Eurographics Symposium on Geometry Processing 2013
Yaron Lipman and Richard Hao Zhang
(Guest Editors)

Volume 32 (2013), Number 5

Dynamic Maps for Exploring and Browsing Shapes

Yanir Kleiman[1] Noa Fish[1] Joel Lanir[2] Daniel Cohen-Or[1]

[1] Tel Aviv University [2] Haifa University

Abstract

Large datasets of 3D objects require an intuitive way to browse and quickly explore shapes from the collection. We present a dynamic map of shapes where similar shapes are placed next to each other. Similarity between 3D models exists in a high dimensional space which cannot be accurately expressed in a two dimensional map. We solve this discrepancy by providing a local map with pan capabilities and a user interface that resembles an online experience of navigating through geographical maps. As the user navigates through the map, new shapes appear which correspond to the specific navigation tendencies and interests of the user, while maintaining a continuous browsing experience. In contrast with state of the art methods which typically reduce the search space by selecting constraints or employing relevance feedback, our method enables exploration of large sets without constraining the search space, allowing the user greater creativity and serendipity. A user study evaluation showed a strong preference of users for our method over a standard relevance feedback method.

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques

1. Introduction

3D object repositories have gone from being small and scarce to large and abundant, a change that led to the development of new ways to easily and intuitively search and explore these available collections. In recent years, the problem of shape retrieval and exploration has been a major focus of many [OLGM11, BBGO11, TGY*09]. Methods to determine the level of similarity between objects have been researched extensively [OFCD02, TV04]. Since in most cases the distinction between a relevant and irrelevant retrieved shape is user subjective, relevance feedback techniques were incorporated into many retrieval systems, allowing the user to guide the search according to personal preference and taste [LMT05].

Generally, relevance feedback involves presenting the user with a set of suggested shapes, out of which the preferred or relevant ones are marked as such by the user. The user is then presented with shapes similar to the selected ones. The process may then be repeated several times until the user is satisfied, often employing machine learning techniques in order to aggregate and refine previous selections. The navigation experience with this approach is not continuous and it often requires the user to go through a tedious task involving many queries by selecting individual preferable shapes. A more intuitive and smooth exploration experience can be achieved by letting the user navigate over an endless two dimensional map of shapes, where similar shapes are displayed closely together. Such a map enables a continuous navigation through the space of shapes, thus eliminating the need for explicit relevance feedback on individual queries. The challenge, however, is the generation of such maps of 3D shapes.

The search space of 3D shapes is of high dimensionality. Generating a cohesive global manifold that preserves similarity relations among all shapes is therefore challenging, if at all possible. However, when a user interactively navigates a map-like interface, only a small portion of the search space is displayed at a time. Our key idea is that for such navigation, global requirements can be relaxed. Navigation is done over a pseudo-map, where the data is dynamically organized into a local manifold, only in the region currently observed by the user. The benefit of generating a dynamic map on the fly is twofold. First, global constraints are relaxed and a locally continuous map can be generated, in which a pair of shapes are near in the embedding only if they are relatively close in the original high dimensional space. Second, the generated map can interactively change according to the

user's interest and direction of browsing, thus providing an effective browsing experience without intrusively querying the user.

Figure 1 illustrates the navigation process in our solution. The user views a local subset of shapes, ordered such that similar shapes are next to each other. In this particular example, trucks with higher or bigger bodies typically appear in the top right and trucks with smaller bodies appear in the bottom left corner. The user decides to focus on trucks with smaller bodies, and thus pans towards the bottom left corner. Another patch of the map is revealed, and instantly filled with models of trucks similar to the models framed by the red rectangle. The currently displayed map can be figuratively viewed as a window that shows a local patch of the pseudo-map. Figure 2 shows a screenshot of our system during a typical browsing session.

The challenge in generating such pseudo-maps is to create local manifolds that keep the sense of continuity. That is, the user pans over the pseudo-map while the manifold is perceived to be continuous. We present a technique of embedding shapes onto dynamic pseudo-manifolds, where the relative positions of shapes respect only local high-dimensional relations. Relative distances among the displayed shapes are not necessarily preserved, allowing for an efficient usage of the display space and a spatially dense representation of the shapes domain. In contrast with common dimensionality reduction techniques (e.g. MDS), the end result of our method is not a global map which contains all shapes at once. The generated local pseudo-maps only exist temporarily within the viewport of the user; when the user navigates to reveal a new region of the map, only local relations to the previous map are maintained. Navigation over the pseudo-map enables a free-form exploration, where users can quickly and seamlessly direct the search towards relevant models of their choice.

Our dynamic map bears some resemblance to the self organizing map (SOM) [Koh90], a popular dimensionality reduction method that produces a dense and intuitive grid-like structure. However, an SOM provides a global solution, in which local discontinuities may occur frequently. In addition, it entails a computationally intensive training process, which is applied globally as a pre-process, making it difficult to use on a very large dataset with frequent updates. Our technique is local and computationally inexpensive, which makes it a viable option for massive online datasets of shapes which are constantly changing.

The generation of local neighborhoods in the dynamic map is based on the assumption that for high dimensional data such as 3D models, short distances are more accurately measured than long distances. We thus use only the shortest distances between shapes in our dataset; only the distances to k nearest neighbors (with k being a small positive integer) of each shape in the dataset are considered. A dense set is expected to have shorter distances, and thus more accurate, than a sparse set, hence our method is especially suitable for massive datasets.

