Parts-based 3D object classification

Daniel Huber dhuber@ri.cmu.edu
Anuj Kapuria akapuria@ri.cmu.edu
Raghavendra Donamukkala raghu@ri.cmu.edu
Martial Hebert hebert@ri.cmu.edu

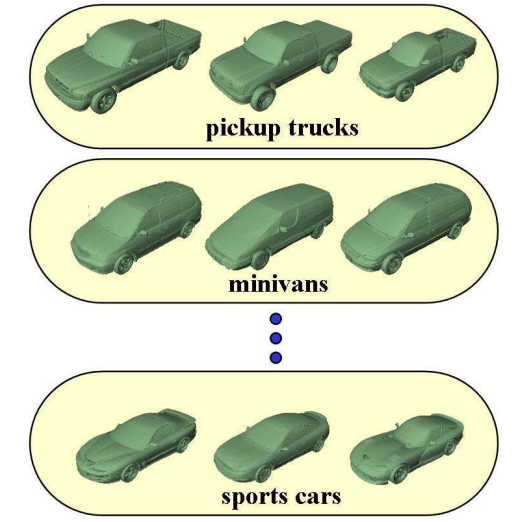
The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

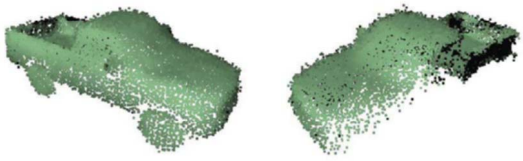
This paper presents a parts-based method for classifying scenes of 3D objects into a set of pre-determined object classes. Working at the part level, as opposed to the whole object level, enables a more flexible class representation and allows scenes in which the query object is significantly occluded to be classified. In our approach, parts are extracted from training objects and grouped into part classes using a hierarchical clustering algorithm. Each part class is represented as a collection of semi-local shape features and can be used to perform part class recognition. A mapping from part classes to object classes is derived from the learned part classes and known object classes. At run-time, a 3D query scene is sampled, local shape features are computed, and the object class is determined using the learned part classes and the part-to-object mapping. The approach is demonstrated by classifying novel 3D scenes of vehicles into eight classes.

1 Introduction

Free-form three dimensional (3D) object recognition is a well-understood problem in computer vision, with many successful approaches [5, 14, 2, 8, 18, 1]. In contrast, 3D object *classification*, where a previously unseen object must be assigned to a generic object class (e.g., as in figure 1), is still an open problem. This paper addresses this classification problem with a parts-based approach. Many existing 3D classification methods function at the object level, requiring a complete surface model of a query object in order for it to be classified [13, 11, 15, 16]. Our parts-based approach works with partial scenes (i.e., scenes that contain significant occlusion). Such self-occlusion typically occurs when a 3D sensor (e.g., a laser scanner) views a scene from a limited number of viewpoints. For example, given the query scene of a novel query object shown in figure 1(b), our goal is to classify it as belonging to the class of pickup trucks.



(a) object classes



(b) query scene (shown from two viewpoints)

Figure 1. (a) Example object classes (pickup trucks, minivans, and sports cars). Objects within a class are similar but still exhibit significant variation among exemplars. (b) Given a novel query scene such as this vehicle, our goal is to classify it as belonging to class pickup truck. Notice that the entire left hand side of the vehicle is self-occluded.

Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'04)
1063-6919/04 $20.00 © 2004 **IEEE**

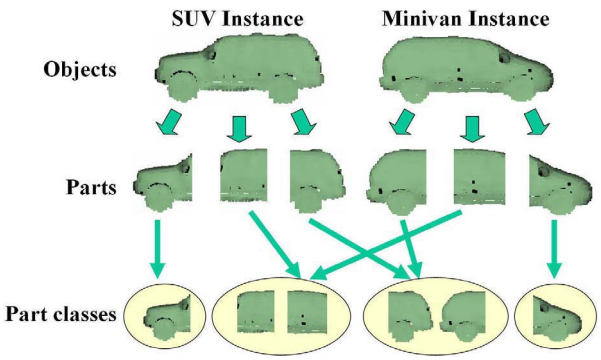


Figure 2. Overview of how object models are divided into parts and grouped into parts classes. Each training model (top row) is split into a fixed number of parts (middle row). The parts from all models are then grouped into part classes based on shape similarity (bottom row). In the example, SUVs and minivans are different object classes that have similarly shaped middle and back sections, so those parts are grouped together at the part level. The front parts are distinctive between object classes, so those parts end up in different part classes.

