Improving Accessibility of Virtual Worlds by Automatic Object Labeling

Ilias Apostolopoulos
Computer Science and Engineering
University of Nevada, Reno
ilapost@cse.unr.edu

Eelke Folmer
Computer Science and Engineering
University of Nevada, Reno
eelke.folmer@gmail.com

George Bebis
Computer Science and Engineering
University of Nevada, Reno
bebis@cse.unr.edu

July 12, 2013

Abstract

User generated virtual worlds, such as Second Life, typically lack accurate metadata for their virtual world objects. This is a significant problem for blind users who rely on textual descriptions in order to access virtual worlds using synthetic speech. In this paper, we consider the problem of automatic object labeling to improve accessibility of virtual worlds for users with disabilities. Taking advantage of the primitive-based representation of virtual world objects in Second Life, we present an approach that leverages histogram-based geometric object representations, machine learning and crowdsourcing to accurately label virtual world objects at a large scale. We report excellent classification results using seven challenging object classes.

1 Introduction

Though virtual worlds, such as Second Life, have somewhat waned in popularity over the past years, they still attract a respectable number of users and are still very profitable [2]. A number of accessible interfaces [8, 1, 3] have been developed that allow users with visual impairments to navigate their avatar and explore Second Life using audio, for example, by extracting textual descriptions that can be read with a screen reader [8]. Such interfaces require textual descriptions of objects to be present in order to create descriptions of the avatar's environment.

Second Life's content is entirely user generated. When users create new objects, they typically leave the object's name to its default value. A study of Second Life content in 2009 revealed that nearly 32% of Second Life's objects are called "object" [16]. This accessibility problem is similar to web images lacking the alt attribute tag. Over the past years, a number of techniques have been developed for labeling virtual world objects in order to make virtual worlds more accessible. This is a challenging problem since the same object can be created in completely different ways (i.e., using different number and type of primitives) by different users. These techniques typically employ some form of crowdsourcing (i.e., humans provide object descriptions).

Although object recognition is a task where average human performance outperforms machine learning, labeling millions of objects is a tedious and time consuming task when done manually. Unlike objects in real images where it may be difficult to segment due to background clutter, extracting virtual world objects does not require explicit segmentation as they can be captured in isolation from the background. Virtual world objects are represented using geometric primitives which enables extracting geometric object representations efficiently and fast. In this paper, we present an approach that uses a small amount of crowdsourcing efforts to train a classifier that enables large-scale, real-time labeling of virtual world objects.

The rest of the paper is organized as follows: Section 2 reviews related work. Section 3 describes the proposed approach. Our experimental evaluation and results are provided in Section 4. Section 5 discusses our results and provides directions of future work.

2 Related Work

Various techniques have been proposed for 3D object classification and retrieval. Many of them analyze 3D objects by considering a monolithic 3D model [13, 14, 15, 5, 10, 17]. These techniques rely on extracting shape features that are invariant to geometric transformations.

In [5], second-order 3D and spherical-kernel moment invariants are extracted from the density function in order to represent objects with a set of rotation-scale-translation invariant features. These features are used to find the closest match over a database of objects.

In [13], the creation of a "shape function" was proposed. First, randomly selected points are sampled on the surface of an object. These points are then used to compute geometric object properties such as the Euclidean distance of two random points, the square root of the area of the triangle formed by three random points, etc. These properties, which are called shape functions, are used to extract shape distributions, and compare them against each other to find out

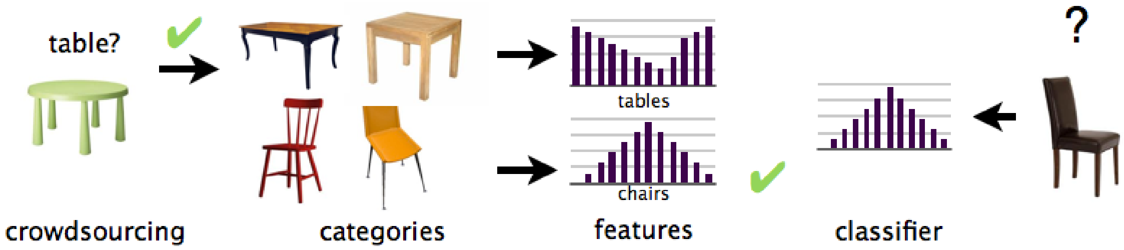

Figure 1: A four step process is used for classifying objects: (1) objects are collected based on their class name; (2) crowdsourcing verifies the correctness of each object class name; (3) useful features are extracted to represent each object and (4) a classifier is trained to classify objects into known object classes.

which one can better represent different classes of objects.

