Integrating Algebraic Functions of Views with Indexing and Learning for 3D Object Recognition

Wenjing Li and George Bebis
Computer Vision Laboratory
University of Nevada, Reno
(wjli,bebis)@cs.unr.edu

Nikolaos Bourbakis
Information Technology Research Institute
Wright State University
bourbaki@cs.wright.edu

Abstract This paper focuses on the problem of 3D object recognition from different viewing angles and positions. In particular, we propose a new approach that integrates Algebraic Functions of View (AFoVs) with indexing and learning. During training, we consider groups of point features and we represent a *sparse* number of views that they can produce in a k-d tree. Moreover, we learn the manifold formed by a *dense* number of views using mixture models and Expectation-Maximization (EM). Learning takes place in a “universal”, lower-dimensional, space computed through Random Projection (RP). The images that a group of model points can produce are computed off-line using AFoVs by combining a small number of reference views that contain the group. Rigidity constraints are imposed during this step to remove unrealistic views. During recognition, groups of point features are extracted from the scene and used to retrieve from the k-d tree the most feasible model groups that might have produced them. To reduce the number of hypotheses for verification, we rank them by computing the probability that the hypothesized model group is present in the scene. Only hypotheses ranked high enough are considered for further verification. The proposed method has been evaluated using both artificial and real data, illustrating good performance.

Keywords: object recognition, algebraic functions of views, indexing, learning

1 Introduction

Building systems capable of recognizing relevant objects in their environment with accuracy and robustness has been a difficult and challenging task in computer vision [24]. Recognition is difficult because the appearance of an object can have a large range of variation due to photometric effects, scene clutter, changes in shape, and viewpoint changes. As a result, different views of the same object can give rise to widely different images.

Our emphasis in this paper is on handling variations in shape appearance due to viewpoint changes more efficiently and effectively. Typical strategies to cope with this problem include the use of invariants [18], explicit 3D models [13, 6], and multiple-views [15, 25, 19]. Each of these strategies, however, has several drawbacks. For example, no general case invariants exist under 3D perspective projection [8], 3D models are not always easy to obtain, and multiple-view methods require storing a large number of views.

In [5, 4], we proposed a new object recognition approach based on AFoVs [26, 21]. AFoVs provide a powerful framework for investigating variations in the shape appearance of a 3D object due to viewpoint changes. In particular, the main result of AFoVs states that “the variety of 2D views depicting the shape appearance of a 3D object can be expressed as a combination of a small number of 2D views of the object”. This suggests a simple but powerful framework for predicting shape appearance: “novel 2D views of a 3D object can be recognized by combining a small number of known 2D views of the object”. The main advantage of this framework is that it does not rely on invariants or 3D models. In fact, no camera calibration or 3D scene recovery are necessary. Also, it is fundamentally different from multiple-view approaches which perform matching by comparing novel views to prestored views of the object (i.e., reference views). In contrast, AFoVs predict the shape appearance of a 3D object by combining a small number of reference 2D views of the object.

Although interesting and appealing, the underlying theory of AFoVs is based on several restrictive assumptions, making AFoVs of less practical use. For example, it assumes that the correspondences between features in the novel view and features in the reference views are known. Also, it assumes that the values of the parameters of the AFoVs are known. We have addressed these issues in our previous work [4, 5, 2, 3] by (1) coupling AFoVs with indexing, to bypass the correspondence problem, and (2) estimating the ranges of values that the parameters of AFoVs can assume using Singular Value Decomposition (SVD) [10] and Interval Arithmetic (IA) [17]. Using two reference views per object, we demonstrated the feasibility of our approach by recognizing novel views of the object from different viewpoints.

This work builds upon our previous work on object recognition using AFoVs with the goal of improving its efficiency and performance. Specifically, an important advantage of using AFoVs for recognition is that they allow us to compute off-line the space of 2D views that a 3D object can produce using a small number of 2D reference views of the object. During the training phase of our algorithm, we use this idea to sample the space of 2D views of an object (i.e., by sampling the space of the AFoVs parameters) and represent information about them in a hash table. This information is used during recognition to form hypotheses between the models and the scene. We have improved both the training and recognition phases of our method in several important ways.