The main advantage of object classification (as opposed to recognition) is that it enables recognition of previously unseen objects, whereas 3D object recognition algorithms generally assume that the objects being recognized are drawn from a known database of object models (e.g., [8]).

Object classification is a natural way to deal with large numbers of objects by exploiting the similarity within large classes of objects (e.g., many objects are similar to a generic truck) and to exploit the common structure shared by all the objects in a class (e.g., all pickups have a truck bed). Classifying at the parts level enables grouping of similar structures at the sub-object level, allowing even more flexibility in class representation. For example, pickup trucks may be composed of any combination of regular/extended/crew cab and short/long truck bed.

Classification can be considered a first step in an object recognition pipeline – first determine that the object is a pickup truck, then identify it as an Chevy S10 pickup. Such a two-stage approach can speed up object recognition significantly. In many applications, a class label is sufficient, and retrieving the exact object identity is not necessary.

Our algorithm consists of a learning phase and an execution phase. The learning phase takes as input a set of 3D training objects and their associated class labels (e.g., pickups, SUVs, and sports cars) and produces a set of part classes and a mapping from part classes to object classes. First, each object is split into parts (figure 2). Our experiments use three parts per object (front, middle, and back), but other part divisions are possible. The 3D shape of each part is represented by a collection of semi-local signatures that capture the regional object shape at locations distributed over the part's surface. Parts with similar signatures have similar overall shape, which leads to a statistical method for computing a quantitative similarity measure between parts. Using this similarity measure, the parts from all objects are automatically grouped into part classes using a bottom-up hierarchical clustering algorithm. A part class is represented by the union of the signatures of the parts within that class. Novel parts with similar overall shape can then be recognized using the same statistical method that is used to compute part similarity. Given the part class labels from hierarchical clustering and the known object class labels, a mapping from part classes to object classes is computed. The part class recognizers and the part-to-object class mapping are combined in a unified framework that is used to classify novel query objects at run-time. The process is demonstrated using a database of 155 exemplars from eight classes of vehicles to classify novel query scenes.[1]

In the remainder of this paper, we begin by surveying existing 3D object classification algorithms (section 2). Section 3 describes the main components of our algorithm, while section 4 provides implementation details and experimental results.

2 Related work

Parts-based methods have been used in 3D object recognition systems since the early days of computer vision [10, 5, 13, 4]. Most existing work focuses on identification of objects from a known database rather than classification of novel objects. The typical parts-based object recognition algorithm represents parts using a small set of volumetric primitives (e.g., generalized cones [10]) or surface patches (e.g., planar patches [13] or smooth regions [5]). Models can then be represented by an attributed graph where the nodes describe parts and the edges represent part-to-part relationships such as adjacency. Recognition involves graph matching between model object graphs and a parts-based graph derived from the scene. To cope with observation variability (e.g., due to viewpoint changes or object articulation), an abstracted representation of the model parts is generally used. As such, some of these algorithms could be used for generic object classification, but we are not aware of any experimentation that demonstrates this capability. Campbell and Flynn's survey provides a comprehensive overview of 3D object recognition techniques [1].

Generic object classification algorithms can operate globally on complete objects or locally on parts (or regions)

[1]The models used in the experiment are from the commercially available Viewpoint database [19].

of objects. In the global category, Osada *et al.* [11] defined a set of global shape distribution descriptors (e.g., the distribution of distances between randomly selected pairs of surface points). Funkhouser *et al.* [15] extended this work and developed a global shape descriptor based on spherical harmonics that was used in a multi-modal interface to a 3D search engine. Tangelder and Veltkamp [16] used a volumetric similarity measure based on the Earth Mover's Distance to perform class-based object retrieval.

In the local category, Csakany and Wallace [3] developed a feature-based classification system in which features are derived from surface patches with similar curvature. Objects are classified using a complex feature matching scheme that takes into account the similarity in shape, size, and relative orientation of features. This work is similar in spirit to an earlier object recognition system by Sengupta and Boyer [13], which used random variables to describe the variability in the observation of parts and their relationships.