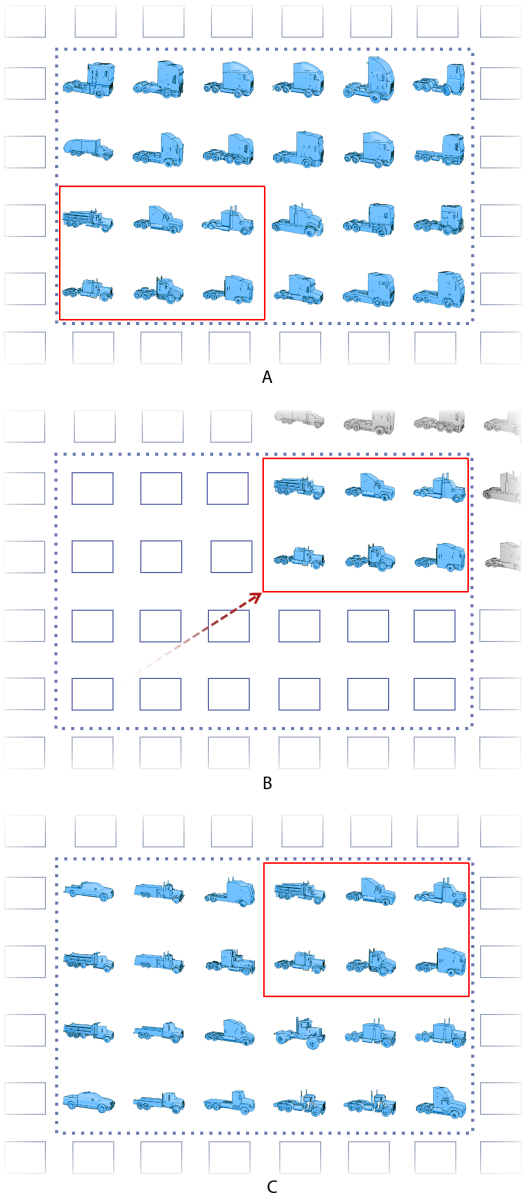

Figure 1: *Browsing shapes using a dynamic map. The map displays a region of shapes ordered by similarity (A). Dragging the map to the up right corner (B) reveals new shapes which are similar to shapes in the dragging direction (C).*

2. Related Work

A common means to explore large shape repositories is by searching for similar shapes through a series of queries. The problem of searching for similar shapes to a given query

object is known as "shape retrieval". During the last two decades a huge body of work in that area has focused on the development of various shape descriptors and signatures to facilitate retrieval. Among them are descriptors based on statistical moments [ETA02, NK03, KFR03], distance [OFCD02], symmetry [KFR04], volume [ZC01, SSCO08]. For more information see a survey by [TV04]. An alternative approach was introduced by Bronstein et al. [BBGO11]. Instead of global shape signatures they compute local features such as Heat Kernel Signature (HKS) [SOG09], quantize them into geometric words, and use them in a bag of words manner to discover similarities between shapes.

Relevance feedback [LMT05, CLT06, ASYS10] helps guide the shape retrieval process according to the user's own individual preference. While this process may be effective at filtering relevant elements out of a massive collection, the use of relevance feedback in commercial search interfaces is still relatively rare [RL03]. One possible explanation is that it requires users to make relevance judgments on each item, which is an effortful user task [RL03,CCTL01]. Our method is inspired by that concept, but operates on the implicit feedback given by the user's advancement through the dynamic map.

Shape exploration is commonly carried out by interactively navigating through design galleries based on a parametric model [SSCO09]. Design galleries have been used for model suggestions based on part correspondence [CK10, KLM*12] and semantic context [TGY*09]. Vieira et al. [VBP*09] utilize design galleries for learning descriptive views of 3D objects, where the user supplies the training data by selecting good and bad object positions. Another form of exploration is presented in Yang et al. [YYPM11], where a shape space is characterized from an input mesh and a set of non-linear constraints is then used for exploration and navigation of new designs that are aligned with the given constraints. Umetani et al. [UIM12] present a method for shape exploration (in this case - furniture) constrained by physical requirements. The user is able to focus on the aesthetic side of the design while the system enforces physical soundness. Ovsjanikov et al. [OLGM11] extract a deformation model from an input shape to explore in a constrained manner the variability within a set of similar shapes.

Planar Mapping. Generating a two dimensional map of high dimensional elements is in essence a dimensionality reduction task. Common dimensionality reduction techniques such as multidimensional scaling (MDS) or locally linear embedding (LLE) [RS00] create a global manifold that aims to preserve the distances among the high dimensional data points, to the extent possible. Such global solutions are beneficial for applications such as clustering and classification, which rely on the underlying geometry or spread of data.

Our premise is, that for browsing tasks, there is no need for an accurate representation of the original distances between shapes. In fact, an even spread of shapes over the map area can be more beneficial than an accurate representation of the original geometry of the search space, especially in cases where the original data includes very distinctive clusters which may appear too far apart for easy navigation. The above mentioned self-organizing map [Koh90] produces a grid which preserves similarity between elements without preserving the distance. Works such as [SKK04] and [LLD12] utilize SOM to visualize a given small set of elements (up to a few hundreds samples) in a global cohesive map. Such methods work well for small sets, however they are too computationally intensive and globally constrained to be effective for massive datasets.