First, when generating the sampled views of an object, we now impose a pair of rigidity constraints to avoid representing unrealistic views in the hash table. This saves both space at indexing and reduces the number of invalid hypotheses during recognition. Second, the recognition performance of the method depends on the number of sampled views represented in the hash table. Increasing this number would improve performance, however, it would also increase space requirements as well as recognition time due to an expected increase in the number of hypotheses generated. Here, we propose a two-stage scheme to deal with these issues. In the first stage, we represent a *sparse* number of sampled views in an indexing structure. This stage allows us to generate hypotheses efficiently through indexing while keeping space requirements low. To account for the sparse number of views used, we have replaced hashing by a more powerful indexing scheme based on k-d trees [12, 23] which perform nearest-neighbor search as opposed to range search performed by hashing. In the second stage, we learn the manifold formed by a *dense* number of sampled views using the EM algorithm [20]. Learning takes place in a "universal", lower-dimensional, space computed through RP [9, 7]. This stage reduces storage requirements considerably (i.e., only a few parameters need to be stored for each manifold) and it allows us to rank the hypotheses generated by the first stage. Ranking saves significant time during verification since the most likely hypotheses are verified first.

The rest of the paper is organized as follows. In Section 2, we provide background information on AFoVs and our previous work on recognition using AFoVs. The proposed improved recognition approach is presented in detail in Section 3. Section 4 presents our experimental procedures and results using both artificial and real 3D objects. Finally, our conclusions and plans for future work are given in Section 5.

2 Background on AFoVs

Simply speaking, AFoVs are functions which express a relationship among a number of views of an object in terms of their image coordinates alone. In particular, Ullman and Basri [26] showed that if we let an object undergo 3D rigid transformations and assume that the images of the object are obtained by orthographic projection followed by uniform scaling (i.e., a good approximation to perspective projection when the camera is far from the object), then novel views of the object can be expressed as a *linear* combination of three other views of the same object (i.e., reference views). This result can be simplified by removing the orthonormality constraint associated with the rotation matrix. In this case, the object undergoes 3D linear transformations in space and AFoVs become simpler, involving only two reference views. Specifically, let us consider two reference views V_1 and V_2 of the same object which have been obtained by applying different linear transformations, and two points $p' = (x', y')$, $p'' = (x'', y'')$, one from each view, which are in correspondence. Then given a novel view V of the same object which has been obtained by applying another linear transformation and a point $p = (x, y)$ which is in correspondence with point p' and p'', the coordinates of p can be expressed as a linear combination of the coordinates of p' and p'' as

$$\begin{aligned} x &= a_1x' + a_2y' + a_3x'' + a_4 \\ y &= b_1x' + b_2y' + b_3x'' + b_4 \end{aligned} \quad (1)$$

where the parameters $a_j, b_j, j = 1, ..., 4$, are the same for all the points which are in correspondence across the three views. It should be noted that the above equations can be rewritten using the y-coordinates of the second reference view instead. Also, without additional constraints, it is impossible to distinguish between rigid and non-rigid transformations of the object. To impose rigidity, additional constraints must be satisfied [26]. The above results hold true in the case of objects with sharp boundaries, however, similar results exist in the case of objects with smooth boundaries [1] as well as non-rigid objects [26] (i.e., more reference views are required in these cases). The extension of AFoVs to the case of perspective projection has been carried out in [21, 11].

Given a novel view of an object, AFoVs can be used to predict the image coordinates of point features in the novel view by appropriately combining the image coordinates of corresponding point features across the reference views. We have employed this idea in our previous work to recognize unknown views of an object from a small number of reference views of the same object, assuming orthographic projection and linear 3D transformations [5, 4]. To bypass the correspondence problem, we proposed coupling AFoVs with indexing. During indexing, we used AFoVs to predict the views that groups of point features can produce and represented the predictions in a hash table. During recognition, groups of points were extracted from the scene and used to retrieve from the hash table hypotheses (i.e., model groups that might have produced them). Each hypothesis was then verified to find the correct model in the scene. To sample the space of views that groups of model points can produce, we sampled the space of parameters of the AFoVs. For this, it is necessary to estimate the allowable ranges of values that the parameters of the AFoVs can assume. We dealt with this issue by introducing a methodology based on SVD [10] and IA [17].

