Shape Distributions on Voxel Surfaces for 3D Object Classification From Depth Images

Walter Wohlkinger, Markus Vincze

Vision4Robotics Group, Automation and Control Institute
Vienna University of Technology, Austria

***Abstract*—In this work we address the problem of 3D shape based object class recognition directly from point cloud data obtained from RGB-D cameras like the Kinect sensor from Microsoft. A novel shape descriptor is presented, capable of classifying 'never before seen objects' at their first occurrence in a single view in a fast and robust manner. The classification task is stated as a matching problem, finding the most similar 3D model and view from a database of CAD models gathered from the web to a given depth image. We further show how locally sensitive hashing can be easily adapted to implement fast matching against a database of 2500 CAD models with more than 200000 views in 160 categories. This shape descriptor utilizes distributions on voxel surfaces and can be used in various applications: As a pure 3D descriptor for 3D model retrieval, as a 2.5D descriptor for finding 3D models to partial views or as our main indention as a classification system in the home-robotics domain to enable recognition and manipulation of everyday objects. Experimental evaluation against the baseline descriptors on a dataset of real-world objects in table scene contexts and on a 3D database shows significant improvements.**

I. Introduction

With the arrival of affordable consumer RGB-D sensors like the Microsoft Kinect which deliver dense depth data at 30fps, new challenges and opportunities awaits the computer vision, computer graphics and robotics communities to adapt to this sensor and use its possibilities in various applications. Shape based object class recognition could get a significant boost if it can be adapted to this sensor and if it can be scaled up to hundreds of classes. The training of state-of-the-art recognition and classification algorithms takes a lot of time and effort and can not be easily scaled. With the availability of web-based 3D model collections like Google Warehouse[1] the fusing of these two worlds leads to easy-to-train recognition algorithms.

This work presents an adaptation from a formerly pure 3D descriptor and its extensions to the depth image domain. The presented adaptation of the 3D shape descriptor to 2.5D data enables us to calculate the feature in real time directly from the 3D points of the sensor, without any calculation of normals or generating a mesh from it which is typical of state-of-the-art methods. Furthermore, we show how such a descriptor can be used in a framework which uses a semi-automatic, user-centric approach to utilize the Internet for acquiring the required training data in the form of 3D models which significantly reduces the time for learning new categories. This is especially important in the service and home robotics field and enables learning while the robot is operating.

[1]http://sketchup.google.com/3dwarehouse/

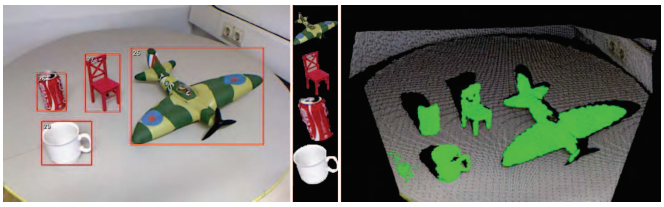


Fig. 1. A scene captured with the RGB-D sensor. The segmented objects are shown as rgb-images in the middle and as segmented point clouds on the right. These segmented point cloud clusters are the input to our 3D object class recognition algorithm.

The domestic setting with its plethora of categories and their huge intraclass variety demands a great deal of generalization skill from a service robot. These categories are characterized by their shape ranging from low intraclass diversification as in the case of fruits and simple objects like bottles up to high intraclass variety such as for liquid containers, tools, furniture and especially toys.

Our contribution consists of a 2-fold strategy for tackling the problems of learning new categories in a fast and semi-automatic manner and of reliable classification of objects from a single view.

First, the robot is provided with access to 3D model repositories on the Internet to use the information found there to cope with the intraclass variation in classification. The problem of large intraclass variability is implicitly solved as the number of available models per object class on the web matches the intraclass variability. Scalability is provided by using locality sensitive hashing on the database to find the best matching view and class in near constant time.