The method proposed by Ruiz-Correa *et al.* [12] is the prior work most similar to ours. In their method, component detectors are trained to identify regions of similar shape across all instances of a class at manually selected salient points. These detectors provide a segmentation of a surface into components, which can be regarded as parts. Then, symbolic surface detectors are trained to recognize the spatial relationship between labeled components. At run-time, the component and symbolic surface detectors are used to find points on query scenes that are similar in shape to the critical points on the training objects. The main similarity between our approaches is that we both begin by computing local surface signatures and grouping them into parts, though our definition of parts is not the same. One difference is that their components are explicitly grouped by corresponding points within a class, while ours are grouped based on similarity between parts independently of the class. As such, our method is capable of handling heterogeneous classes (i.e., classes for which there is no one-to-one correspondence between surface points on all instances).

3 Parts-based classification

In this section we give the details of our approach. Section 3.2 shows how part classes are created from training objects and how objects belonging to these part classes are recognized in query scenes. Section 3.3 describes the mapping from part classes to object classes. Finally, section 3.4 shows how these components are employed in our parts-based classification algorithm. But first, we provide a brief background on semi-local shape signatures.

3.1 Semi-local shape signatures

Previous research has shown that the shape of a 3D object is well-represented by a collection of semi-local shape signatures computed at points distributed over the object's surface [14, 2, 8]. A signature computed at a point p summarizes object shape in a compact region of support surrounding p. The dis-similarity between two signatures s_a and s_b is computed using a distance metric $D(s_a, s_b)$. The details of our choice of signature type and distance metric are given in section 4.1.

Sets of shape signatures are used to represent parts as well as part classes. Hereafter, the term "part" is used interchangeably to refer to a part label, the 3D model of the part, or the set of signatures for the part. A similar rule applies for part classes.

Let $M = \{M_1 ... M_N\}$ be a set of models (either parts or part classes). M is represented by a set of signatures $s(p_{ij})$, where p_{ij} is the i^{th} point sampled from model j. Given a new signature $s(q)$, sampled, for example, from a query scene Q, the model j for which the distance $D(s(q), s(p_{ij}))$ is minimized is said to *match* $s(q)$. This matching is simply a nearest neighbor classifier in signature space. By collecting together the matches from a set of query scene signatures $\{s(q_1), s(q_2) ... s(q_K)\}$ and normalizing by the total number of scene signatures K, we form a distribution π_M over the set of models M. When the query consists of a single part, π_M can be interpreted as the posterior probability $p(M_j|Q)$.

3.2 Forming part classes

In order to form part classes, signatures are computed at basis points uniformly sampled from the surface of every training object. Each object is then segmented into parts, resulting in a combined set of N_{r} parts from all objects $r = \{r_l : l = 1 \ldots N_{\text{r}}\}$. The details of the segmentation will be given in section 4.2. A part r_l is represented by the set of signatures whose basis points lie within part r_l according to this segmentation. These parts are then grouped into N_{R} part classes $R = \{R_i : i = 1 \ldots N_{\text{R}}\}$ based on their shape similarity. This grouping is accomplished through a bottom-up hierarchical clustering method called agglomerative clustering [7]. Agglomerative clustering is initialized by placing each part in a separate class. At each iteration, the two most similar classes are merged. The merging stops when a fixed number of part classes has been obtained.

Agglomerative clustering requires a similarity measure for clusters (i.e., part classes). A number of criteria have been put forth for computing similarity between two clusters [9]. We use the average similarity between parts in different clusters. The similarity between pairs of parts is de-

fined in terms of a similarity matrix A, where element a_{ij} stores the similarity between parts r_i and r_j.

The similarity matrix A is computed using an algorithm based on the signature matching method described in section 3.1. First, signatures are computed for points sampled uniformly over the surface of each part. Next, the similarity between each part r_i and all the remaining parts (excluding part r_i), $M_i = \{r_j : j \neq i\}$, is computed as follows: The signatures from part r_i are matched with the parts M_i to obtain π_{M_i}, the entries of which are estimates of $p(m(r_i) = r_j|r_i)$, where $m(r_i) = r_j$ denotes the event that a signature from part r_i matches part r_j from the set M_i. The symmetric similarity matrix A is formed by averaging the probabilities for pairs of parts:

$$a_{ij} = \begin{cases} \frac{p(m(r_i)=r_j|r_i)+p(m(r_j)=r_i|r_j)}{2} & \text{if} i \neq j \\ 1 & \text{otherwise} \end{cases} \quad (1)$$

Once the agglomerative clustering algorithm has determined the part class labels, each part class is simply the union of the signatures of the parts with that class label.

The (typically) large set of signatures in each part class is reduced to a smaller set of "prototype" signatures using k-means clustering in signature space [7]. The prototypes provide two benefits. First, they generalize the part class representation, which can fall victim to overfitting, hence improving the matching performance when new, unseen objects are classified. Second, there is much redundancy in part classes, since the parts within a class have similar shape and therefore contain similar signatures. Prototype signatures eliminate this redundancy.