3 The Proposed Framework

The proposed recognition framework has two main phases: training and recognition, as shown in Fig. 1. Compared to our previous work, both phases have been improved in several important ways. First of all, we have improved the feature extraction step of our algorithm in order to obtain more stable and robust point features. While the point features in our previous work were extracted using a corner detector [22], in this work, we extract point features corresponding to intersections of lines forming convex groups [14]. Second, when sampling the space of views that groups of model points can produce, we impose a pair of rigidity constraints to eliminate unrealistic views. This saves both space during indexing and time during recognition (i.e., reduces the number of invalid hypotheses). Third, we propose a more effective scheme to represent the space of views that groups of model points can produce. This reduces the space requirements of our method considerably, a major issue involved in our previous work.

This scheme is based on two distinct stages. The first stage relies on indexing as before, however, to keep space requirements low we index only a *sparse* number of sampled views per model. To improve the quality of the hypotheses generated during recognition, we have replaced hashing, which performs a range search, with a more powerful indexing scheme based on k-d trees [12, 23], which performs nearest-neighbor search. In the second stage, we learn the manifold formed by a *dense* number of sampled views per model using the EM algorithm [20]. Learning takes place in a “universal”, lower-dimensional, space computed through RP [9, 7]. The only information that needs to be stored at this stage is just a few parameters for each manifold.

The main purpose of the first stage is to generate hypothetical matches between the models and the scene very fast. Although this is to be expected by using indexing, it is also reasonable to expect that this step would generate a large number of hypotheses, many of which would be invalid due to the sparseness constraint. The main purpose of the second stage to filter out quickly and inexpensively as many invalid hypotheses as possible. This stage provides a way to rank each hypothesis prior to verification. This saves time since only hypotheses ranked high enough are considered for further verification. Verification is performed by matching the predicted model appearances with the scene.

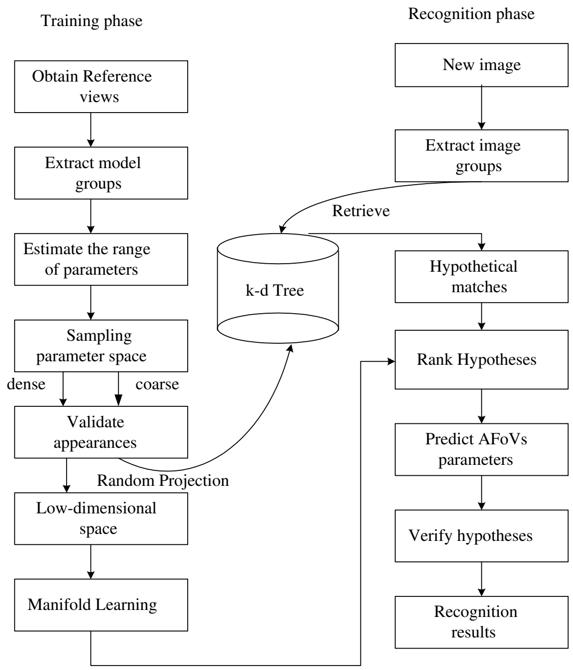


Figure 1: The proposed framework

3.1 Eliminating Unrealistic Views

When sampling the parameters of the AFoVs to generate the sampled views of an object using Eq. (1), it is possible to generate views that are not realistic in practice. This is because of two reasons. First, Eq. (1) corresponds to the case of linear 3D transformations, a superset of 3D rigid transformations. Second, the interval solutions for the parameters of the AFoVs are not tight [5, 4]. We can eliminate such views by imposing a pair of rigidity constraints [26]. This requires some information from the reference views. Specifically, if we assume that the first reference view has been obtained by the identity transformation, and the second reference view has been obtained from the first reference by rotation R, then the parameters of the AFoVs of Eq. (1) $(a_1, a_2, a_3, b_1, b_2, b_3)$ must satisfy two following two constraints:

$$\begin{aligned} a_1b_1 + a_2b_2 + a_3b_3 + (a_1b_3 + a_3b_1)r_{11} \\ +(a_2b_3 + a_3b_2)r_{12} = 0 \end{aligned} \tag{2}$$