Using machine learning to automatically label 3D objects was proposed in [17] using features such as volume-surface ratio, moment invariants, and Fourier transform coefficients. A different approach considered the topological and skeletal object structure by utilizing multi-resolution Reeb graphs [10].

In [11], 3D objects are divided into smaller parts followed by classification. These parts are grouped into part classes. To classify an object, points are sampled on the query object and are compared to different parts, yielding a probability that this object belongs to a class.

In a related approach, instead of dividing the object into parts, the object is rendered in a volume using voxels [6]. Only the voxels inside the object are taken into consideration. Features are extracted from those voxels and stored in histograms. Support Vector Machines (SVM) are then used for training and classification.

A more recent work utilizes the bag of features technique [7]. Similarly to previous technique, the object is rendered in a volume using voxels. Points are sampled in the object and local patches are extracted based on the points. These patches are then represented as histograms which are used for classification.

3 Approach

We present an automated approach which consists of four main steps (see Fig. 1):

1. Collect objects using a spider.
2. Filter objects to remove noise and duplicates.
3. Extract useful features.
4. Train a classifier to distinguish among different object classes.

Object collection: An automated agent/spider has been developed which can control an avatar in the virtual world, and can teleport automatically from region to region. In each region, the agent collects objects in that region and stores them in an XML file using their unique identifier as their name. By passing keywords as parameters, the agent collects specific objects we are interested in; for example, it can collect all objects that have a specific name as a substring in their name.

Filter objects: The agent collects objects based on their name. For example, passing the keyword "table" results in a collection of objects whose name includes the string "table". This means that the agent may collect objects that are both tables (e.g., "my table") but also not tables (e.g. "table lamp", "table cloth"). Moreover, it can collect mislabelled objects. To verify the correctness of the collected objects, we need to filter them by visually inspecting them.

Because this may involve large amounts of human effort, we use a crowdsourcing marketplace where human workers are paid to verify the correctness of a label given a picture of that object. Crowdsourcing can be done in a large scale at a low cost; a higher level of accuracy may be expected when involving a higher number of individuals for verifying the labels. Our agent can take the XML definition of an object as input and render it in the virtual world. We render objects in a white box to allow for a good contrast between the object and the background. For objects that are white, a different background color is used. The objects are scaled to fit and fill the box so as to make sure the picture contains the whole object. The agent takes a snapshot from each object at a certain distance. Then, the snapshot is saved as an image into a folder that is named after the object's class.

These pictures are used to create human intelligence tasks (HITs) at a crowdsourcing marketplace, where workers are paid to confirm whether the label of an object matches its picture. The results of multiple HITs are compared to determine the accuracy of a given label for an object. Although this reduces the possibility of wrong labels, it does not completely eliminate objects having wrong labels. Therefore, when training a classifier, some training examples might have been labelled incorrectly.

Determine useful features. The previous step results in a set of objects with accurate labels that can be used to train a classifier. In Second Life, 3D objects are formed by 3D primitives or prims (e.g., boxes, spheres etc.) instead of meshes. Various features can be extracted from each primitive. Here, we have experimented with the type of prims, their size, orientation, and eccentricity.

Prim type describes the type of the primitive (e.g., box, sphere, cylinder, etc.). Second Life contains eight main prim types. These prim types are box, sphere, cylinder, prism, torus, tube, ring, and unknown. It should be mentioned that besides the main prim types, there is a special prim type called "sculpted" prim whose shape is determined by an array of x, y, z coordinates stored as RGB values in an image file (i.e. texture or map). Since the properties of each sculpted prim are mainly defined by embedded textures rather than parameters which are common in all other types, the encapsulated information may be misleading for a learning agent. Therefore, objects with sculpted prims were not included in our data set.