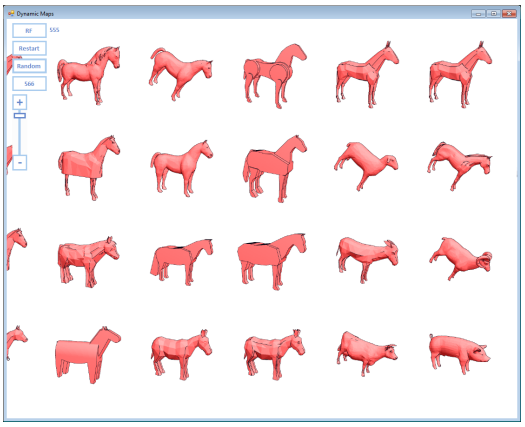

Figure 2: *A screenshot of a typical browsing session.*

3. Map Generation

We provide the user with a dynamic grid-like map which is instantly and continuously generated during user interaction. The input to the map generation process is a precomputed nearest neighbors graph with a similarity score for each edge. The map can be seeded around a specific shape or constrained by any number of shapes. As the user is navigating by panning the map, the map is extended locally to the region of interest, using previously placed shapes as constraints. The map is generated by iteratively filling in empty cells in the grid with the most compatible shape for each cell. The compatibility of a shape to a cell in the grid depends on the shapes that are already assigned to adjacent cells in the grid; each adjacent shape votes for its nearest neighbors as candidates, and the scores of all candidates are accumulated to produce a majority vote.

Every model M in the dataset is associated with a list of nearest neighbors $M' \in Near(M)$, and their respective similarity scores $S(M,M')$. Each cell c in the grid is connected to a weighted list of adjacent cells c_i with respective weights w_i. For example, in our implementation each cell is connected to neighbors on the five by five grid centered at the cell in question, with weights that are inversely correlated

with the Euclidean distance between the cells. We refer to existing models that occupy the adjacent cells of cell c as *reference models* or $R(c)$ where each filled cell c_i is associated with a model M_i. The compatibility score for placing a model M in cell c is defined as the weighted sum of similarity scores for each neighbor that appears in the list of reference models:

$$C(M,c) = \sum_{M_i \in R(c)} w_i \cdot S(M,M_i)$$

where $S(M,M_i) = 0$ when model M_i is not a nearest neighbor of model M. At each iteration, we choose a vacant cell c in the grid, and search for the model that maximizes the compatibility score,

$$M_c = \underset{M}{\operatorname{argmax}}\, C(M,c).$$

To reduce the search space, we only consider models which are nearest neighbors of the reference models. We also exclude shapes that already appear on the map from the candidates list, to avoid repetitions. Since the number of adjacent cells is bound (depending on the grid size that is chosen), the computational cost of creating the map amounts to a small constant, independent of the dataset size, which allows creating the map on-the-fly during user interaction at a rapid fashion.

The voting process gives precedence to cells that are filled early in the map generation process. We use this to further enhance the user experience, by selecting the vacant cells in accordance with the user actions. In general, we give precedence to cells that have the most filled cells which are direct neighbors in the 8-connected grid. However, since the map is a regular grid, often there will be ties and many cells will have the same number of reference models, for example along the edge of the previous region of interest. We break ties using the following process. We compute vectors from the previous center of the map to each cell, and to the new center of the map. We then select the cell with smallest angle between the map's center vector and the cell vector. This causes the grid to start growing from the user's focus area on and outwards into the rest of the map.

Figure 3 illustrates the order in which empty cells in the grid are filled. The user drags the map two cells up and one cell to the right. The center of the user's viewport thus moves on the map in the opposite direction; two cells down and one cell to the left. The cells marked with numbers will be filled first in their respective order, followed by the rest of the cells on the grid. Existing shapes which are closer to the panning direction effectively have more weight in the map generation, since their neighbors are selected first.

The map-filling algorithm is simple and easy to adjust to custom graphs. It can be applied to graphs of any shape, and does not require regularity or planarity. Supporting weighted graphs requires a minute change in the compatibility score.

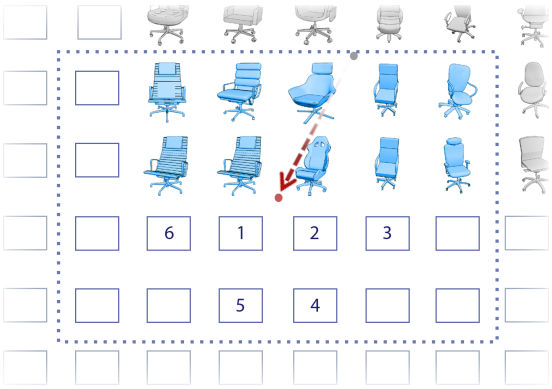

Figure 3: *The order in which cells in the map are filled is relative to the direction of browsing. In this example, the user dragged the map two models up and one model to the right. The gray dot and red dot, respectively, mark the previous and new center of the viewport. The numbers state the order in which the first six cells on the map are filled.*

3.1. Zoom Levels

Our dynamic maps support zooming out to see a larger variety of shapes, and zooming in on a region to see more similar shapes. We support zooming operations by selecting a hierarchy of *high-level delegates* that represent every model in the dataset. All models in the dataset are contained in the first zoom level; every delegate in the second level represents a group of models in the first level, every delegate in the third level represents a group of delegates in the second level, and so on. For each zoom level we connect the delegates in a nearest neighbors graph as described below.

When the user is browsing the map in zoom level l, only models of level $l' \geq l$ are displayed, and the k-NN graph of level l is used. When the user zooms in to level $l-1$, the shapes are spaced out by a given amount, as illustrated in Figure 4. In our implementation the map size doubles, so there is one vacant cell between every two shapes in every direction. Then the map generation process fills the gaps using the k-NN graph of level $l-1$. Precedence is again given to cells that have the most filled neighbor cells, with ties broken arbitrarily. Note that higher level delegates are not excluded from the map when browsing lower levels, and can appear among low level models according to the low level k-NN graph.