The resulting part class models R can then be used to recognize novel parts with similar shape in a query scene. The part-level recognition uses the same signature matching method described in section 3.1 to compute the distribution over part classes π_R.

3.3 Mapping from part classes to object classes

Next, a mapping from part classes to object classes is defined. We assume that the database of training objects has been manually partitioned into N_{O} object-level classes $\mathcal{O} = \{O_j : j = 1 \ldots N_{\mathrm{O}}\}$. The part-to-object class mapping is an estimate of the probability $p(O_j|R_i)$. That is, the probability that a scene contains an object of class O_j given that a part belonging to part class R_i has been observed. Using Bayes' rule,

$$p(O_j|R_i) = \frac{p(R_i|O_j)p(O_j)}{p(R_i)} \quad (2)$$

The likelihood $p(R_i|O_j)$ is estimated from the empirical distribution of the training data. Specifically, $p(R_i|O_j)$ is the fraction of training object parts in object class O_j that belong to part class R_i. Similarly, $p(R_i)$ can be computed using the total probability theorem: $p(R_i) = \sum_j p(R_i|O_j)p(O_j)$. The prior $p(O_j)$ can be estimated from a sample of query objects, but in our implementation, we assume a uniform distribution.

3.4 Object classification framework

Using the results from the previous two sections, our overall part-based object classification algorithm can now be defined. Given a query scene, points are sampled from the scene at random. The signature $s(q)$ is computed at each sample point. Using these signatures, part-level matching is performed, as described in section 3.1, to obtain the distribution π_R. The object class is determined by computing the class likelihood L:

$$L(j) = \sum_{R_i \in \mathcal{R}} \pi_R(R_i) p(O_j|R_i) \quad (3)$$

and classifying the object as belonging to the class that maximizes this likelihood:

$$j^* = \arg\max_j L(j) \quad (4)$$

In equation 3, the entries of π_R with large values correspond to predominant part classes in the scene. For a given part class R_i, the part-to-object class mapping, distributes $\pi_R(R_i)$ among the object classes. For part classes that are common between multiple object classes, $\pi_R(R_i)$ will be diluted across those object classes. For example if a minivan/SUV-back is recognized, it is equally likely that the scene contains a minivan or an SUV. On the other hand, for part classes that are unique to one object class, $\pi_R(R_i)$ will be attributed exclusively to that object class, giving strong evidence for that object class. For example, the pickup truck back class is unique to the pickup truck object class. When a pickup truck back is recognized in a scene, it is highly likely that the object is in fact a pickup truck. In this way, our algorithm automatically determines which part classes are most salient for a given object class.

4 Implementation and results

4.1 Input data

In our experiments, we use a database of 155 commercially available 3D object models of vehicles [19]. The models were correctly scaled to their actual size and then manually divided the into eight classes (sedan, sports car, pickup, SUV, minivan, station wagon, truck, and convertible) based on their appearance and function (figure 1(a)).

The database was randomly divided into independent training and test sets, with 2/3 of the objects in each class used as training and the rest used for testing.

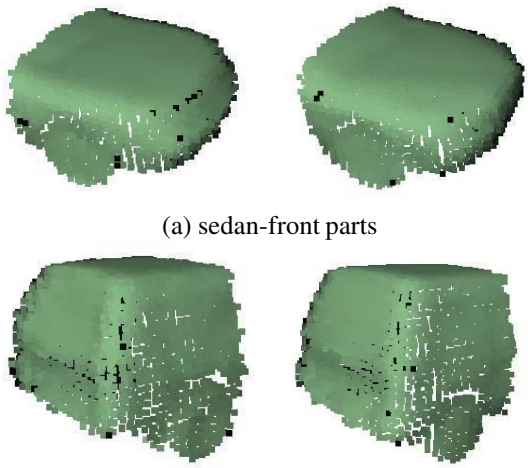
We are interested in classifying object scenes obtained from a real 3D sensor, such as a laser scanner, that observes the scene from a limited number of vantage points. This scenario is typical of a laser scanner mounted on an airborne vehicle viewing a scene from a low-altitude fly-by or of a scanner mounted on a humanoid robot observing an office environment from a standing position. In such cases, significant parts of the query object will be occluded, primarily due to self-occlusion.