Second, a single-view shape model based approach is used for depth image to 3D model matching to give the system its required speed. The methodology works directly on the 3D data without a need for time-consuming and sensor noise dependent operations such as normals calculation and mesh-generation from the point clouds. The key novelty of the extended shape distribution descriptor is the classification of the point distances as on/off surface by tracing the lines in a voxel grid which significantly boosts this descriptor with respect to discriminability and by this enables to scale to a higher number of categories.

978-1-4577-0242-6/11/$26.00 ©2011 IEEE

The object class recognition system presented uses the extended shape distribution descriptor together with the database of 3D models. It classifies objects independent of scale. Thus scaling differences are handled well, e.g., real objects and toys (dining chair vs. puppet chair) as well as interclass scaling (espresso cup vs. big coffee mug). A depiction of the presented method is given in Figure1 with segmented and classified objects in a table-top scene.

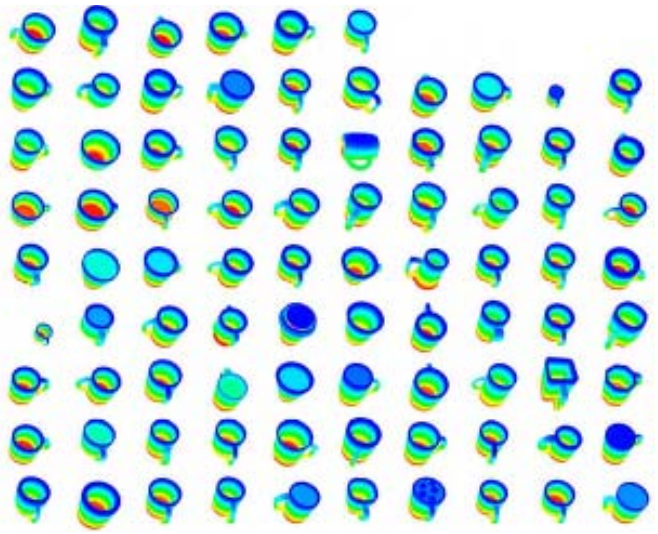
II. Related Work

Automatically accessing an Internet database with 3D models for robotics applications was presented by [1]. They do morphing to increase the number of models and use 2D contours for image based object recognition. Two 3D based approaches with Spin Images for a 3D to range image matching on 3D LIDAR point clouds were presented by [2] and by [3] who also uses 3D models from the web to match against. Range image to 3D model matching for robotic grasping was done by [4] using a dense SIFT [5] based descriptor first introduced by [6] which is the top performer in the SHREC shape retrieval contest of range images[2] [7]. The authors of[4] achieve promising results and improved the matching by extending the visible area of the objects with a movable sensor head. Object categorization from multiple views using a humanoid robot was demonstrated in [8]. Spin Images [9], D2 Shape Distribution [10] and geometric properties like bounding box and volume of the real-world sized 3D model were used for categorization, thus requiring acquisition of a specialized database for training with a structured light sensor. A global descriptor based on histograms of normals was introduced by [11] which delivers excellent results on range scans for container-like objects, but requires calculation of normals.

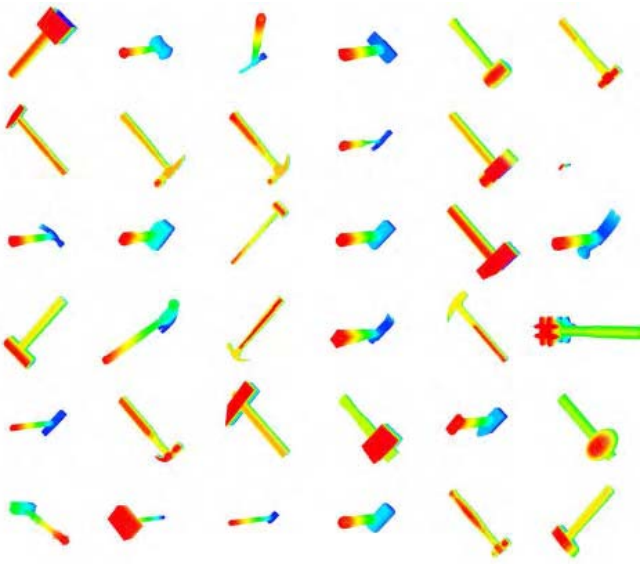
For a general purpose 3D shape based categorization application utilizing RGB-D sensors, acquiring multiple views of the object or restricting the target object to have a previously known scale needlessly restrains the range of applications. Therefore a global descriptor – which describes the whole segmented object extracted from the point cloud – is needed which can be calculated directly on the point cloud, is fast and easy to calculate and still has potential to be adapted to specific needs.