$$\begin{aligned} a_1^2 + a_2^2 + a_3^2 - b_1^2 - b_2^2 - b_3^2 - 2(b_1b_3 - a_1a_3)r_{11} \\ -2(b_2b_3 - a_2a_3)r_{12} = 0 \end{aligned} \tag{3}$$

where r_{11} and r_{12} are the first two elements of the rotation matrix R. By applying these two constraints, the sampled views can be effectively refined. In practice, we implement this test by checking whether the expressions on the left hand-side are less than a small threshold. If the matrix R is not known, a third view can be used to recover the values of r_{11} and r_{12} by solving two linear equations [26]. In practice, we can apply specific rotations (e.g., by placing the object on a turn table) to get the required entries of R .

3.2 Indexing Based on k-d Tree

To reduce space requirements but also to enable fast hypothesis generation, we index only a *sparse* number of sampled views per object. In our previous work, hashing was used to retrieve the closest model views to a given novel view. Hashing, however, would not be appropriate now since it does a range search. In contrast, employing more powerful indexing schemes performing nearest-neighbor search would be more appropriate due to the sparseness constraint.

Perhaps the most widely used algorithm for performing nearest-neighbor search in multiple dimensions is a static space partitioning technique based on a k dimensional binary search tree, called k-d tree [12, 23]. The k-d tree is a data structure which partitions the space hierarchically using hyper-planes. In a typical k-d tree [12], the partition hyper-plane is perpendicular to the coordinate axes. In this work, we use the Sproull k-d tree [23], a radical refinement to the traditional k-d tree. The choice of the partition plane is not orthogonal or "coordinate based". Instead, it is chosen by computing the principal eigenvector of the covariance matrix of the points.

Similarly to our previous work, we store information only about the x-coordinates of the sampled views in the k-d tree. This is because the process generating the x-coordinates of the sampled-views is the same to that generating the y-coordinates of the sampled views [5, 4]. During recognition, however, the k-d tree must be accessed twice. First, the x-coordinates of the novel view are used to generate hypotheses predicting the a_j parameters of the AFoVs, and second, the y-coordinates of the novel view are used to generate hypotheses predicting the b_j parameters of the AFoVs.

3.3 Manifold Learning

Although AFoVs allow us to generate the views that an object can produce efficiently, representing this information compactly would be critical. We have decided to use statistical learning techniques for this purpose. In particular, the views that an object can produce form a manifold in a lower-dimensional space. This manifold can be learned efficiently using mixture models and the EM algorithm. The main advantage in our case is that we can generate a large number of sampled views using AFoVs, therefore, improving our chances to capture the true structure of the manifold. This is in contrast to similar approaches in the literature where a large number of images is required to ensure good results [19].

Mixture models are a type of density model which comprises a number of component functions, usually Gaussian. These component functions are combined to provide a multi-modal density. In the past, they have been employed to model the color distribution of objects for real-time segmentation and tracking [16]. Mixture models provide greater flexibility and precision in modelling the underlying statistics of sample data. Once a model is generated, conditional probabilities can be computed. Let the conditional density for the sample data ξ belonging to an object O be a mixture of M component densities:

$$p(\xi|O) = \sum_{j=1}^{M} p(\xi|j)\pi(j) \tag{4}$$

where the mixing parameter $\pi(j)$ corresponds to the prior probability that data ξ was generated by component j and where $\sum_{j=1}^{M} \pi(j) = 1$. Here, each mixture component is a Gaussian with mean μ and covariance matrix Σ, i.e.

$$p(\xi|j) = \frac{1}{(2\pi)^{N/2}|\Sigma_j|^{\frac{1}{2}}} e^{-\frac{1}{2}(\xi-\mu_j)^T \Sigma_j^{-1}(\xi-\mu_j)} \qquad (5)$$

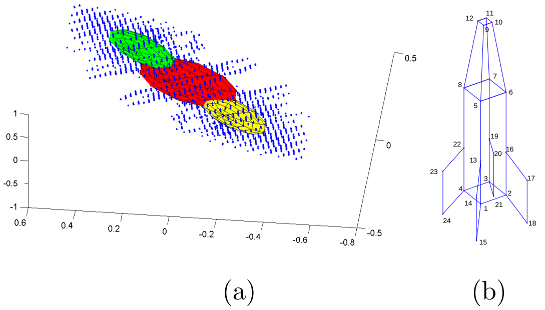


Figure 2: The mixture model obtained (shown in (a)) for a group of 8 point features (9 to 16) from the artificial object shown in (b).