Our implementation is based on 3D point clouds of data rather than on 3D surfaces (e.g., surface meshes). Working with point clouds, the standard output of many 3D sensors, allows classification of query scenes containing large levels of noise typical of the aforementioned scenarios without solving the difficult problem of surface reconstruction from noisy point data.

The point clouds for models and scenes used in our experiments were generated using a non-commercial laser scanner simulator. The point clouds for the training objects were generated by viewing the target objects at a fixed distance from eight viewpoints equally spaced over a 360 degree arc around the object, with a declination angle of 45 degrees. The query scenes were generated using seven viewpoints spaced at 15 degree azimuth intervals for a total arc of 90 degrees. The initial azimuth angle was randomly chosen in the range [0, 360) degrees and the declination angle was randomly chosen between in the range (30, 55) degrees. Gaussian random noise with $\sigma = 5$ cm was added to the scene points along the sensor's line of sight (figure 1(b)).

For our shape signatures, we use spin-images [8], though we have found that other semi-local signatures, such as a 3D extension of shape contexts [6], work as well. Spin-images encode the surface shape surrounding an oriented point[2] b by projecting nearby surface points q into a 2D histogram using the radius and height of q with respect to b in cylindrical coordinates. The spin-image histograms are 10 bins in width and height and have a cylindrical support with radius and height of 2.5 m. These parameters were chosen based on independent experiments on a separate data set. Experiments have shown that recognition using spin images is relatively insensitive to the choice of number of bins and support size. The surface normal at each basis point b (needed for spin-image computation) is estimated by fitting a plane to the points within a 20 cm radius of b using the least-squares method. The L_2 norm is used for computing signature similarity ($D(s_a, s_b)$). Other distance metrics (e.g., L_1, χ^2, and KL-divergence) were found to work equally well.

[2] An oriented point consists of a 3D point p and an orientation l, which is the local surface normal in our experiments.



(a) sedan-front parts

(b) minivan/SUV-back parts

Figure 3. Examples from two part classes a) Part class 41 contains fronts of sedans (two of which are shown). b) Part class 49 contains backs of minivans and SUVs, which are similar in shape.

4.2 Parts and part classes

There are many ways to partition an object into parts: data-driven methods, such as segmenting based on curvature; and data independent methods, such as segmenting into random, fixed sized blocks. The objects in our model database contain substantial bilateral symmetry, with primary variation occurring along the front-back axis. Therefore, we chose to divide the objects into a fixed number of segments along the front-back axis. For this work, three parts per object – front, middle, and back – were used. We have also experimented with separating articulating objects based on their articulating parts. We perform agglomerative clustering on this set of parts for the training objects, forming a set of 60 part classes. The part classes contained an average of 3000 signatures, which were reduced to 300 prototype signatures per class via k-means clustering.

Example parts from the resulting part classes are shown in figure 3. While many part classes are specific to one object class (e.g., sedan-front), a number of part classes span multiple object classes (e.g., minivan/SUV-back). Part classes that span object classes represent object regions that are not very useful for discriminating between those classes. In the case of minivans and SUVs, the front of the vehicle is the most salient for discriminating between the two classes (figure 2).

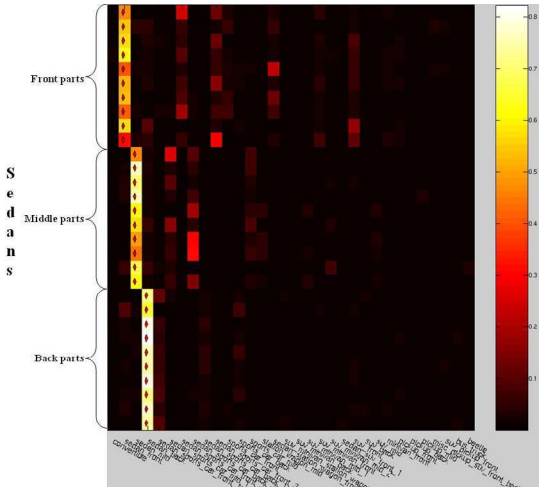
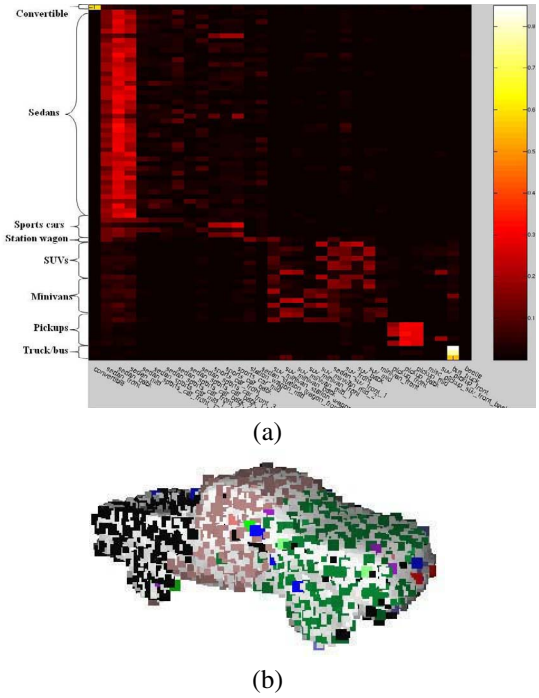


