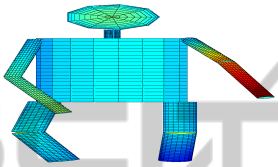

Figure 3: Example of designed humanoid.

4.1.3 Suitable 3D-shape Features

Each suitable 3D-shape is represented by its features in the learning process. The features of a suitable 3D-shape is the set of features of the primitives the shape is composed of, plus angles between primitives. A primitive is characterized by its type of shape and its size. Thus, there are sixteen features for a chair and forty four for an humanoid. Then, an harmonization step is performed in order to have the same number of features for each kind of shape in the data. It consists in repeating information on primitive in each model as much as needed to have the same number of primitives in both. After this step we then worked with a hundred and sixteen features for each shape. These features were the inputs used in our learning methods.

4.2 Learning Semantic Concept: Creating a Machine Filter

Learning a semantic concept from a sample of 3D-shapes is equivalent to learning the characteristics of objects supposed to belong to the same class. It is a standard problem in machine learning called one-class classification problem also known as problem of data description. We refer to (Khan and Madden, 2009) to have a glimpse at recent one class classification methods. We investigated two methods which address this problem in order to create a machine filter.

Here, we choose to use only Support Vector Machine (SVM) based methods. SVM-based methodology for one class classification was first introduced by (Schlkopf and Smola, 2001). Schlkopf's approach is to find an separation hyperplane between targets and outliers. This is the most widely used approach in the literature. We also concentrate here on another approach, proposed in (Tax and Duin, 2004), called Support Vector Data Description (SVDD). This method aims to find a hypersphere which contains all possible targets. This approach has proved to be efficient and SVDD is part of top methods when it comes to one-class classification. SVDD is detailed in section 4.2.1 and SimpleMKL in section 4.2.2.

When a user asks for a semantic concept. First, he has to choose manually a sample of suitable 3D-shapes that match his query. This sample is learned and saved using a learning method. Hence, the next time the same concept is asked, we use our fuzzy shape generator to produce several shapes. Then the same learning method is used to determine which shapes belong to this concept. The best matching shapes are the filtered suitable 3D-shapes.

4.2.1 SVDD

SVDD (Tax and Duin, 2004), solves a multidimensional outlier detection problem. The aim here is not to estimate a probability density like in standard methods, but rather to construct a spherically shaped boundary around the target set. Hence, the following optimization problem is solved:

$$\begin{aligned} &\text{Minimize} \quad R^2 + C\sum_i \xi_i \\ &\text{subject to} \quad \| x_i - a \|^2 \leq R^2 + \xi_i, \quad \xi_i \geq 0 \quad \forall i \end{aligned} \tag{1}$$

Where x_i are the targets, R is the radius of the spherically shapes boundary, ξ_i are slack variables and C controls the trade-off between the volumes and the errors.

One of the advantages of this method is that it allows the possibility of outliers in the training set. Then the optimization is modified as explained in (Tax and Duin, 2004). However, it still well performs. Furthermore, it is really flexible as one can choose to use his kernel function instead of the inner product used by default.

4.2.2 SimpleMKL

SimpleMKL is a Multiple Kernel Learning (MKL) method introduced by (Rakotomamonjy et al., 2008). MKL was first introduced in (Lanckriet et al., 2004), then it was subject to enhancements, extensions and algorithms to solve the problem in (Bach et al., 2004; Sonnenburg et al., 2006), before (Rakotomamonjy et al., 2008) propose a new formulation and a new algorithm. SimpleMKL uses the same approach as

traditional SVM but instead of looking for only one kernel it looks for several kernels and linear combinations of these kernels then finds the best one and optimizes it.