The EM algorithm [20] is a well established maximum likelihood estimation algorithm for fitting a mixture model to a set of training data. It is iterative with the mixture parameters being updated in each iteration. It has been shown that it monotonically increases the likelihood with each iteration, converging to a local maximum. EM suffers from singularities in the covariance matrix when the dimensionality of the data is high. We encountered similar problems here using large groups of point features. To avoid these problems, we used RP to project the sampled views into a low dimensional space before running the EM algorithm [9, 7]. The same RP was used for each object, therefore, we refer to this low-dimensional space as the "universal" object space.

Coupling RP with the EM algorithm has shown to have some important advantages. First, the data from a mixture of M Gaussian functions can be projected into just $O(logM)$ dimensions while still retaining the approximate level of separation between clusters. This projected dimension is independent of the number of data points and of their original dimension. Second, even if the original clusters are highly eccentric (i.e, far from spherical), RP will make them more spherical. In our case, we used RP to reduce the dimension of our data down to 3.

It should be noted that the EM algorithm requires the *a-priori* selection of the number M of components. Here, we determined the number of components automatically by using a method based on mutual information [27]. Fig. 2 shows the mixture model obtained for a group of 8 points from an artificial object (i.e., rocket) used in our experiments.

3.4 Hypothesis Ranking

Each hypothesis generated by the k-d tree search, is ranked by computing its probability using the mixture models described in the previous subsection. Specifically, for each test view, we compute two probabilities, one from the x-coordinates of the view and the other from the y-coordinates of the view. The overall probability for a particular hypothesis is then computed as follows:

$$p(j) = \frac{log(p_x(j) * p_y(j))}{log(\max(p_x(i)) * \max(p_y(i)))} \qquad (6)$$

where $i = 1..., H$. H the number of hypotheses generated by the k-d tree search, $p_x(j)$ and $p_y(j)$ are the probabilities from the x- and y-coordinates of the current hypothesis, and $p(j)$ is the overall probability of the current hypothesis.

4 Experimental Results

We describe below a number of experiments to demonstrate the proposed approach. To enable robust feature extraction, we consider objects containing sharp edges. Each object view is represented by a set of point features corresponding to intersections of line segments comprising the boundary of the object. To account for occlusion, we use subsets (i.e., groups) of point features as opposed to using all point features. In practice, we can select salient groups of point features, for example, corresponding to intersections of perceptually important groups of lines (e.g., convex groups [14]). In this case, each point feature has a certain ordering in the group which can facilitate matching.

4.1 Artificial Objects

A set of 10 artificial 3D models (i.e., car, truck, tank, rocket, airplane, monitor, bench, house, desk, stapler) was used to evaluate the performance of the proposed approach. Each model was represented by 2 reference views which were obtained by applying different orthographic projections on the 3D models. For each model, we considered all possible groups having 8 point features (i.e., 22 groups on average for each model). First, a coarse k-d tree was built by storing information about a sparse set of views that the model groups can produce. A total of $2,242$ sampled views were generated and stored in the k-d tree. Then, a dense number of views was generated for each model group and its manifold was learned using the EM algorithm. The ratio of sparse to dense views used was 2%.

The test views were generated by applying random orthographic projections on the 3D models. We also added 3 pixels random noise to point features of the test views. We did not assume any knowledge of the point feature correspondences between model and scene groups, however, we did assume that point features have certain ordering in the group (i.e., see our discussion in the previous subsection). Assuming that there is no easy way to select the initial point feature in a group, we considered all possible circular shifts (i.e., 8 in our case) of point features when searching the k-d tree.