When the user zooms out to level $l+1$, a continuous browsing experience is maintained by keeping some of the models that were displayed. As with zooming in, spacing may vary. In our implementation the map size is reduced by half in every direction towards the center, so there is one shape left out of every four around the shape closest to the center of the map. Since the map is now smaller than the region of interest, shapes are filled around it using the map

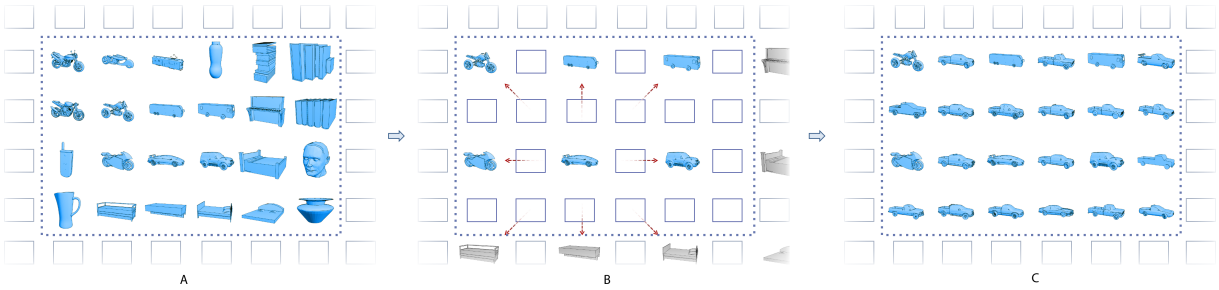

Figure 4: *Zooming in. (A) The initial map. (B) The map after a zoom in operation, with empty spaces between the shapes. (C) The map is filled with new shapes that match their surrounding.*

generation process and the k-NN graph of level $l+1$. Since some of the models are potentially from a lower level, they may be missing from the k-NN graph, in which case they use the k-NN lists of their closest delegates.

Delegate selection can be implemented using various algorithms, as long as every model has at least one delegate in its nearest neighbors list. In our implementation we use a straightforward algorithm, which can be done once for the whole dataset or incrementally when new shapes are added. For each shape in the dataset, we check whether one of its nearest neighbors is already a high-level delegate. If none of the nearest neighbors of the shape is a delegate, the shape itself becomes a delegate for all of its neighbors. The same process can be done when adding a new shape to an existing dataset.

Next, a list of high-level nearest neighbors is created for each high-level delegate M. A high-level neighbor is a delegate M' that has at least one common nearest neighbor with M. The score of the high-level neighbors is the maximum accumulated score of the path in the k-NN graph that connects the two delegates:

$$S(M,M') = \max_{M'' \in Near(M) \cap Near(M')} (S(M,M'') + S_j(M'',M')).$$

If a delegate has more than k high-level neighbors, only the k neighbors with the least scores are kept. This process is repeated recursively on the high-level k-NN graph to create multiple zoom levels. The list of high-level delegates and their k-NN graph is computed as part of the pre-processing, so there is no additional computational cost for browsing when there are multiple zoom levels.

4. Shape Similarity

The map generation is decoupled from the k nearest neighbors computation, which could be replaced by any k-NN dataset. However, the question remains how to effectively measure similarity between shapes in order to define the nearest neighbors of each shape.

Many shape descriptors have been suggested for the task of shape retrieval. A most popular one is the *lightfield* descriptor (LFD) introduced by [CTSO03]. LFD consists of rendering orthographic silhouettes of the model from ten different angles on a dodecahedron. To compare two models, the rendered silhouettes of the models are compared. All possible rotations of the dodecahedron (60 in total) are considered to compensate rigid rotations of the compared models. LFD is one of the most effective descriptors to discriminate between different shape classes [TV04, SMKF04]. Yet, a nearest neighbors query using LFD may still contain irrelevant shapes. An example is shown in Figure 5.

To identify similar objects where LFD fails to do so, we consider two more shape descriptors. D2 descriptor [OFCD02] is a histogram of Euclidean distance between pairs of points on the shape. The pairs are sampled in a way that ensures invariance to triangulation. Last, we compute a histogram of the discrete Gaussian curvature [MDS*02, ALH05], sampled over each vertex in the shape. Each of the descriptors has different strengths and weaknesses, and we aim at combining the results from all descriptors to identify different aspects of similarity between objects.

A few works combine different features to produce a unified similarity score. Atmosukarto et al. [ALH05] sort all the objects in the dataset based on their distance to a query object, to produce a similarity rank for each model in each feature space. A similarity score is then defined according to said rank. The combined score over all feature spaces is a linear combination of the scores per feature space, where the weight for each feature space is determined by the relative ranks between query objects in that feature space (assuming the query contains several relevant objects). Bustos et al. [BKS*04] use a classified set of models as a training set to define a purity function for each query and feature vector. The purity function indicates the number of training samples from the same class that are retrieved from a k nearest neighbors search using that feature vector. Then the feature vectors are combined linearly using the purity function values as weights.

We employ a simple but effective approach to combine

similarity scores from several feature spaces. Any number of different descriptors can be plugged into our system. We observe that for shape retrieval, a model that is very similar in one specific descriptor space is most likely more relevant than a model that is only moderately similar in several descriptor spaces. Furthermore, for each query and descriptor space, the most similar object might still be quite different from the query object if the descriptor is inadequate for that type of model.