The idea behind MKL as summarized in (Rakotomamonjy et al., 2008) is to look for a different solution of the learning problem. Indeed in one class learning problem the solution is of the form

$$f(x) = \sum_{i=1}^{l} \alpha_i^* K(x, x_i) \tag{2}$$

where α_i^* are some coefficients to be learned from examples x_i and $K(.,.)$ is a given definite kernel associated with a reproducing kernel Hilbert space (RKHS), H. (Lanckriet et al., 2004) proposed to consider the kernel $K(.,.)$ as a convex combination of basis kernels

$$K(x,y) = \sum_{m=1}^{M} d_m K_m(x,y) \quad with \quad d_m \geq 0, \sum_{m=1}^{M} d_m = 1 \tag{3}$$

where M is the total number of kernels, K_m, which are classical kernels. Hence, MKL is learning both the coefficients α_i^* and d_m in a single optimization problem. To work out this problem, and following (Rakotomamonjy et al., 2008) that claims that their method can be extended to a one-class problem, we used the following extension of SimpleMKL:

$$\begin{aligned} &\underset{\{f_m\},\xi_i,d_m,\rho}{\text{Minimize}} && \frac{1}{2}\sum_m \frac{1}{d_m} \| f_m \|^2_{\mathrm{H}_m} + \frac{1}{\nu n}\sum_i^n \xi_i - \rho \\ &\text{subject to} && \\ & && \sum_m f_m(x_i) \geq \rho - \xi_i, \forall i, \\ & && \xi_i \geq 0, \forall i, \\ & && \sum_m d_m = 1, d_m \geq 0, \forall m, \end{aligned} \tag{4}$$

where n is the number of examples and ν the error allowed. H_m are the RKHS associated to each $K_m(.,.)$ and each function f_m belongs to a different H_m. It is assumed that one looks for a decision function of the form $f(x) = \sum_m f_m(x)$.

4.3 Retrieval

After learning the semantic concept, we retrieve real-world shapes that match this semantic concept using our designed suitable shapes. Here we introduce the method we use to retrieve real world 3D-shapes.

We perform retrieval with a recent unsupervised method for 3D retrieval using Kernels on Extended Reeb Graphs from (Barra and Biasotti, 2013). This method used some real functions to perform different Extended Reeb Graphs (ERG) of each 3D model. This family of Extended Reeb Graphs is considered as the graph representation of those 3D models. Hence, in order to assess similarity of 3D models, a Gaussian kernel similarity measure is defined on the set of graphs. The measure is based on a representation of graphs in terms of bag of shortest paths with bounded maximal length. It is used to determine similarity between ERG of the same real function. Therefore, the r nearest neighbors of a query 3D-shape, where r is an integer, can be computed according to the ERG similarity of each real function. Then, an aggregated function is used to sum up the outcomes for each real functions and solve the 3D retrieval problem, that is to say giving the overall r nearest neighbors. This method has been compared with state-of-the-art methods and performs very well, as most of the time it outperforms reference methods. We refer to (Biasotti et al., 2008) for Extended Reeb Graphs.

This method is the one we use to run experiments presented in this article.

5 EXPERIMENTS

In this section, we investigate how our process can be useful to retrieve semantic concept. Doing this, we learned three concepts (Figure 4) which correspond to the following semantic requests:

- high chair: chair with a seat raised at a fair distance from the ground or with high feet length.
- low chair: chair with small feet length.
- stand up humanoid (stand hum).

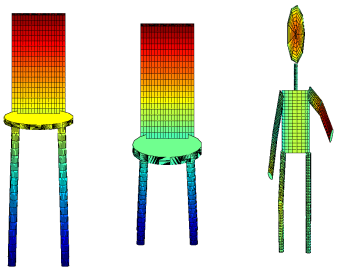

Figure 4: Examples of high chair, low chair and stand hum.

Then we used our algorithm to retrieve matching shapes in the database of the watertight track of the Shape Contest SHREC'07 (Giorgi et al., 2007). We give details of the comparison between our three learning methods in section 5.1. Section 5.2 gives some examples of designed shapes, afterwards retrieval outputs are detailed in section 5.3.