Shape Distributions were introduced as a 3D descriptor by [10] who found that the distribution of distances between two randomly chosen points on the surface of 3D models – tagged as 'D2' – results in a signature which can be used for 3D model matching. Although D2 shape distributions give a good overall shape description, using only the distance as descriptor is not descriptive enough to differentiate between models and classes once the number increases. To increase the descriptiveness in the context of mechanical parts [12] classified each line created by two randomly sampled points on the surface to be either inside the 3D model, outside or both (mixed). This extension is clearly only applicable to full 3D models, but can be adapted if the line classification is reformulated. We build upon the work by [12] and use D2 shape distributions with the extension of classifying the lines created by two random points as on/off the surface of the object.

[2]http://www.itl.nist.gov/iad/vug/sharp/contest/2010/RangeScans/



(a) 3D CAD models of mugs available in our system.



(b) 3D CAD models of hammers.

Fig. 2. Models acquired from Google's 3D Warehouse have no common scale nor do they have a common coordinate system, although most of them share a common main orientation. The large number of 3D models per class allows for coping with the intraclass variety of classes like 'mug' and 'hammer'.

III. Adaptation to Depth Images & Model Preparation

The input into our model acquisition system is the name of the object class. With this keyword, we query the lexical database WordNet[13] to disambiguate the keyword by presenting the different meanings to the user to select the appropriate one. Once the correct meaning of the keyword is

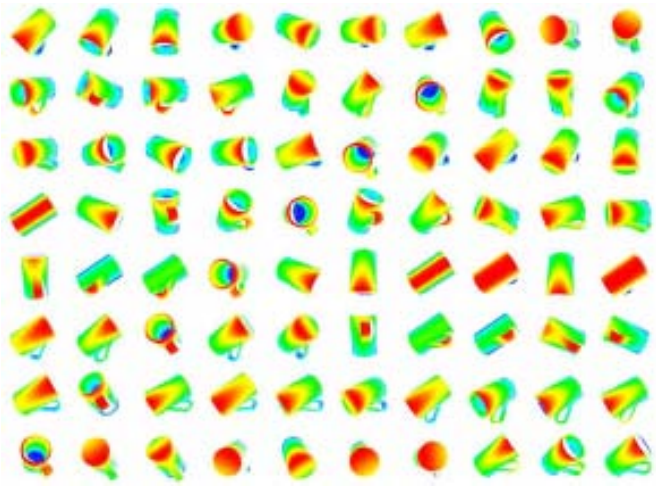


Fig. 3. Synthetically generated point clouds for all the 80 views around a mug model by rendering and sampling the z-buffer.

known, the synonyms and hyponyms (words sharing a 'type-of' relationship with the keyword) provided by WordNet and the translations to the most common languages are used for the 3D model search on 3D Warehouse to get as many models as possible, see Figure2. Having a semantic meaning and an index in the hierarchy provided by WordNet enables further semantic meaningful manipulation applications like pouring something into a container-like object.

One way of matching depth images to full 3D models is to equate the problem of finding the appropriate view of the 3D model. This can be achieved by formulating the problem as a *partial-view to partial-view* matching problem. To use the models from the web for depth image to depth image matching, synthetic depth images are generated by rendering the 3D models and sampling the z-buffer from 80 equally spaced views around the model as shown in Figure3. A high view count of 80 views enables matching of structurally complex objects and further gives an accurate pose estimate for the model. For each of these model views, the 3D descriptor is calculated and stored in the database. The most similar model and view can now be found by comparing the descriptors calculated from the depth image delivered by the sensor to all descriptors in the database. As our approach is orientation invariant and normalized for scale, we can detect objects in any pose and of any size, making no distinction between toy-chairs and real-sized chairs for example. If scale and pose are needed, e.g. for a robotic manipulation task, a subsequent alignment step estimates the roll (about the camera axis) and scale.