Figure 4. The confusion matrix for the part classification experiment. Rows are query parts for sedans. Columns are part classes. A red diamond indicates the recognized part class. See text for details.



(a)

(b)

Figure 5. a) The distribution π_R for each query scene. Rows are query scenes. Columns are part classes. The entries with large values correspond to the part classes that are predominant in the scenes. The part classes have been arranged in the same order as the queries (to the extent possible) to emphasize the result. The block diagonal structure indicates that the correct part classes are being detected. b) An example query scene with the sample points classified according to their most likely part class given by π_R. The color coding shows that the part classes are consistently recognized.

4.3 Part classification experiment

Using these part classes, we can perform classification at the part level. The ability to classify parts (rather than entire objects) is useful in situations where more advanced part-level processing is involved. For example, if we can correctly recognize that a given region of a scene describes a pickup truck back, those scene points could be registered with a generic, possibly deformable, model of a pickup truck back model. This idea is the subject of ongoing research.

For this experiment, we divide the query scenes into parts using the same method used for the training objects. Classification is performed by matching the signatures from each part against the part class models R, producing the distribution π_R. Figure 4 shows the confusion matrix for the experiment. Each row in the confusion matrix corresponds to π_R for a single query part, and each column corresponds to a part class. The intensity of entry (i, j) is the fraction of basis points on part i that matched to part class j. The maximum in each row is indicated by a red diamond. For visualization, part classes containing objects from only one category were combined by summing their $\pi_R(j)$ entries in each row and are treated as a single part class. For example, all part classes that contained only fronts of sedans were collapsed into a single sedan-front category.

The results show that the part classes can be used to effectively recognize novel parts. For example, points on a sedan-front query are overwhelmingly classified as the "sedan-front" part class, while most of the remaining points are distributed among mixed classes that contain sedan-fronts as well as fronts from other classes. Due to space limitations, only the queries for sedan parts are shown, but the results are typical of the complete set of queries.

4.4 Parts-based object classification experiment

This experiment demonstrates the overall parts-based object classification algorithm. We used the part classes from section 4.2 to derive the part-to-object class mapping as described in section 3.3.

Figure 5(a) shows the distribution π_R for each query scene. The columns have been combined in the same manner as in figure 4. The entries with large values of $\pi_R(i)$ correspond to the part classes that are predominant in the scenes. The block diagonal structure indicates that the correct part classes are being detected. Figure 5(b) illustrates

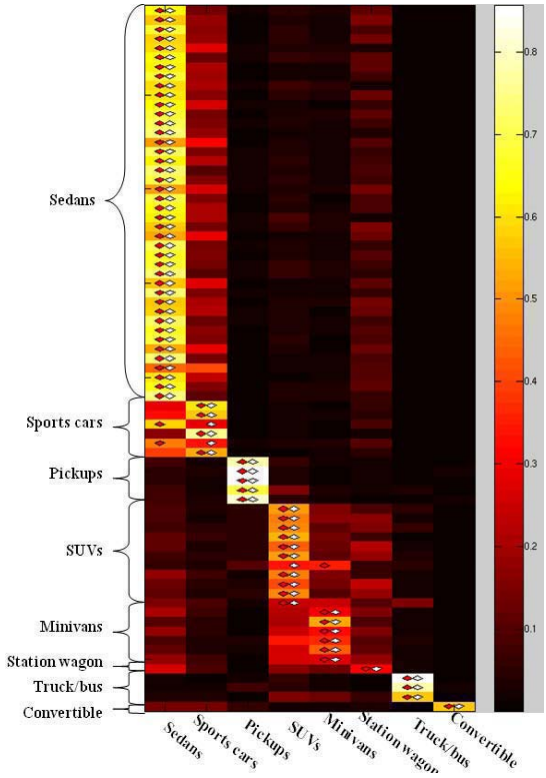


Figure 6. The confusion matrix for the parts-based classification experiment. Rows are query scenes. Columns are object classes. A white diamond indicates the ground truth class, while a red one shows the algorithm's result.