We therefore look for models that are exceptionally similar to the query object, relative to all pairs of models in the dataset. To this end, we compute for every descriptor d the mean distance μ_d and standard variation σ_d between all pairs of shapes in the dataset. If the dataset is quite large, these values can be estimated over a few small random subsets of the dataset. For each query model Q and descriptor space d we have a list of neighbors $M \in N_d(Q)$ with corresponding distances $D_d(Q,M)$. To normalize distances in different descriptor spaces we compute a similarity score for each model in each descriptor space, which is a reversed *standard score*:

$$Z_d(Q,M) = -(D_d(Q,M) - \mu_d)/\sigma_d.$$

A high score means the distance between models is below the average distance between any two models by a certain amount of standard deviations. This normalization allows us to compare distances in different descriptor spaces to obtain a unified similarity rank. We define the normalized score of each model M with respect to the query model Q as the maximum of its score over all descriptor spaces:

$$S(Q,M) = \max_d Z_d(Q,M)$$

where $Z_d(Q,M) = 0$ if M is not one of the nearest neighbors of Q in descriptor d. Note that we do not assume we have the full affinity matrix between all models in the system, so we only use the data of the nearest neighbors for these calculations. The nearest neighbors for each query object Q are the k models which have the maximum normalized score value.

Figure 5 shows an example of nearest neighbors search for two models. For each model on the left, the nearest neighbors are displayed on the right, ordered in two rows from left to right by their relevance score. For the bike model, LFD descriptor works well, and indeed, our feature selection method retrieves the same models as LFD alone. For the bird model, only the first model retrieved by LFD is relevant. Note that there are several other bird models in the dataset which are a better match for the query object, as can be seen in Figure 5c. Using our feature selection mechanism, we retrieve four relevant models out of the first eight, where the first nearest neighbor is retrieved using LFD and the rest are retrieved using D2 descriptor.

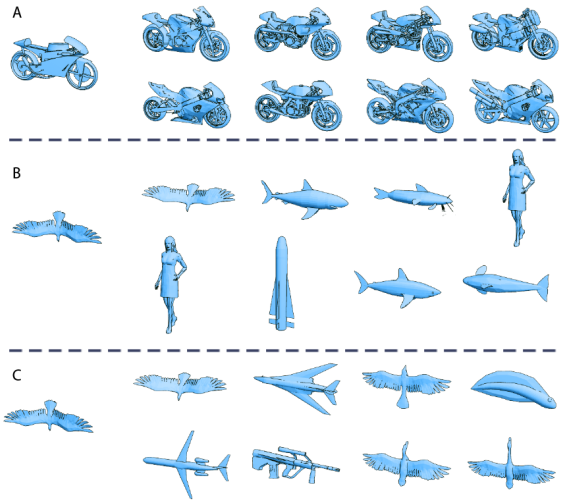

Figure 5: *Examples of k-NN search for two models. (A) A case where LFD works well. (B) Using LFD yields one relevant model out of the first eight. (C) Using a combination of all descriptors yields four relevant models out of the first eight.*

5. Dataset and Implementation Details

In order to demonstrate the soundness and scalability of the solution, a user evaluated system is presented which contains a dataset of 4,573 shapes, collected from two SHREC datasets [VGD*10,LGA*12] and the *Shape COSEG Dataset* [WAvK*12, SvKK*11]. Our implementation is divided into two separate systems. Computing shape descriptors and finding the k nearest neighbors of each shape was done as a pre-process in Matlab, and the user interface and map generation algorithm were implemented using C#.

The user interface displays a grid of shapes, pre-rendered to image files, and enables the user to navigate in the shape space by dragging the mouse cursor over the shapes. The grid pans according to the drag command similar to the way it is done in online maps. As soon as the user releases the mouse button when dragging, the map is populated with new shapes. For zooming, we provide an interface similar to online mapping services, e.g. Google Maps, in which the user sees the current zoom level and can click to zoom-in or zoom-out of the current map. The map can be initialized from a random location or from a manually set location, at the user's discretion. In addition, we provide a double-click feature which allows the user to quickly focus on a single shape. When the user double-clicks on a shape the map is reinitialized around the selected shape. This is useful to give precedence to nearest neighbors of the focused shape over neighbors of neighbors which may have been selected before the focused shape appeared on the map.

New shapes are loaded almost instantly after every navigation action. Internal profiling of the system shows that the map generation algorithm takes between 0.001 and 0.02 seconds for each page, depending on the number of new shapes

that are fetched. The bottleneck of our system is loading the representative image files from disk which takes a portion of a second. This shows that the algorithm is suitable for handling large datasets with ease. Since the number of candidate shapes for each cell in the grid is bounded by a constant, regardless of the number of shapes in the dataset, the time complexity of displaying the map should be the same for very large datasets such as millions of shapes. The space complexity is linear since only k nearest neighbors are kept for each shape, so running the system with a very large dataset does not require extraordinary computational resources.

6. Evaluation

To evaluate our method, we conducted a user study that compared it with a relevance feedback method, implemented using the same dataset and same k-NN structure as the dynamic map. The system initializes using a specific model as a starting point. In the relevance feedback system, the first 20 nearest neighbors of the initial model are displayed on a grid five cells wide and four cells long. The user can then select one, two or three shapes to focus on, and clicks a button to fetch the next set of shapes. The nearest neighbors of all selected models are marked as candidates, and the score of each candidate is accumulated for each selected model which is a neighbor of the candidate. This way candidates that are mutual neighbors of several selected models are preferred. Then the 20 candidates with the maximum accumulated similarity scores are displayed to the user for another round of relevance feedback.