IV. D2 Shape Distribution

To find the similarity between two objects, in our case two depth images of views of objects, descriptors are calculated from the data and compared against each other. We use a multi-resolution version of the D2 shape distribution descriptor of [10] who introduced this descriptor for full 3D model matching. The advantage of this descriptor is that the histogram of distances between randomly sampled points can be calculated directly from the point cloud and it is invariant to translation, rotation and with proper normalization also invariant to scale. Figure 4 depicts the creation of the D2 shape distribution histogram. To capture coarse structures and fine details, the best bin-size of the distance histogram has to be chosen. This is avoided in our descriptor by combining multiple bin resolutions into one histogram.

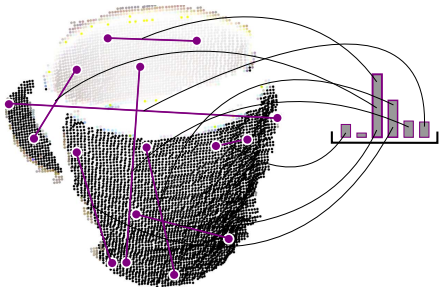


Fig. 4. Distances between randomly sampled points from the partial object view are put – with proper normalization – into a histogram which forms the D2 Shape Distribution descriptor presented in [10].

A. Extending Shape Distributions onto Voxel Surfaces

D2 shape distribution can differentiate between rough shapes but once the number of classes increases or in our case only one view of the object is visible, the descriptive power of this simple feature is not sufficient any more. In the context of 3D mechanical parts [12] extended the D2 shape distributions by classifying each line created by two randomly sampled points to be either inside the 3D model, outside or a mixture of both. We take on this idea and classify lines to be on/off the surface of the point cloud. This is accomplished by tracing the line with the help of the 3D Bresenham algorithm in a voxel grid with side length of 64 as shown in Figure 5. This coarse voxel grid serves as an approximation of the real surface but can be created in a computationally efficient manner. After the tracing, the classified lines are put into three distinct histograms representing ON, OFF and MIXED distances as depicted in Figure 6.

The final descriptor is put together by a coarse histogram with 32 bins and a fine histogram with 128 bins of the D2 shape distribution. This is extended by three 64 bin histograms representing the classified distances on the voxel surface and finally one 64 bin histogram of line ratios of the mixed distances which gives additional descriptive power. To decrease the influence of this last histogram within the whole descriptor, a weight of 0.5 is applied, which was empirically found to give best results.

Matching of these histograms is done with L1-distance which proved to be able to handle the sensor noise without sacrifying classification performance. The distinct varying shapes of the descriptor histograms for different models and poses can be seen in Figure 7 and Figure 8.

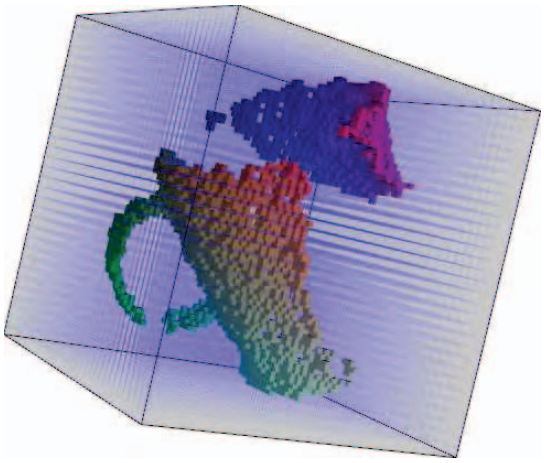


Fig. 5. A 64x64x64 voxel grid is created from the point cloud and used to trace the line distances within.