The query results for three of our models (i.e., car, tank, and rocket) are shown in Table 1, as well as their rankings, computed by the mixture models. The first column in Table 1 indicates the query group and the model it comes from, the second column indicates the circular shift applied (i.e., "shift 0" corresponds always to the correct hypothesis), and the third column shows the model candidates retrieved by the k-d tree query. The fourth column of the table shows the un-normalized probabilities computed from the x- and y-coordinates respectively while the overall probabilities, computed using Eq. (6), are shown in the last column. The overall probabilities indicate the level of confidence for each hypothesis and are used to rank them.

Table 1: Probabilistic ranking for the queries

Query	shift	Cand.	Prob.	Rank
Car-g1	0	Car-g1	(99.11,34.07)	1.00
	6	Bench-g5	(29.59,28.89)	0.83
	4	Car-g1	(99.77,0.73)	0.53
	7	Car-g2	(0,0),	0
Car-g2	0	Car-g2	(164.65,50.85)	1
	4	Rocket-g2	(0.48,0.22)	0
Tank-g1	0	Tank-g1	(74.35,38.73)	1
	3	Monitor-g1	(18.54,2.10)	0.46
	4	Monitor-g1	(0.00,22.46)	0
	4	Bench-g1	(0,0)	0
Tank-g2	0	Tank-g2	(227.30,85.29)	1
Tank-g3	0	Tank-g3	(1158.0,905.8)	1
	3	Truck-g1	(179.73,263.93)	0.78
	3	Rocket-g3	(39.43,60.15)	0.56
	4	Rocket-g2	(43.72,54.30)	0.56
	2	Car-g1	(22.49,5.8191)	0.35
	6	Car-g1	(18.54,4.22)	0.31
	7	House-g1	(0,0)	0
Rocket-g1	0	Rocket-g1	(539.1,1922.9)	0.94
	4	Rocket-g1	(674.4,3562.1)	1
Rocket-g2	0	Rocket-g2	(32.66,171.94)	1
	4	Bench-g2	(0,0)	0
Rocket-g3	0	Rocket-g3	(21.45,137.07)	1
	4	Bench-g4	(0,87.22)	0

Once the hypotheses have been ranked, we apply further verification to those hypotheses ranking high enough (i.e., 0,9 or above). In this case, the parameters of the AFoVs are estimated accurately from the hypothetical match using a least-squares approach such as SVD. Using the estimated AFoVs parameters, we then predict the appearance of the candidate model using Eq. (1) and compare it with the scene. Computing the mean square error (MSE) between the predictions and the scene provides a measure of similarity for deciding the presence of the candidate model in the scene. Fig. 3 shows the verification results for the hypotheses listed in Table 1 in the case of the rocket model. We received extremely small MSE errors in all of our experiments using artificial data sets.

Table 1 shows that the hypotheses with the highest probabilities were also the correct hypotheses in all cases except in one case (i.e., Rocket-g1). In that case, the first group of the rocket model was matched to the model assuming two different solutions due to symmetry, as shown in Fig. 3(a). We denote the test group of point features using "+", while the blue lines indicate the predicted views. Such symmetric solutions can be resolved later during the verification step.

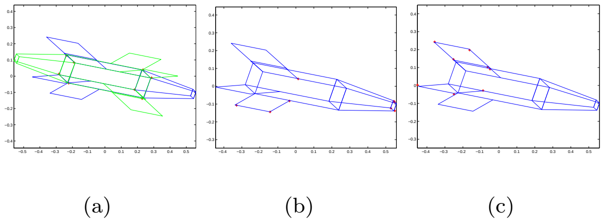
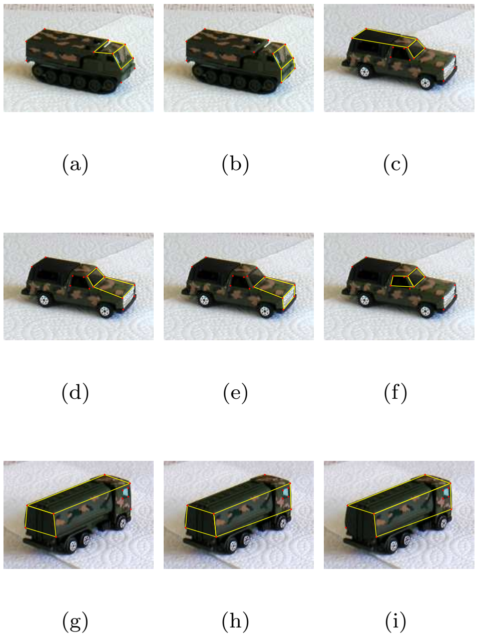