Size refers to the volume of the bounding box of a primitive. For scale invariance, we represent it as the ratio of the volume of the primitive to the total volume of the object (i.e., sum of prim volumes). Orientation is defined as the relative angle between the primitive's orientation and the object's average orientation; this is also rotation invariant. We estimate the average orientation of an object by the orientation of its largest dimension. Finally, eccentricity is defined in scale invariant way as the ratio of the smallest to the largest dimension of the primitive.

Besides these features, several other features were explored. Some prims in Second Life, for example, have taper and/or twist. Taper thins out either the top or the bottom side of prim while twist twists either the top or the bottom side of the object. However, after running several experiments using these features, our results indicated that they were not helpful in distinguishing between object classes. In addition, color was taken into consideration but was discarded quickly as in virtual worlds, objects typically have a wide range of colors.

To take advantage of the prim-based representation of objects in Second Life, we have adopted a bag-of-features approach [4] which has demonstrated good success in computer vision. In the bag-of-features approach, objects are represented as order-less collections of local features using histograms. The main idea is quantifying the number of occurrences of local features in an object. Since the number of local features might be very large, a visual vocabulary is typically built by clustering the local features. Features belonging to the same cluster are represented by the center of that cluster which is referred to as "visual" word. Therefore, bag-of-features represent objects as histograms of visual words drawn from a visual vocabulary which is built from local features.

When using type as a feature, each prim type is considered as a visual word. In this case, each object is represented as a histogram which encodes the relative frequency of occurrence of each prim type in an object. When using any of other of the three features (i.e., size, orientation, eccentricity), the visual words are defined by dividing the range of values that each feature can assume into a number of bins and associating the center of each bin with a visual word. In this case, each object is represented as a histogram which encodes the relative frequency of occurrence of various feature values in the object.

It should be mentioned that building the visual vocabulary for each feature is fast and easy. In general, the bag-of-features approach is sensitive to background clutter since it cannot distinguish between objects and background. This is not an issue in Second Life since objects in virtual worlds are defined independently of other objects.

Classifier Training. The histograms described in the previous section are used for training the object classifier. From our experiments, we have found that using each feature separately does not provide enough discrimination. However, more powerful object representations can be built by considering combinations of features. For this reason, we have investigated combinations of features by building joint histograms where each dimension corresponds to a different feature. Joint histograms provide more discriminatory information, however, they are memory and time consuming. Moreover, they tend to be quite sparse which implies that some kind of dimensionality reduction is necessary to eliminate

Table 1: The first line indicates the number of items collected from Second Life. Consecutive lines represent the number of items that were filtered out and the last line shows the final number of remaining items.

	Buildings	Cars	Chairs	Lamps	Plants	Tables	Trees
Items Collected	2880	821	3144	2170	2658	1207	3317
1-prim items	-1639	-374	-14	-666	-1608	-147	-2179
2-prim items	-185	-46	-22	-247	-269	-117	-223
Duplicate items	-284	-104	-58	-548	-520	-47	-593
Items with sculpts	-4	-44	-27	-57	-62	-11	-139
AMT	-329	-165	-2619	-115	-59	-153	-50
Final # of items	439	88	404	537	140	732	133

redundancy and improve classification time and accuracy.

To address the issues above, we have adopted Linear Discriminant Analysis (LDA) [12]. LDA is a powerful dimensionality reduction technique which projects the data into an appropriate space where inter-class variability is maximized while intra-class variability is minimized. The reduced dimensionality data is used to train a Support Vector Machine (SVMs)classifier [9]. SVMs are supervised classifiers that have been shown to be an attractive and more systematic approach to learning linear or nonlinear decision boundaries. Given a set of points and assuming two classes, SVM finds the hyper-plane leaving the largest possible fraction of points of the same class on the same side, while maximizing the distance of either class from the hyper-plane. This is equivalent to performing structural risk minimization. Here, we use a multi-class SVM which is a generalization of the two-class SVM classifier.

4 Experiments

We have validated our technique experimentally using objects extracted from Second Life.

Collect objects: An agent for Second Life was developed in Windows using the LibOpenMetaVerse library. For our experiments, we had the agent collect objects for the following seven object classes: “building”, “car”, “chair”, “lamp”, “plant”, “table”, “tree”. Table 1 shows the number of objects collected for each class. These classes were chosen because a the plethora of examples available for each one of them in Second Life. It should be noted that some object classes are structurally very similar (e.g., see Figure 2), making it challenging to define appropriate features for accurate classification.