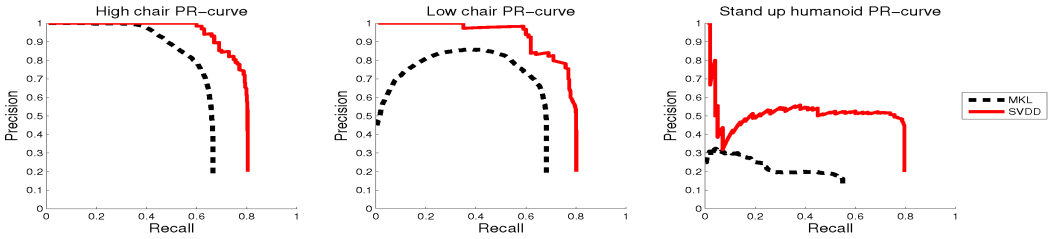

Figure 5: Precision/Recall graphs for each concept. Precision is the fraction of relevant retrieved objects to a given query, and recall is the fraction of relevant objects which have been retrieved from the database. If O is the set of relevant objects to the query, A is the set of objects retrieved, O_A is the set of relevant objects in the result set then $P = \frac{|OA|}{|A|}$ and $R = \frac{|OA|}{|O|}$.

5.1 Choice of the Learning Method

In order to test our algorithm, we compared the learning methods presented in section 4.2 to choose the best one to learn our semantic concepts. A cross validation was performed to determine which parameters were the best for each learning method. We worked with a training dataset of a hundred of shapes of each concept: stand up humanoids, low chairs and high chairs. The test dataset was composed of a hundred of instances of each concept and one hundred shapes of non stand up humanoids.

We assumed that a standard user will provide at most five instances representing the concept he wants to be learned. Therefore, for each concept we randomly partitioned our training data into twenty subsamples of five instances. Then, we repeatedly learned each subsample and test our precision using the remaining subsamples.

We only mention here the average error obtained for each method. Figure 5 shows the precision and recall graphs for each concept. SVDD has the best precision recall curves. It outperforms SimpleMKL for the three concepts. SimpleMKL performs well for both chair concepts but give poor performances for the stand up humanoid concept.

Table 1 summarizes the overall precision on the testing database. SVDD has the lowest error for high chair and stand up humanoid concepts. However, SimpleMKL is the best for the low chair concept. Both methods have approximately the same error for chair concepts but SVDD outperforms SimpleMKL for the humanoid concept.

Hence, based on these observations, we chose to use SVDD as our method to learn semantic concept.

Table 1: Percentage errors for each concept.

Concept	MKL	SVDD
High chair	12.58	12.29
Low chair	13.43	13.46
Stand hum	39.70	23.11

5.2 Suitable 3D-shapes

SVDD was run to select the three best matching objects from a random dataset of two hundred chairs and two hundred humanoids for each of our three concepts. Figure 6 shows the first three selected high chairs, figures 7 and 8 the first three selected shapes for respectively low chairs and stand hum.

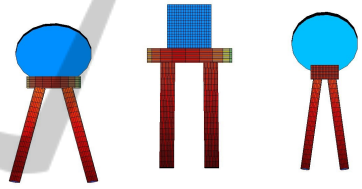

Figure 6: Examples of first three selected high chairs (ranked from left to right).

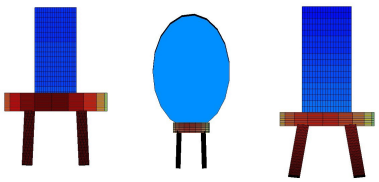

Figure 7: Examples of first three selected low chairs (ranked from left to right).

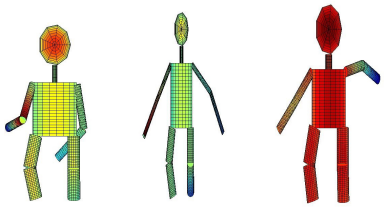

Figure 8: Examples of first three selected stand hum (ranked from left to right).

The outcomes are good according to what we expected. These shapes are used to retrieved object in the next section.

5.3 Retrieval Outputs

We did a retrieval in SHREC'07 database with the method of (Barra and Biasotti, 2013) to see what we can get from a thorough database with our designed shapes. We chose SHREC'07 (Giorgi et al., 2007) because it is a 3D-shape database which was manually established with enough variation in order to evaluate the effectiveness of 3D-shape retrieval algorithms. The collection is composed of 400 mesh models, subdivided into 20 classes of 20 elements each.