Note that the described system is far more simplistic than a state of the art relevance feedback method; such methods generally employ powerful learning algorithms which help improve the results with every iteration. Nevertheless, it is a representation of the typical user interface of a relevance feedback system, which provides the basic user experience aspects of relevance feedback systems.

Setting. We employed a 2(method) x 3(task) within-subject design to compare performance and subjective opinions of participants. The main variable, *method*, describes the search system used and included either the Dynamic Map method (DM) or the Relevance Feedback method (RF).

The second variable, *task*, describes the tasks that participants were asked to perform. Three different task types were given:

1. Choose a model out of the collection according to subjective preferences (e.g., "find a dining room chair that you would like to have in your home")
2. Find multiple models in a category (e.g., find ten different types of four legged animals such as horse, cow, dog, etc).
3. Given a specific reference image of a model, find that specific model in the collection (e.g., a model of an electric guitar with very distinct body shape was provided. The starting point was a collection of guitars).

For each task type, we devised two similar tasks to be performed. For example, for the third task, either an image of a guitar or an image of a person was given. In addition, for each task, a starting shape was determined. The starting shape was located in the vicinity of the target/s (e.g., for finding a dining room chair, the starting point was a swiveling office chair). For the DM method, the starting shape was used as a basis to create the initial grid. For the RF method, the nearest neighbors of the starting shape were presented as the initial grid.

Sixteen (16) participants took part in the experiment. Participants were mainly students from a local university. Eleven participants were male and five were female with an average age of 29.5 (SD = 5.2). All participants had previous experience with searching images on the Web, and no participants had previous experience with searching 3D models.

Participants were seated in front of a 22" screen with 1600x900 pixel resolution. This allowed for a grid of 5x4 models to be displayed. Participants were then presented with one of the two methods. The user interface features were first explained to the participants, who were then allowed to freely browse around the model space using the interface until they felt comfortable using it. Participants were then given the three tasks one after another and were asked to perform each task as best as possible. When they completed all three tasks, participants were asked to fill in a questionnaire on their subjective opinion of the interface. Participants were then presented with the second interface on which they completed the same procedure (using three different tasks of the same task type). At the end of the experiment, a comparative questionnaire was given. The order of interfaces (which interface was first used) as well as which set of tasks to perform on which interface was counterbalanced.

Results. Figure 6 presents the average amount of search time per task for both methods. A two-way ANOVA was conducted to assess the time differences between the two methods. Results indicate that it took participants significantly less time to search with the DM method than with the RF method, $F(1,15) = 44.1$, $p<0.001$. A post-hoc analysis using the bonferroni adjustment, examining each task separately, showed that there were also significant differences between the two methods in tasks 1 and 3 with task 2 being very close to significance ($p=0.052$). It should be noted that seven participants in the RF condition were unable to complete task 3 compared to only one participant who was unable to complete the task in the DM condition.

Next, we analyzed participants' opinion of the interfaces. Table 1 presents the set of statements presented to participants after interacting with each method (DM and RF) as well as their average responses. Ratings were given on a 7-point Likert scale to indicate how much participants agreed with each statement, ranging from strongly disagree (1) to

Statement	DM	RF	p-value
The search was enjoyable	5.43(1.26)	3.65(1.5)	0.003
The system was effective for the search purposes	4.56(1.67)	3.50(1.71)	0.065
The system limited my possibilities	4.31(1.4)	5.25(1.52)	0.095
During the search, I stumbled across items I didn't think about	5.25(1.52)	5.06(1.94)	0.687
I easily understood how to use the interface	5.56(1.71)	6.12(1.58)	0.331
I easily understood the efficient way to conduct the search	5.12(1.45)	3.68(1.30)	0.005
It was easy to conduct the searches	5.12(1.31)	3.37(1.58)	0.007
During the search I felt frustrated	3.37(1.66)	4.93(1.73)	0.021

Table 1: *Average ratings (and standard deviation) of the two interfaces. Ratings are given on a 7-point Likert scale ranging from strongly disagree (1) to strongly agree (7). P-values of the Wilcoxon signed test, comparing the two interfaces are provided.*

Statement	DM	RF	No opinion
Which system was more effective for task 1?	11	3	2
Which system was more effective for task 2?	11	5	0
Which system was more effective for task 3?	9	4	3
Which system was more effective overall?	11	3	2
When you have a vague idea of the search, which system is better?	12	4	0
When the target of the search is clear which system is better?	8	7	1
Overall, which system do you prefer?	12	2	2

Table 2: *Direct preferences between the two interfaces (N=16).*

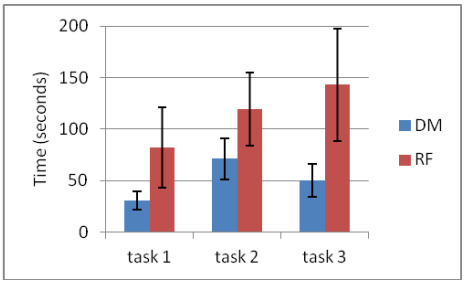

Figure 6: *Average search time for the three tasks in both the DM and the RF methods (N=16). Error bars display 95% confidence interval.*

strongly agree (7). With ranked ordinal data and a relatively small sample size, it is recommended to use a nonparametric statistical test [HCB74]. We therefore used the Wilcoxon signed nonparametric test to examine differences in ranking between the groups. As can be seen in Table 1, results indicate a preference to the DM method on almost all questions.

Finally, we analyzed the direct comparison questions given at the end of the experimental session. These results are presented in Table 2.