Filter objects: Before training, the objects collected are filtered through various stages. First, objects with one or two primitives are removed because they do not contain enough information for classification purposes. Then, we filter objects using Amazon Mechanical Turk (AMT) which is a popular marketplace for HITs. Pictures were created for each object and used to create HITs in AMT. Workers were asked to confirm a given label for 10 images by

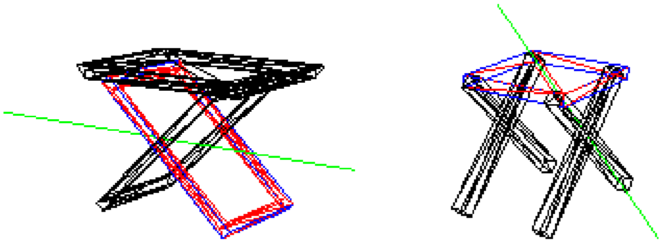

Figure 2: An example of a table and a chair that look very similar.

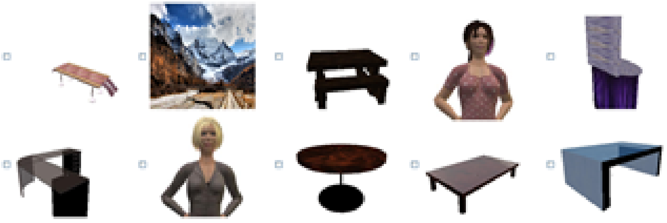

Figure 3: A sample HIT with 7 images of tables and 3 noisy images.

checking a checkbox under each image. Seven to nine out of ten images used in each HIT were taken out of our collected items and the remaining one to three were taken from another set of images of virtual people (avatars) and places. Noise was introduced to be able to detect and filter out the entries submitted by unreliable workers, who may just label all or no objects. Figure 3 shows an example HIT.

The remaining objects are further filtered in order to remove potential duplicates.

Classification:

To investigate the importance of various feature combinations, we have experimented with joint feature histograms where each dimension corresponds to a different feature. The entries of joint histograms are determined by computing the number of occurrences of combinations of feature values in an object.

When considering three or four features together, the corresponding histograms might have very large size which becomes problematic when applying LDA. Since most entries in joint histograms are empty, we only keep track of non empty bins. By taking the union of non-empty bins across all objects in our dataset, we determine which bins are most important (i.e., contain non-zero

Table 2: Examples of joint histogram sizes before and after excluding empty bins.

	number of bins before	number of bins after
2D	4500	2257
3D	90000	5528
4D	720000	7537

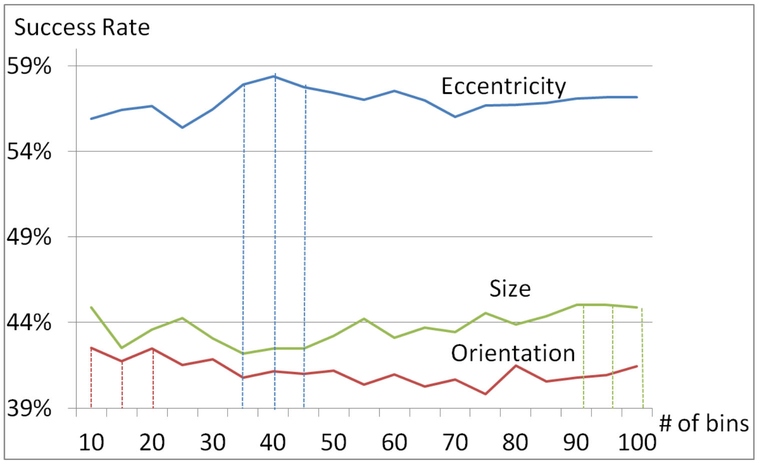

Figure 4: Success rate of different features by varying the number of histogram bins.

entries). These bins are then concatenated into a one dimensional histogram of fixed length which is used to represent objects in our dataset more efficiently. This process reduces the size of histograms significantly as shown in Table 2, without omitting any useful information.

The concatenated 1D histograms are then provided to LDA which determines the most discriminatory features. The results of LDA are used to train the SVM classifier To test the accuracy of the classifier, we used a five-fold cross-validation procedure. In five-fold cross-validation, the data is randomly divided into five mutually exclusive partitions of equal size. Then, one of this partitions is chosen for testing the classifier while the other four are used for training the classifier. This process is repeated five times, each time using a different partition for testing and the other four partitions for training. We report the average five-fold cross validation error.