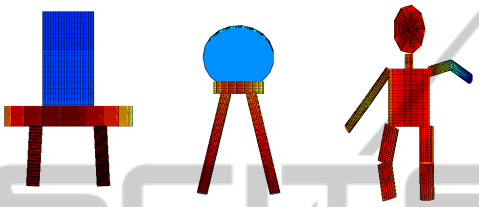

Figure 9: Instances used to retrieve 3D-shapes in SHREC'07.

The three shapes showed in Figure 9 were the one used to retrieve 3d-shapes. Figure 10 shows the first retrieved shapes for each concept. The retrieved

Figure 10: The first retrieved shapes for each concept.

shapes are on the top shapes we expected to have in our first retrieved one. This shows arguably that we can connect our suitable 3D-shapes with state-of-the-art designed one.

6 DISCUSSION

This paper presents results of a prototype phase. The learning and retrieval methods used were chosen not only because they are widely used in the literature and perform very well but also because they are easy and fast to implement. Readers can feel free to replace those methods by methods of their choice.

Similarly, here only chairs and humanoids are modeled but new objects can be added quickly because our method does not require thorough models.

Moreover, we said that we used five learning examples to build our filter because we thought that a user will arguably select at least five elements. However, there is another reason: we ran some experiments to know with how many learning examples we were able to build a reliable machine filter. Hence, we splitted up our learning database of one hundred examples in several set of the same number starting by set of one element to set of 50 elements. These process was done for each concept. Then, we did a cross validation for each set to determine the best parameters to use to learn with according to the number of element in the set. The same test data used in section 5 has been used to perform tests and we gathered the average learning error obtained for each number of learning examples. This was repeated for each concept and each learning method. Figure 11 shows the average test error according to the number of learning examples used to build the machine filter. The results show that we are able to build a reliable filter with less than five learning examples for each concept but the number of learning examples that performs the best depends on the learned concepts and the learning method used. For instance, when learning the high chair concept, SVDD has better performances with 30 learning examples and SimpleMKL with 10 learning examples. However, If we look at the error obtained when learning with less than five elements, the best performances are obtained when learning with three or five elements. We chose to learn with five examples but one may prefer to learn with only three.

Another issue is the number of semantic requests presented. This choice has been driven by the large degree of freedom of our generator. The probability to get shapes matching the concept the user wants is really low, even if we already know which class of object he wants, there are thousand of possible postures. One of our futur work is to develop a tool that allows the users to specify what he wants without choosing between thousand of models. However, theoreticaly user who wants to ask for 3D-shapes representing a new semantic request just needs to select a sample of 3D-shapes. Then those shapes are processed and analyzed as explained in the former section.

7 CONCLUSIONS

An algorithm to generate suitable 3D-shapes corresponding to a specific semantic concept is proposed in this article. The principle is to allow the user to specify his semantic query by selecting representing 3D-shapes designed by a 3D-shapes generator that integrates a part of randomness, enabling to explore the uncertainty characteristic of such a request. The concept behind the selected shapes is learned to create automatic filters for future usage and used to retrieve shapes in SHREC'07 database.

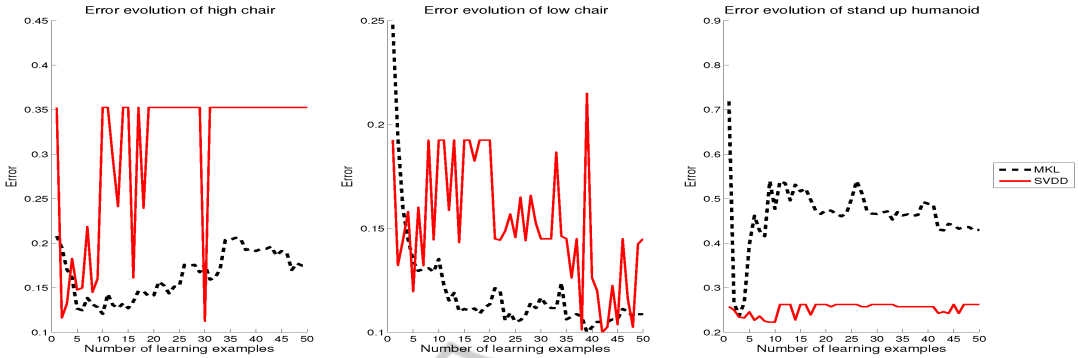

Figure 11: Evolution of the test error according to the number of learning examples used to build the machine filter.