Evaluation Summary. Overall, participants clearly preferred the DM method over the RF one. This is demonstrated both in the direct comparison results (Table 2) and in the independent ratings of each interface (Table 1). Participants felt that the DM interface was more enjoyable, effective, efficient and easy to use. Together, these measures are used as an indicator of a system's usability [Bro96], thus our results suggest that the DM method is more usable than the RF method for the given search tasks. Better efficiency is also indicated by the fact that participants completed their tasks faster using the DM method.

7. Discussion and Limitations

We presented a shape exploration technique where objects are laid out on a two-dimensional dynamic map that is locally updated according to user navigation. One of the most prominent features of our approach is the locality of the solution. The local approach enables the construction of an unconstrained, easy to use and scalable system; it can support massive datasets containing millions of models with ease. At the same time, some limitations stem from this locality.

Since we do not keep models outside the current boundaries of the map, models may be repeated during a browsing session, appearing at multiple locations on the map. In practice, it is possible to prevent some repetition of models by excluding models that were recently seen from the search space and remembering previously generated patches on the map. However, this requires a delicate balance, since keeping previously seen regions of the map creates global constraints that often cannot be fully satisfied. Informal feedback from participants in our user study suggests that users do not feel the repetition of models is hindering the user experience, since it is usually easy to avoid by navigating away from seen models, or using the zoom ability to view a greater variety.

On the other hand, some shapes may have very few rele-

vant neighbors in the k-NN graph, and therefore are likely to be excluded from the generated maps. This is due to the voting mechanism which ensures only shapes that are relevant to the surrounding appear on the map, thus pruning outliers.

The presented method is most suitable for free-form search, where the user does not have a specific target in mind, and the goal is to browse a variety of shapes rather than retrieving the single most relevant shape. A primary goal of the dynamic map is to aid the refinement of 3D object search. As such, it is our vision that the technique is used in tandem with keyword shape search. In such a setup, the dynamic map can be seeded around an shape which is the best match for the textual keyword search, to provide the user with a variety of objects that resemble the best match. The map generation method is decoupled from the construction of the k-NN graph, which makes the method applicable for other domains as well, such as searching images, text documents or any kind of high dimensional data.

Acknowledgments. The authors would like to thank Chen Greif for his comments and suggestions, the anonymous reviewers for their feedback, and the participants in our user study.

References

[ALH05] ATMOSUKARTO I., LEOW W. K., HUANG Z.: Feature combination and relevance feedback for 3d model retrieval. In *Multimedia Modelling Conference, 2005. MMM 2005. Proceedings of the 11th International* (2005), IEEE, pp. 334–339. 5

[ASYS10] AKGÜL C. B., SANKUR B., YEMEZ Y., SCHMITT F.: Similarity learning for 3d object retrieval using relevance feedback and risk minimization. *Int. J. Comput. Vision 89* (September 2010), 392–407. 3

[BBGO11] BRONSTEIN A. M., BRONSTEIN M. M., GUIBAS L. J., OVSJANIKOV M.: Shape google: Geometric words and expressions for invariant shape retrieval. *ACM Transactions on Graphics (TOG) 30*, 1 (2011), 1. 1, 3

[BKS*04] BUSTOS B., KEIM D., SAUPE D., SCHRECK T., VRANIC D.: Automatic selection and combination of descriptors for effective 3d similarity search. In *Multimedia Software Engineering, 2004. Proceedings. IEEE Sixth International Symposium on* (2004), IEEE, pp. 514–521. 5

[Bro96] BROOKE J.: Sus-a quick and dirty usability scale. *Usability evaluation in industry 189* (1996), 194. 8

[CCTL01] CROFT W., CRONEN-TOWNSEND S., LAVRENKO V.: Relevance feedback and personalization: A language modeling perspective. In *DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries* (2001). 3

[CK10] CHAUDHURI S., KOLTUN V.: Data-driven suggestions for creativity support in 3d modeling. In *ACM Transactions on Graphics (TOG)* (2010), vol. 29, ACM, p. 183. 3

[CLT06] CAO L., LIU J., TANG X.: 3d object retrieval using 2d line drawing and graph based relevance reedback. In *Proceedings of the 14th annual ACM international conference on Multimedia* (New York, NY, USA, 2006), MULTIMEDIA '06, ACM, pp. 105–108. 3

[CTSO03] CHEN D.-Y., TIAN X.-P., SHEN Y.-T., OUHYOUNG M.: On visual similarity based 3d model retrieval. In *Computer graphics forum* (2003), vol. 22, Wiley Online Library, pp. 223–232. 5

[ETA02] ELAD M., TAL A., AR S.: Content based retrieval of vrml objectsïŧ¿an iterative and interactive approach. In *Multimedia 2001*. Springer, 2002, pp. 107–118. 3

[HCB74] HUCK S. W., CORMIER W. H., BOUNDS W. G.: *Reading statistics and research*. Harper & Row New York, 1974. 8

[KFR03] KAZHDAN M., FUNKHOUSER T., RUSINKIEWICZ S.: Rotation invariant spherical harmonic representation of 3d shape descriptors. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (2003), Eurographics Association, pp. 156–164. 3

[KFR04] KAZHDAN M., FUNKHOUSER T., RUSINKIEWICZ S.: Symmetry descriptors and 3d shape matching. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (2004), ACM, pp. 115–123. 3

[KLM*12] KIM V. G., LI W., MITRA N. J., DIVERDI S., FUNKHOUSER T.: Exploring collections of 3d models using fuzzy correspondences. *ACM Trans. Graph. 31*, 4 (July 2012), 54:1–54:11. 3

[Koh90] KOHONEN T.: The self-organizing map. *Proceedings of the IEEE 78*, 9 (1990), 1464–1480. 2, 3