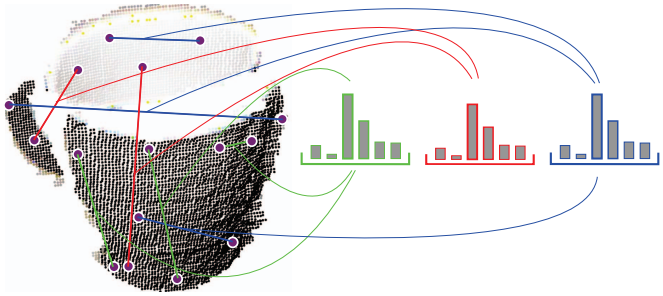


Fig. 6. The classified distances. Green represents the connecting lines lying ON the surface of the object, red represents the lines where only the endpoints are on the surface and the connecting line is OFF surface and the clue coloured line distances are classified as MIXED as they are party ON and OFF the surface. These distances are put into three distinct histograms which give an increased descriptor performance as more information is captured as with one distance histogram alone.

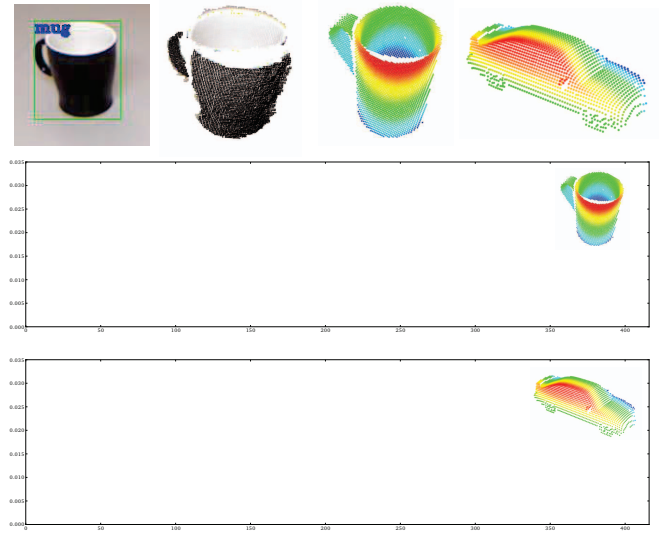


Fig. 7. The query object – the black mug, depicted as image on the left and as point cloud next to it – is compared to two sample database objects, a mug and a car. The green histogram represents the query object, the overlayed red curves represent the two database objects. The similarity of the query object to the mug is clearly visible as their histograms overlap to a high percentage. The car-view generates a completely different shape distribution histogram which results in a low similarity score.

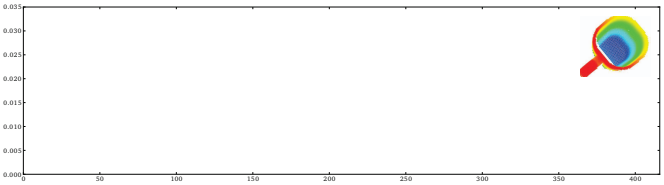


Fig. 8. The query object is compared to the same mug as in Figure 8 but with another pose. The red curve resembles the green histogram only in a coarse manner as the classes are similar but the pose is different.

B. Locality Sensitive Hashing

Matching against a large database - comparing the query descriptor to each descriptor in the database - becomes the bottleneck of the system once the number of descriptors goes beyond several thousands. A naive matching implementation to find the best match takes $O(n)$ which results in approximately 20s for matching against 120000 descriptors in our Python implementation, which is unacceptable given 40 ms for the descriptor calculation. Hashing offers a solution for this scaling problem by arranging the descriptors in a dictionary by their keys (hash) which is calculated from the data and is defined to be unique. Access to the data is given in $O(1)$ with some fixed but neglect-able overhead.

The basic idea of locality sensitive hashing is to relax the uniqueness constraint and calculate the hash in such a way that similar descriptors fall into a common bucket and similar hash keys have similar descriptors in their buckets. Using the hamming distance to calculate the similarity of the hash keys (bit-strings), the query descriptor is compared to all descriptors in the buckets within a hamming-ball of size k, where k is the number of differing bits, i.e. hamming distance. This approach results in a comparison complexity of $O(m * b)$ with m the number of descriptors in a bucket, b the number of buckets within a hamming-ball of size k and $m * b << n$.