Figure 3: Verification results for the rocket query (a)group 1 (b)group 2 (c)group 3

4.2 Real Objects

In this section, we demonstrate the proposed approach using the real 3D objects shown in Fig. 4. Each object was represented from a particular aspect only using two reference views. Fig. 4 shows the first of the reference views for each model. The second reference view was obtained by rotating the object about the y-axis (by a small angle (e.g., 10 to 20 degrees). Knowledge of the rotation between the reference views allows us to enforce the rigidity constraints as discussed in Section 3. In these experiments, we used groups containing 6 point features. These groups were formed by two convex subgroups [14] of size 4, having two point features in common. Fig. 4 shows the groups used for each of our models. To order the points in a group during recognition, we choose the common points as starting points and trace the rest of the points counterclockwise. A sparse set of 2060 sampled views of the groups were represented in a k-d tree. The manifold of each group was then learned using the EM algorithm. The ratio of sparse to dense views used in this case was 35%.

Fig. 5 shows some of the test views used in our experiments. As before, we extract groups of point features from the scene and we use them to retrieve hypothetical matches from the k-d tree. Each hypothesis is then ranked using the mixture models of the model groups. We do not present detailed information in this case due to lack of space, however, it should be mentioned that the correct model was always ranked first or second in our experiments. The verification results can be seen in Fig. 5 where the yellow lines correspond to the scene groups and the red lines to the predicted models. The models present in the scene were recognized correctly in all cases. The MSE error was less than 0.6. Fig. 5(i) shows a case where an object not belonging to the set of models is present in the scene. This object produced a MSE higher than 3 and it was rejected at the verification stage.


(a) (b) (c)

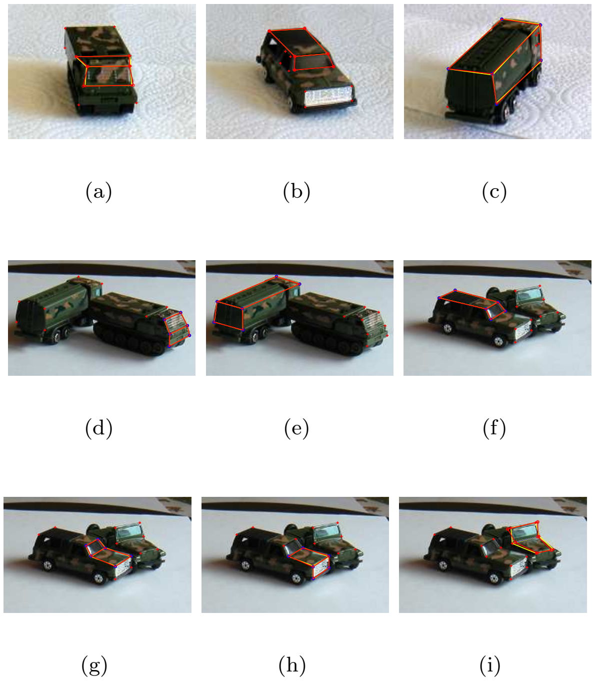
(d) (e) (f)

(g) (h) (i)

Figure 4: Models (a)-(b) two groups of model1, (c)-(f) 4 groups for model2, (g)-(i) 3 groups of model3

5 Conclusions

In this paper, we presented a new approach for 3D object recognition from different viewing angles and positions. The new approach builds on our previous work on using AFoVs for 3D recognition. Specific improvements include (1) eliminating unrealistic views during indexing by using rigidity constraints, (2) reducing space requirements significantly by combining indexing with learning, and (3)improving recognition time by ranking hypotheses based on probabilistic mixture models. For future research, we plan to perform larger scale experiments and investigate the problem of selecting a small but sufficient number of reference views to be able to recognize a given object from every possible aspect. One idea, for example, is choosing the reference views based on the quality of the groups of point features that they contain. Another research direction is combining AFoVs (i.e., a powerful framework for representing changes in geometrical appearance), with empirical models of appearance (i.e., eigenspace methods [19]).