Parameters: An important parameter when computing the histograms is the number of histogram bins. To see how this parameters affects classification performance, we performed experiments by varying the number of bins from ten to hundred with a step of five (except for type where the number of bins was equal to the number of prim types). Figure 4 shows our results in the case of size, orientation, and eccentricity. As it can be observed, best results were obtained using 90-100 bins for size, 10-20 bins for orientation, and 35-45 bins for eccentricity.

We also considered removing primitives having very small size as they might introduce mostly noise. Specifically, prims that were smaller than a percentage of the whole object were removed. Various levels of thresholding were considered starting from 0% and going as high as 10% in increments of 1%. For each feature, we experimented with all three optimum number of bins found in the previous experiment. Our experimental results, shown in Figure 5, indicate that thresholding does not improve classification accuracy.

Results: In this section, we report our results using 1D histograms (i.e., us-

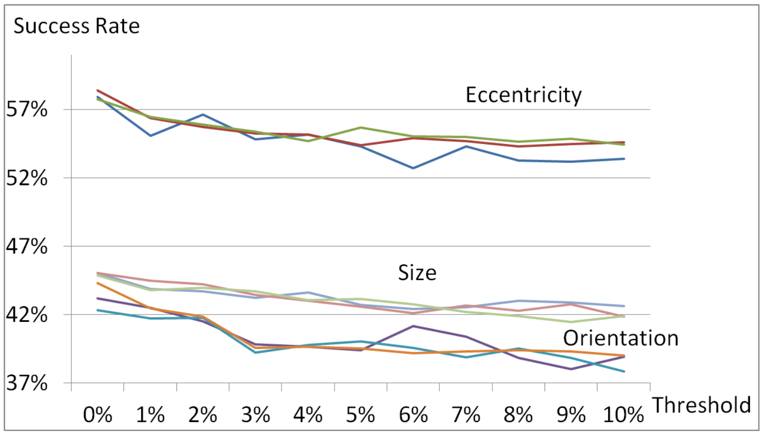

Figure 5: Graphs of various features' success rate based on the amount of threshold.

ing each feature separately), 2D histograms (i.e., using pairs of features) and 3D histograms (i.e., using triplets of features). For each feature, we have considered the optimum number of bins found in the previous subsection.

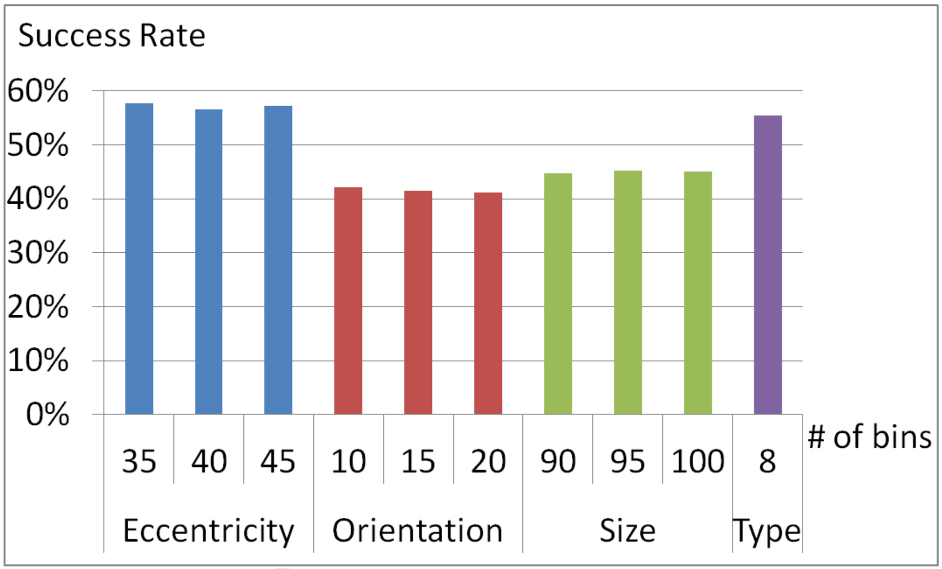

Figure 6: Results for 1D histograms.