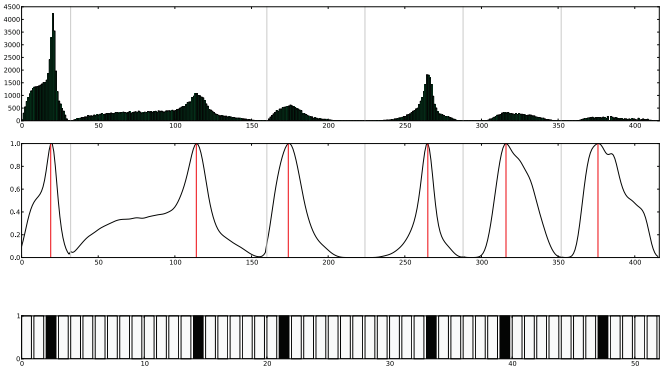


Fig. 9. Generation of the 52-bit hash key from the shape distribution descriptor histogram by setting the bits at the peak positions of every sub-histogram to ones.

Our approach to generate the hash keys is fast, simple and accurate and consists of a thresholding operation on the

descriptor histogram. Figure 9 depicts the calculation of the hash key from the histogram by finding the maxima of the sub-histograms and setting the respective bits at the positions of the maxima to 1. The length of the hash key was empirically found and set to 52 which results in a compression of 8 bins of the histogram into 1 bit in the hash key. Despite the simple hash key generation, the resulting clustering of similar views into buckets is most promising and can visually be seen in Figure 10.

For retrieval of the nearest neighbour view to a given query view, the hash-key is calculated from the query descriptor and all descriptors in the buckets within a hamming-ball of size k are compared to the query descriptor. To speed up the hamming-ball search, each bucket can store links to buckets within hamming distance of k. With a parameter setting of $k = 4$ we get a retrieval result in less than one second in our Python implementation.

V. Experimental Evaluation

The database consists of 163 categories with 2460 3D models and 196800 views which are organized in a hash table with 3604 buckets, averaging a 54 views per bucket. Segmentation of the object from the point cloud takes 0.3 sec on average. The calculation time of the descriptor varies from 40 ms for 6000 points to 25 ms for 1300 points. The evaluation was done on a laptop with a Core2Duo processor at 2.50 GHz. Table I shows some of the categories with their root node present in the database. The evaluation is done on the full database with all 163 classes.

Table II shows the classification result per object class, the number of object views, the classification rate and the class which causes most confusion. By analysing these confusion classes, further improvement on the descriptor level can be made. We will discuss some of the findings here.

- Adding color and size as additional features can clearly increase the classification rate, as some categories are similar in shape but are mainly distinguishable by color (lime, lemon, apple, peach) or by size (golf ball, tennis ball, soccer ball).
- Another point for improvement is the segmentation, as object self-shadowing, small parts and shiny surfaces cause missing points in the point cloud and therefore artificially altering the shape of the point cloud cluster. A combined depth and color segmentation could help to get more complete objects and by that increase the categorization rate further.
- Segmentation of flat or low objects is problematic as for flat objects like bowls and pans the lower part is cut away and objects with a low shape as rulers and pens are often not segmented from the table plane.
- The pose of the object can also give further cues for resolving confusion. One example is the class 'pliers' against 'stapler' where a stapler is roughly a 90 degrees rotated (non lying) pliers.
- Number of 3D models is not covering the whole spectrum

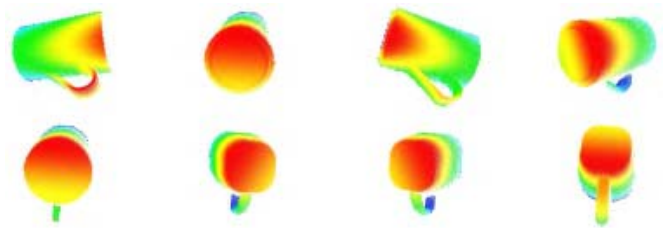


Fig. 10. Visually similar models – which share the same hash key – are clustered into the same bucket. Not all views are exactly the same view in the bucket, as only the peaks of the descriptor histograms are used for the hash-key generation, but the views resemble a rough common view and class.