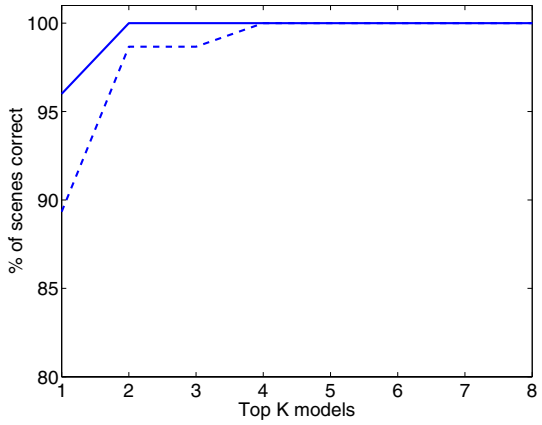


Figure 7. Performance as a function of the number of object classes considered. A query is considered correct if the ground truth class is in the top K matches. The graph compares parts-based classification (solid line) with the performance obtained using only object-level classification (dashed line).

the part-level classification of individual basis points. Each basis point on the query scene is color-coded according to the part class that best matched that point. The color coding shows three distinct parts, which verifies that the part classification is spatially consistent.

When we perform parts-based object classification on the query scenes, we obtain the confusion matrix shown in figure 6. For each query scene, we compute $L(j)$ for each object class j and normalize by the number of object classes N_O. A part is classified as the class j for which $L(j)$ is the largest. The ordering of entries in $L(j)$ from largest to smallest gives an ordered list of classes from best matching to worst. Figure 7 shows the classification performance as a function of the number of part classes considered (i.e., a query is considered correct if the ground truth class is in the top K matches). The figure shows that the parts-based classification algorithm achieves 96% correct classification, with 100% in the top 2.

For comparison, we performed object-level classification without using parts. Object-level classification operates the same way as part-level classification with two differences. First, the object-level classes O are used instead of the part-level classes R. Second, the part-to-object mapping is not needed, so the object class is simply the class j for which $\pi_O(j)$ is the maximum. For the object based classification, the performance rate is under 90%, and the top $K = 4$ classes must be considered before we achieve a perfect score.

A closer analysis of the confusion matrix in figure 6 shows that the classification algorithm follows our intuitive notion of object similarity. For example, the figure shows that sedans and sports cars are similar in shape as are minivans and SUVs. Sports cars that are more "sporty" (e.g., the TransAM – first sports car – or the Viper – fourth sports car) are more easily distinguished from sedans than those that

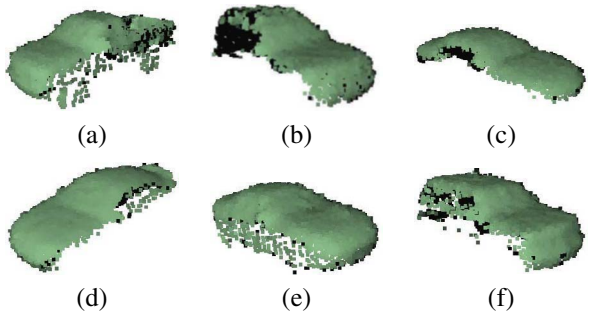


Figure 8. Examples of correctly classified query scenes (a-c) and all three incorrectly classified queries (d-f).

are more sedan-like (e.g., Mercury Cougar – third sports car). Station wagons are similar to sedans and SUVs and minivans. Figure 8 shows some examples of queries that were correctly classified and all three queries that were incorrectly classified. Two of the failures were sports cars that were incorrectly classified as sedans, and one was an SUV that was classified as a minivan (note that the likelihoods for each class are almost equal in this last case).

5 Summary and future work

In summary, we have described an algorithm for part-based classification of 3D objects. The algorithm forms a set of part classes by grouping parts from example objects by similarity, and learns a mapping from part class to object class. At run time, the part classes are used to detect the dominant part classes in a query scene, and the resulting distribution over part classes is combined with the part-to-object class mapping to determine the query object class.