We plan to make numerous improvements in our work. We are currently working on a method allowing the user to more easily select suitable shapes for concept matching. Moreover, we hope to run directly our algorithms to retrieve uncertain shapes.

REFERENCES

Bach, F., Lanckriet, G. R. G., and Jordan, M. I. (2004). Multiple kernel learning, conic duality, and the smo algorithm. In *Proceedings of the twenty-first international conference on Machine learning*, ICML '04, New York, NY, USA. ACM.

Barra, V. and Biasotti, S. (2013). 3d shape retrieval using kernels on extended reeb graphs. *Pattern Recognition*.

Biasotti, S., Floriani, L. D., Falcidieno, B., Frosini, P., Giorgi, D., Landi, C., L.Papaleo, and Spagnuolo, M. (2008). Describing shapes by geometrical-topological properties of real functions. *Computing Surveys*, 40(4). In: Computing Surveys, vol. 40 (4) ACM, 2008.

Chaudhuri, S., Kalogerakis, E., Guibas, L., and Koltun, V. (2011). Probabilistic reasoning for assembly-based 3d modeling. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 30(4).

Eitz, M., Hays, J., and Alexa, M. (2012). How do humans sketch objects? *ACM Transactions on Graphics (Proceedings SIGGRAPH)*, 31(4):44:1–44:10.

Funkhouser, T., Kazhdan, M., Shilane, P., Min, P., Kiefer, W., Tal, A., Rusinkiewicz, S., and Dobkin, D. (2004). Modeling by example. *ACM Transactions on Graphics (Proc. SIGGRAPH)*.

Giorgi, D., Biasotti, S., and Paraboschi, L. (2007). Watertight models track. Research report, IMATI, Genova, Italy.

Giorgi, D., Frosini, P., Spagnuolo, M., and Falcidieno, B. (2010). 3d relevance feedback via multilevel relevance judgements. *Vis. Comput.*, 26(10):1321–1338.

Han, D., W., and Li, Z. (2008). Semantic image classification using statistical local spatial relations model. *Multimedia Tools and Applications*, 39(2):169–188.

Kalogerakis, E., Chaudhuri, S., Koller, D., and Koltun, V. (2012). A probabilistic model of component-based shape synthesis. *ACM Transactions on Graphics*, 31(4).

Khan, S. S. and Madden, M. G. (2009). A survey of recent trends in one class classification. In *Artificial Intelligence and Cognitive Science*, pages 181–190.

Lanckriet, G. R. G., Cristianini, N., Bartlett, P., Ghaoui, L. E., and Jordan, M. I. (2004). Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72.

Rakotomamonjy, A., Bach, F., Canu, S., and Grandvalet, Y. (2008). Simplemkl. *Journal of Machine Learning Research*.

Schlkopf, B. and Smola, A. J. (2001). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA.

Sonnenburg, S., Rtsch, G., Schfer, C., and Schlkopf, B. (2006). Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7:1531–1565.

Tangelder, J. W. H. and Veltkamp, R. C. (2008). A survey of content based 3d shape retrieval methods. *Multimedia Tools and Applications*, 39(3):441–471.

Tax, D. M. J. and Duin, R. P. W. (2004). Support vector data description. *Machine Learning*, 54(1):45–66.

Zhang, Z. and Jin, J. (2010). Fuzzy relevance feedback in content-based 3d model retrieval. In *Proceedings of the seventh international conference on Fuzzy Systems and Knowledge Discovery*, pages 565–568.