TABLE I
A subset of the categories present in the database, arranged according to the WordNet hierarchy.

coarse categories	shape-categories
animal	fish, seahorse, camel, cow
	dinosaur, dog, elephant, giraffe
	horse, shark, sheep, turtle, zebra
musical instrument	banjo, cello, drums, flute
	guitar, piano, saxophone, trombone
	trumpet, violing
tableware	fork, plate, spoon
vessel	boat, sailing boat, ship, spaceship, u-boat
kitchen appliance	espresso maker, toaster
container	bottle, box, can, cask, champagne glass
	handbag, jar, luggage, mug, pitcher
	tetra pack, trash can, wine glass
edible fruit	apple, avocado, banana, cherry
	lemon, lime, peach, pear
	pineapple, plum, strawberry, watermelon
vehicle	car, convertible, locomotive, motorcycle
	pickup, race car, suv, tank, truck, wheelbarrow
vegetable	carrot, cucumber, eggplant, mushroom
	onion, pumpkin, radish, squash
	sweet corn, turnip, zucchini
food	donut, egg, peanut
tool	axe, boltcutter, drill, knife
	lawn mower, scissors, ulu
aircraft	airplane, biplane, fighter jet, helicopter
device	calculator, comp, eyeglasses, fire extinguisher
	flashlight, keyboard, light bulb, mouse, notebook
	overhead projector, paper punch, remote, stapler, watch
cooking utensil	kettle, ladle, pan, pot, spatula
furniture	armchair, chair, couch, office chair, table lamp
covering	boot, cap, hat, sandals
	shoe, heels, ski boot, umbrella
weapon	pistol, rifle, sword
hand tool	bottle opener, hammer, pliers, screwdriver
	shovel, wrench

TABLE II
Categorization results per object class

category	# of test sets	classification rate	most similar class (error)
banana	7	71%	bottle
mug	120	94%	trashcan
hammer	38	74%	spoon
plier	7	29%	stapler
airplane	46	45%	monitor
fighter jet	12	67%	paddle
car	15	56%	shoe,motorcycle
suv	8	25%	shoe,pickup
screwdriver	8	25%	pen

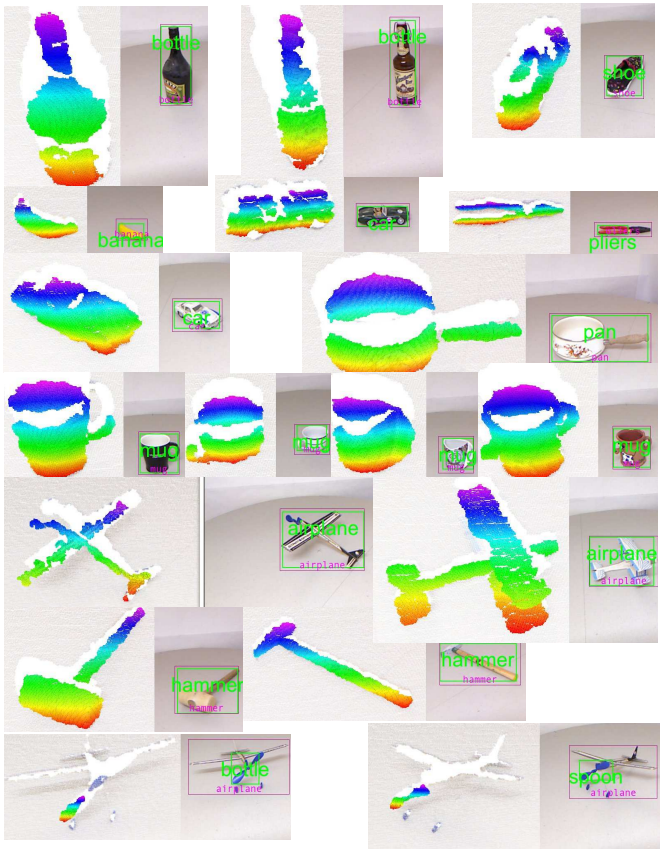


Fig. 11. A subset of the evaluation data is shown to provide to visually pinpoint the challenges of object class recognition from imperfect sensor data. The last row shows two failures where only a small part of the object is captured correctly by the sensor due to steep viewing angle. The remaining points are matched toward the closest shape in the database, which in these two cases is a bottle on the left, and a spoon on the right, both parts of the propeller of an airplane.