[LGA*12] LI B., GODIL A., AONO M., BAI X., FURUYA T., LI L., LÓPEZ-SASTRE R., JOHAN H., OHBUCHI R., REDONDO-CABRERA C., ET AL.: Shrec'12 track: Generic 3d shape retrieval. In *Proceedings of the 5th Eurographics conference on 3D Object Retrieval* (2012), Eurographics Association, pp. 119–126. 6

[LLD12] LASRAM A., LEFEBVRE S., DAMEZ C.: Procedural texture preview. In *Computer Graphics Forum* (2012), vol. 31, Wiley Online Library, pp. 413–420. 3

[LMT05] LEIFMAN G., MEIR R., TAL A.: Semantic-oriented 3d shape retrieval using relevance feedback. *The Visual Computer 21*, 8-10 (2005), 865–875. 1, 3

[MDS*02] MEYER M., DESBRUN M., SCHRÖDER P., BARR A. H., ET AL.: Discrete differential-geometry operators for triangulated 2-manifolds. *Visualization and mathematics 3*, 2 (2002), 52–58. 5

[NK03] NOVOTNI M., KLEIN R.: 3d zernike descriptors for content based shape retrieval. In *Proceedings of the eighth ACM symposium on Solid modeling and applications* (2003), ACM, pp. 216–225. 3

[OFCD02] OSADA R., FUNKHOUSER T., CHAZELLE B., DOBKIN D.: Shape distributions. *ACM Transactions on Graphics (TOG) 21*, 4 (2002), 807–832. 1, 3, 5

[OLGM11] OVSJANIKOV M., LI W., GUIBAS L., MITRA N. J.: Exploration of continuous variability in collections of 3d shapes. *ACM Transactions on Graphics (TOG) 30*, 4 (2011), 33. 1, 3

[RL03] RUTHVEN I., LALMAS M.: A survey on the use of relevance feedback for information access systems. *The Knowledge Engineering Review 18*, 02 (2003), 95–145. 3

[RS00] ROWEIS S., SAUL L.: Nonlinear dimensionality reduction by locally linear embedding. *Science 290*, 5500 (2000), 2323–2326. 3

[SKK04] SAKAMOTO Y., KURIYAMA S., KANEKO T.: Motion map: image-based retrieval and segmentation of motion data. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2004), SCA '04, Eurographics Association, pp. 259–266. 3

[SMKF04] SHILANE P., MIN P., KAZHDAN M., FUNKHOUSER T.: The princeton shape benchmark. In *Shape Modeling Applications, 2004. Proceedings* (2004), IEEE, pp. 167–178. 5

[SOG09] SUN J., OVSJANIKOV M., GUIBAS L.: A concise and provably informative multi-scale signature based on heat diffusion. In *Computer Graphics Forum* (2009), vol. 28, Wiley Online Library, pp. 1383–1392. 3

[SSCO08] SHAPIRA L., SHAMIR A., COHEN-OR D.: Consistent mesh partitioning and skeletonisation using the shape diameter function. *The Visual Computer 24*, 4 (2008), 249–259. 3

[SSCO09] SHAPIRA L., SHAMIR A., COHEN-OR D.: Image appearance exploration by model-based navigation. In *Computer Graphics Forum* (2009), vol. 28, Wiley Online Library, pp. 629–638. 3

[SvKK*11] SIDI O., VAN KAICK O., KLEIMAN Y., ZHANG H., COHEN-OR D.: Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. In *ACM Transactions on Graphics (TOG)* (2011), vol. 30, ACM, p. 126. 6

[TGY*09] TALTON J. O., GIBSON D., YANG L., HANRAHAN P., KOLTUN V.: Exploratory modeling with collaborative design spaces. *ACM Transactions on Graphics-TOG 28*, 5 (2009), 167. 1, 3

[TV04] TANGELDER J. W., VELTKAMP R. C.: A survey of content based 3d shape retrieval methods. In *Shape Modeling Applications, 2004. Proceedings* (2004), IEEE, pp. 145–156. 1, 3, 5

[UIM12] UMETANI N., IGARASHI T., MITRA N. J.: Guided exploration of physically valid shapes for furniture design. *ACM Transactions on Graphics 31*, 4 (2012). 3

[VBP*09] VIEIRA T., BORDIGNON A., PEIXOTO A., TAVARES G., LOPES H., VELHO L., LEWINER T.: Learning good views through intelligent galleries. In *Computer Graphics Forum* (2009), vol. 28, Wiley Online Library, pp. 717–726. 3

[VGD*10] VANAMALI T., GODIL A., DUTAGACI H., FURUYA T., LIAN Z., OHBUCHI R.: Shrec'10 track: Generic 3d warehouse. In *Proceedings of the 3rd Eurographics conference on 3D Object Retrieval* (2010), Eurographics Association, pp. 93–100. 6

[WAvK*12] WANG Y., ASAFI S., VAN KAICK O., ZHANG H., COHEN-OR D., CHEN B.: Active co-analysis of a set of shapes. *ACM Transactions on Graphics (TOG) 31*, 6 (2012), 165. 6

[YYPM11] YANG Y.-L., YANG Y.-J., POTTMANN H., MITRA N. J.: Shape space exploration of constrained meshes. *ACM Trans. Graph 30*, 124 (2011), 1–124. 3

[ZC01] ZHANG C., CHEN T.: Efficient feature extraction for 2d/3d objects in mesh representation. In *Image Processing, 2001. Proceedings. 2001 International Conference on* (2001), vol. 3, IEEE, pp. 935–938. 3