(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 5: Verification results (a)-(c) novel views, (d)-(h) novel views with occlusion (i) unknown object

Acknowledgements: This work was supported by the Office of Naval Research (ONR) under grant N00014-01-0781.

References

[1] R. Basri and S. Ullman. The alignment of objects with smooth surfaces. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 57(3):331–345, 1993.

[2] George Bebis, M. Georgiopoulos, N. V. Lobo, and M. Shah. Learning affine transformations of the plane for model-based object recognition. *13th*

International Conference on Pattern Recognition, IV:60–64, 1996.

[3] George Bebis, M. Georgiopoulos, N. V. Lobo, and M. Shah. Learning affine transformations. *Pattern Recognition*, 32:1783–1799, 1999.

[4] George Bebis, Michael Georgiopoulos, Mubarak Shah, and Niels da Vitoria Lobo. Algebraic functions of views for model-based object recognition. *International Conference on Computer Vision*, pages 634–639, 1998.

[5] George Bebis, Michael Georgiopoulos, Mubarak Shah, and Niels da Vitoria Lobo. Indexing based on algebraic functions of views. *Computer Vision and Image Understanding*, 72(3):360–378, Dec. 1998.

[6] Jeffrey S. Beis and David G. Lowe. Indexing without invariants in 3d object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(10):1000–1015, Oct. 1999.

[7] Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction: application to image and text data. *in Proc. of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 245–250, Aug. 2001.

[8] D. Clemens and D. Jacobs. Space and time bounds on indexing 3d models from 2d images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):1007–1017, 1991.

[9] Sanjoy Dasgupta. Experiments with random projection. *In Proc. of 16th Conference on Uncertainty in Artificial Intelligence*, 2000.

[10] W. Press et al. *Numerical Recipies in C: The Art of Scientific Programming*. Cambridge University Press,UK, 1990.

[11] O. Faugeras and L. Robert. What can two images tell us about a third one? *In Proc. of third European Conference on Computer Vision*, pages 485–492, 1994.

[12] Jerome H. Friedman, Jon Lousi Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathmatical Software*, 3(3):209–226, Sep. 1977.

[13] D. Jacobs. Mathcing 3d models to 2d images. *International Journal of Computer Vision*, 21(1/2):123–153, 1997.

[14] David W. Jacobs. Robust and efficient detection of salient convex groups. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(1):23–37, Jan. 1996.

[15] Y. Lamdan, J. Schwartz, and H. Wolfson. On recognition of 3d objects from 2d images. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1407–1413, 1988.

[16] Stephen J. McKenna, Yogesh Raja, and Shaogang Gong. Tracking colour objects using adaptive mixture models. *Image and Vision Computing*, 17:225–231, 1999.

[17] R. Moore. *Interval analysis*. Prentice-Hall, 1966.

[18] L. Mundy and A. Zisserman. *Geometric Invariance in Computer Vision*. MIT Press, 1992.

[19] Hiroshi Murase. Visual learning and recognition of 3-d objects from appearance. *International Journal of Computer Vision*, 14:5–24, 1995.

[20] R. A. Redner and H. F. Walker. Mixture densities, maximum likelihood and the em algorithm. *SIAM Review*, 26(2):195–239, 1984.

[21] Amnon Shashua. Algebraic functions for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):779–789, Aug. 1995.

[22] S. Smith and J. Brady. Susan: A new approach to low level image processing. *DRA technical report TR95SMS1, Dept. of Engineering Science, Oxford University*, 1995.

[23] R. F. Sproull. Refinements to nearest-neighbor searching in k-dimensional trees. *Algorithmica*, 6:579–589, 1991.

[24] P. Suetens, P. Fua, and A. Hanson. Computational strategies for object recognition. *Computing Surveys*, 24(1):5–61, 1992.

[25] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[26] Shimon Ullman and Ronen Basri. Recognition by linear combinations of models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):992–1005, Oct. 1991.

[27] Zheng Rong Yang and Mark Zwolinski. Mutual information theory for adaptive mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(4):396–403, Apr. 2001.