of shapes. This can be seen with the airplane class, as the test models are either air-filled models or build from plywood and their shape is not matching the 3D models' shapes. This is especially an important point when dealing with toys as their scale is anisotropic and parts are exaggerated.

VI. Conclusion

In this work a object class recognition system for fast and robust classification of depth image data was presented. We proposed the idea to use 3D models from the web to cope with the large intra-class variety and introduced a novel descriptor which is suited to work with 2.5D data from RGB-D sensors. The system delivers the name of the category, the best matching 3D model and the best view to a given segmented point cloud. A hashing technique was introduced to provide the necessary means to scale up to hundreds of categories while maintaining fast recognition speed. The experimental evaluation verifies the proposed approach and also highlighted the strengths and weaknesses of the descriptor. As of the modular nature of our proposed descriptor, further development of improved descriptors are within our reach and will seamlessly be built into the system for improved performance. The proposed descriptor will be available for download as part of the open source point cloud library PCL[14].

Acknowledgment

This work was conducted within the EU Cognitive Systems project GRASP (FP7-215821) funded by the European Commission.

References

[1] U. Klank, M. Z. Zia, and M. Beetz, "3d model selection from an internet database for robotic vision," in *International Conference on Robotics and Automation (ICRA)*, 2009.

[2] A. Golovinskiy, V. G. Kim, and T. Funkhouser, "Shape-based recognition of 3d point clouds in urban environments," *International Conference on Computer Vision (ICCV)*, 2009.

[3] K. Lai and D. Fox, "Object detection in 3d point clouds using web data and domain adaptation," *International Journal of Robotics Research*, 2010.

[4] C. Goldfeder, M. Ciocarlie, J. Peretzman, H. Dang, and P. Allen, "Data-driven grasping with partial sensor data," in *International Conference on Intelligent Robots and Systems (IROS)*, 2009.

[5] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[6] R. Ohbuchi and T. Furuya, "Scale-weighted dense bag of visual features for 3d model retrieval from a partial view 3d model," in *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, 2009, pp. 63 –70.

[7] H. Dutagaci, A. Godil, C. P. Cheung, T. Furuya, U. Hillenbrand, and R. Ohbuchi, "Shrec 2010 - shape retrieval contest of range scans," in *Eurographics Workshop on 3D Object Retrieval*, 2010.

[8] D. Gonzalez-Aguirre, J. Hoch, S. Roehl, T. Asfour, E. Bayro-Corrochano, and R. Dillmann, "Towards shape-based visual object categorization for humanoid robots," in *International Conference on Robotics and Automation (ICRA)*, 2011.

[9] A. E. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3d scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligences*, vol. 21, no. 5, pp. 433–449, 1999.

[10] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin, "Matching 3d models with shape distributions," in *Shape Modeling and Applications, SMI 2001 International Conference on.*, May 2001, pp. 154 –166.

[11] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3d recognition and pose using the viewpoint feature histogram," in *International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 2155 –2162.

[12] C. Y. Ip, D. Lapadat, L. Sieger, and W. C. Regli, "Using shape distributions to compare solid models," in *Proceedings of the seventh ACM symposium on Solid modeling and applications*, ser. SMA '02. New York, NY, USA: ACM, 2002, pp. 273–280. [Online]. Available: http://doi.acm.org/10.1145/566282.566322

[13] C. Fellbaum, "Wordnet: An electronic lexical database," *Cambridge, MA: MIT Press*, 1998.

[14] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *International Conference on Robotics and Automation*, Shanghai, China, 2011 2011.