In the future, we plan to focus on three areas: handling clutter, alternate part segmentations, and geometric part analysis. While our experiments tested classification of objects in isolation, real 3D scenes generally contain significant non-target regions, which are considered clutter. One promising approach for handling clutter is to explicitly model clutter (e.g., using sets of signatures computed for training clutter scenes). Another idea is to adapt geometric consistency constraints, which have been effective for object identification, to the classification problem. Geometric consistency constraints take advantage of the fact that pairs of points on the scene should match to points on a model that have the same relative position, whereas pairs of points in clutter are unlikely to exhibit such geometric consistency. Our second area of future work is to investigate alternative part segmentation methods. While the front/middle/back segmentation scheme works well for vehicles, it is a rather crude approach. We are investigating general-purpose methods for segmenting objects that finds parts that are similar within classes and unique between classes. Finally, we are looking at geometric classification algorithms. For example, encoding the relative positions and orientations of parts in an object class in a manner similar to Weber's part constellations [17] could prove to be effective.

6 Acknowledgements

This work was supported by the DARPA E3D program (F33615-02-C-1265). We would like to thank Harpreet Sawhney, Bogdan Matei, Ying Shan, and Yi Tang at Sarnoff Corporation, Andrea Frome at UC Berkeley, and Nicolas Vandapel for helpful comments and discussions on this work.

References

[1] R. Campbell and P. Flynn. A survey of free-form object representation and recognition techniques. *Computer Vision and Image Understanding (CVIU)*, 81(2):166–210, 2001.

[2] C. S. Chua and R. Jarvis. Point signatures: a new representation for 3D object recognition. *International Journal of Computer Vision*, 25(1):63–85, Oct. 1997.

[3] P. Csakany. Representation and classification of 3D objects. *IEEE Trans. on Systems, Man, and Cybernetics, Part B*, 33(4):638–647, Aug. 2003.

[4] S. Dickinson, D. Metaxas, and A. Pentland. The role of model-based segmentation in the recovery of volumetric parts from range data. *IEEE Trans. on Pattern Analysis and Mach. Int. (PAMI)*, 19(3):259–67, Mar. 1997.

[5] T. Fan, G. Medioni, and R. Nevatia. Recognizing 3D objects using surface descriptions. *IEEE Trans. on Pattern Analysis and Mach. Int. (PAMI)*, 11(11):1140–57, Nov. 1989.

[6] A. Frome, D. Huber, R. Kolluri, T. Bulow, and J. Malik. Recognizing objects in range data using regional point descriptors. In *European Conf. on Computer Vision (ECCV)*, May 2004.

[7] A. Jain and R. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Inc., 1988.

[8] A. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Trans. on Pattern Analysis and Mach. Int. (PAMI)*, 21(5):433–49, May 1999.

[9] S. Kamvar, D. Klein, and C. Manning. Interpreting and extending classical agglomerative clustering algorithms using a model-based approach. In *Proceedings of the International Conference on Machine Learning (ICML)*, July 2002.

[10] R. Nevatia and T. Binford. Description and recognition of curved objects. *Artificial Intelligence*, 8(1):77–98, Feb. 1977.

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

[12] S. Ruiz-Correa, L. Shapiro, and M. Meila. A new paradigm for recognizing 3D object shapes from range data. In *Intl. Conf. on Computer Vision (ICCV)*, pages 1126–1133, Oct. 2003.

[13] K. Sengupta and K. Boyer. Organizing large structural modelbases. *IEEE Trans. on Pattern Analysis and Mach. Int. (PAMI)*, 17(4):321–32, Apr. 1995.

[14] F. Stein and G. Medioni. Structural indexing: efficient 3D object recognition. *IEEE Trans. on Pattern Analysis and Mach. Int. (PAMI)*, 14(2):125–45, Feb. 1992.

[15] T. Funkhouser et al. A search engine for 3d models. *ACM Trans. on Graphics*, 22(1):83–105, Jan. 2003.

[16] J. Tangelder and R. Veltkamp. Polyhedral model retrieval using weighted point sets. In *Proceedings of Shape Modeling International*, pages 119–29, May 2003.

[17] M. Weber, M. Welling, and P. Perona. Unsupervised learning of models for recognition. In *European Conf. on Computer Vision (ECCV)*, pages 18–32, June 2000.

[18] J. Wyngaerd, L. V. Gool, R. Kock, and M. Proesmans. Invariant-based registration of surface patches. In *Intl. Conf. on Computer Vision (ICCV)*, pages 301–6, Sept. 1999.

[19] Viewpoint corporation – www.viewpoint.com.