Fig. 6 shows our results using 1D histograms; as expected, performance is rather poor indicating that more information is need for discriminating among the object classes. Best results (i.e., 57.67%) were obtained using eccentricity.

Next, we investigated 2D histograms; Tables 3,4,5,6, show different combinations with performance being for most combinations between 55% to 65%. While these results are better than the results obtained using 1D histograms,

Table 3: Results for 2D histograms Type-Size, Type-Orientation, and Type-Eccentricity.

Type-Size			Type-Orientation			Type-Eccentricity		
90	95	100	10	15	20	35	40	45
64.15%	65.01%	64.89%	54.77%	54.65%	54.40%	61.37%	64.73%	64.48%

Table 4: Results for 2D histogram Size-Orientation.

	10	15	20
90	56.21%	62.27%	65.92%
95	57.27%	63.05%	67.39%
100	58.38%	64.07%	68.05%

Table 5: Results for 2D histogram Size-Eccentricity.

	35	40	45
90	94.35%	95.58%	97.05%
95	95.62%	96.27%	98.07%
100	96.31%	97.17%	97.95%

Table 6: Results for 2D histogram Orientation-Eccentricity.

	35	40	45
10	59.12%	62.02%	62.97%
15	62.39%	63.50%	65.87%
20	66.00%	68.13%	69.32%

they are still not satisfactory. However, the combination of size with eccentricity yields a much higher performance between 94% and 98%. This implies that combining size and eccentricity provides high discriminatory information for the 7 object classes.

Our results using 3D histograms are shown in Tables 7,8,9 ,10. As it can be observed, the combination of size, orientation, and eccentricity yields the best results, reaching a classification accuracy as high as 99.96%. These results are consistent with the results obtained using 2D histograms; the addition of an extra feature to the size-eccentricity combination has improved performance.

While it is possible to consider 4D histogram (i.e., combine all four features), our results using 3D histograms are already very satisfactory. Therefore, the extra computational time needed to compute 4D histograms would not be justified by a possible small increase in performance.

Once training has been completed, a classifier using size-orientation-eccentricity takes on average 2.4 seconds to classify a 3D object.

5 Conclusion

We have presented a new approach for automatically classifying virtual 3D objects in Second Life. Our experimental results show that it is possible to obtain very high classification accuracy using 3D histograms based on size, orientation,

Table 7: Results for 3D histogram Type-Size-Orientation.

	10	15	20
90	83.24%	89.27%	90.17%
95	83.04%	88.28%	90.76%
100	84.76%	89.88%	91.48%

Table 8: Results for 3D histogram Type-Size-Eccentricity.

	35	40	45
90	98.77%	99.34%	99.55%
95	99.22%	99.39%	99.63%
100	99.30%	99.59%	99.63%

Table 9: Results for 3D histogram Type-Orientation-Eccentricity.

	35	40	45
10	78.86%	80.87%	82.18%
15	83.78%	85.21%	86.36%
20	85.91%	87.75%	88.73%

Table 10: Results for 3D histogram Size-Orientation-Eccentricity.

	10-35	10-40	10-45	15-35	15-40	15-45	20-35	20-40	20-45
90	99.75%	99.88%	99.92%	99.84%	99.88%	99.92%	99.84%	99.88%	99.92%
95	99.71%	99.88%	99.92%	99.88%	99.88%	99.92%	99.88%	99.88%	99.92%
100	99.88%	99.92%	99.96%	99.88%	99.92%	99.96%	99.92%	99.92%	99.96%

and eccentricity. Our technique can be integrated in existing accessible virtual world clients [8, 1, 3]. When a user encounters an object lacking a name, our technique may allow for recognizing it in real time.

It is worth mentioning that our training set might contain wrong labels and that there is a tradeoff between cost and accuracy when filtering objects using crowdsourcing. More accurate object labels can be obtained by choosing workers with higher acceptance rates, injecting more noise, having more strict agreement rules, decreasing the number of images in a HIT and increasing payment. However, each of these strategies will increase the associated costs. Especially when having to filter large numbers of objects, cost may become a serious consideration. Future work will research whether it is still possible to reach acceptable recognition rates using fewer examples.

For future work, we plan to build a hierarchical classification system in order to assign objects to more abstract categories (e.g., furniture, vehicles). It would be interesting to use crowdsourcing again in order to have workers verify whether objects with a given label (*cat*) belong to a certain category (*animal*). This can help us to establish rules for a taxonomy *animal*←*cat* that the classifier could use. This approach is similar to Yuan [16], with the difference that it could be performed at a much larger scale by taking advantage of crowdsourcing marketplaces, such as AMT.

Acknowledgements

This work is supported by NSF Grant IIS-0917362

References

[1] Ibm human ability and accessibility center, virtual worlds user interface for the blind, http://www-03.ibm.com/able/accessibility_research_projects/virtual_worlds_accessible_UI.html, access date 4-5-2009.

[2] Linden lab's second life 'extremely profitable,' company looking to expand, http://massively.joystiq.com/2012/03/15/linden-labs-second-life-extremely-profitable-company-looking/, url access date 8-18-2012.

[3] Virtual guide dog project, http://virtualguidedog.org, access date: 3-4-2009.

[4] Gabriela Csurka, Chris Dance, Jutta Willamowski, Lixin Fan, and Cedric Bray. Visual categorization with bags of keypoints. In *ECCV International Workshop on Statistical Learning in Computer Vision*, 2004.

[5] George Cybenko, Aditya Bhasin, and Kurt D Cohen. Pattern recognition of 3 d cad objects: Towards an electronic yellow pages of mechanical parts. *INT J SMART ENG SYST DESIGN*, 1(1):1–13, 1997.

[6] J Fehr and H Burkhardt. Harmonic shape histograms for 3d shape classification and retrieval. In *IAPR conference on machine vision applications*, pages 384–87, 2007.

[7] Janis Fehr, Alexander Streicher, and Hans Burkhardt. A bag of features approach for 3d shape retrieval. In *Advances in Visual Computing*, pages 34–43. Springer, 2009.

[8] Eelke Folmer, Bei Yuan, Dave Carr, and Manjari Sapre. Textsl: a command-based virtual world interface for the visually impaired. In *Proceedings of the 11th international ACM SIGACCESS conference on Computers and accessibility (ASSETS'09)*, pages 59–66, Pittsburgh, Pennsylvania, USA, 2009.

[9] M.A. Hearst, S.T. Dumais, E. Osman, J. Platt, and B. Scholkopf. Support vector machines. *Intelligent Systems and their Applications, IEEE*, 13(4):18–28, 1998.

[10] Masaki Hilaga, Yoshihisa Shinagawa, Taku Kohmura, and Tosiyasu L Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 203–212. ACM, 2001.

[11] Daniel Huber, Anuj Kapuria, Raghavendra Donamukkala, and Martial Hebert. Parts-based 3d object classification. In *Proceedings of the 2004 IEEE computer society conference on Computer vision and pattern recognition*, CVPR'04, pages 82–89, Washington, DC, USA, 2004. IEEE Computer Society.

[12] Sebastian Mika, Gunnar Ratsch, Jason Weston, Bernhard Scholkopf, and KR Mullers. Fisher discriminant analysis with kernels. In *Neural Networks for Signal Processing IX, 1999. Proceedings of the 1999 IEEE Signal Processing Society Workshop.*, pages 41–48. IEEE, 1999.

[13] B. Chazelle R. Osada, T. Funkhouser and D. Dobkin. Matching 3d models with shape distributions. In *Shape Modeling and Applications*, pages 154–166, 2001.

[14] Philip Shilane and Thomas Funkhouser. Selecting distinctive 3d shape descriptors for similarity retrieval. In *Shape Modeling and Applications, 2006. SMI 2006. IEEE International Conference on*, pages 18–18. IEEE, 2006.

[15] Dejan V Vranić. 3d model retrieval. *University of Leipzig, Germany, PhD thesis*, 2004.

[16] Bei Yuan, Manjari Sapre, and Eelke Folmer. Seek-n-tag: a game for labeling and classifying virtual world objects. In *Proceedings of Graphics Interface 2010 (GI'10)*, pages 201–208, Ottawa, Ontario, Canada, 2010.

[17] Cha Zhang and Tsuhan Chen. Indexing and retrieval of 3d models aided by active learning. In *Proceedings of the ninth ACM international conference on Multimedia*, pages 615–616. ACM, 2001.