Using Algebraic Functions of Views for Indexing-Based Object Recognition

George Bebis[1], Michael Georgiopoulos[2], Mubarak Shah[3], and Niels da Vitoria Lobo[3]

Department of Computer Science, University of Nevada, Reno, NV 89557[1]
Department of Electrical & Computer Engineering, University of Central Florida, Orlando, FL 32816[2]
Department of Computer Science, University of Central Florida, Orlando, FL 32816[3]

Abstract

Current indexing-based approaches build the hash table using either a large number of reference views or 3D models. In this paper, we propose building the hash table using algebraic functions of views. During preprocessing, we consider groups of model points and we represent all the views (i.e., images) that they can produce in a hash table. These views are computed using algebraic functions of a small number of reference views which contain the group. Fundamental to this procedure is a methodology based on Singular Value Decomposition and Interval Arithmetic for estimating the ranges of values that the parameters of algebraic functions can assume. During recognition, scene groups are used to retrieve from the hash table the model groups that might have produced them. Using algebraic functions of views for indexing-based recognition offers a number of advantages. First of all, the hash table can be built easier, without requiring 3D models or a large number of reference views. Second, recognition does not rely on the similarities between new and reference views. Third, verification becomes simpler. Finally, the approach is more general and extendible.

1. Introduction

Recognizing 3D objects from 2D images is one of the most challenging problems in computer vision. The indexing-based approach to object recognition has been given considerable attention lately [1]-[7]. It is based on the idea of using *a-priori* stored information about the models in order to quickly eliminate non-compatible model-scene feature matches during recognition. Usually there are two steps: *preprocessing* and *recognition*. During preprocessing, groups of model features are considered and a description for each one of them is computed. These descriptions are then used to construct an index to a hash table where information about the group is stored. During recognition, groups of points are chosen from the scene and their descriptions are used to access the hash table.

Ideally, one would like the index computed from a group of model features to remain invariant under viewpoint changes. Although affine [1] and projective [3] invariants have been proposed in the case of planar objects, no general-case invariants exist for single views of general 3D objects [6]. As a result, model-based invariants [4] and probabilistic indexing [5] approaches have been proposed. Another approach is to model 3D objects with a large number of views [2]. Then, indexing based on affine invariants can be used. Alternatively, a 3D model per object is assumed to be available. Then, the viewing sphere is sampled and a description about the views (i.e., projections) that groups of model features can produce from each viewpoint is stored in a hash table [6]. In the case of orthographic projection and 3D linear transformations, the hash table can actually be built using analytical formulas, without having to sample the viewing sphere (i.e., the images of groups of 3D points are represented as a pair of 1D lines in two high-dimensional spaces) [7].

In this paper, a new indexing-based object recognition approach is proposed using algebraic functions of views [8]-[12]. Algebraic functions of views are functions which express a relationship among a number of views of the same object in terms of their image coordinates alone. The key idea in using algebraic functions of views for indexing is that they allow us to compute all possible views that a group of model points can produce using only a small number of views which contain the group. We will be referring to the space of views that a group of points can produce as the *space of transformed views* of the group. During indexing, the space of transformed views is sampled and the sampled views are represented in a hash table. During recognition, scene groups are used to retrieve from the hash table the model groups that might have produced them. To sample the space of transformed views, we first estimate the ranges of values that the parameters of algebraic functions can assume. This is performed using a methodology based on Singular Value Decomposition (SVD) [13] and Interval Arithmetic (IA) [14]. Then, we sample the space of transformed views by sampling the space of parameters.

Building the hash table using algebraic functions of views is more practical since 3D models are not required. In [7], for example, the lines which represent the images of the model groups can be found easily only if the 3D structure of the object is available. In addition, the proposed approach is different from [2] which requires a large number of reference views to ensure that new views are similar to some of the reference views. This is not required by the proposed approach; all that is required for the new views is to contain common groups of points with a small number of reference views. Another advantage for using algebraic functions is that verification becomes simpler: candidate models can be back-projected onto the scene by combining a small number of their reference views. Finally, the fact that algebraic functions of views exist over a wide range of transformations and projections [8]-[12], makes the proposed methodology more general and extendible.

2. Background on algebraic functions of views

Algebraic functions of views were first introduced, in the case of scaled orthographic projection, by Ullman and Basri [8][9]. Let us consider three reference views of the same object V_1, V_2, and V_3, which have been obtained by applying different rigid transformations, and three points $p' = (x', y')$, $p'' = (x'', y'')$, and $p''' = (x''', y''')$, one from each view, which are in correspondence. If V is a new view of the same object, obtained by applying a different rigid transformation, and $p = (x, y)$ is a point which is in correspondence with p', p'', and p''', then the coordinates of p can be expressed in terms of the coordinates of p', p'', and p''' as follows:

$$x = a_1 x' + a_2 x'' + a_3 x''' + a_4 \quad (1)$$

$$y = b_1 y' + b_2 y'' + b_3 y''' + b_4 \quad (2)$$

where the parameters a_j, b_j, $j = 1, \ldots, 4$, are the same for all the points which are in correspondence across the four views. The parameters actually satisfy certain constraints [8]. The above result can be simplified if we generalize the orthographic projection by removing the orthonormality constraint associated with the rotation matrix. In this case, the object undergoes a 3D linear transformation in space and only two reference views are required. The corresponding algebraic functions are shown below:

$$x = a_1 x' + a_2 y' + a_3 x'' + a_4 \quad (3)$$

$$y = b_1 x' + b_2 y' + b_3 x'' + b_4 \quad (4)$$

It should be noted that (3) and (4) can be rewritten using the y-coordinates of the second reference view instead. The extension of algebraic functions of views in other cases can be found in [10]-[12].

3. A framework for indexing

Algebraic functions of views can be used to predict the image coordinates of points in a novel view by appropriately combining the image coordinates of the same points across a number of reference views. This idea can be used for recognizing unknown views of an object. There are, however, two stumbling blocks with this approach: first, we need to find which points from the reference views correspond to which points from the unknown view and second, we need to find the set of values for the parameters of the algebraic functions (i.e., a_j's, b_j's). Both problems can be very computationally intensive. Here, we propose the coupling of algebraic functions of views with indexing. The idea is to use algebraic functions of views to predict all images that groups of model points can produce. These predictions are then represented in a hash table along with information about point correspondences and parameter values. During recognition, groups of points are chosen from the scene and the hash table is accessed to retrieve the model groups that might have produced them as well as information about point correspondences and parameter values.

The first step in this approach is to compute the ranges of values that the parameters of algebraic functions can assume. Then, these ranges are sampled and for each sampled set of parameter values, a new view is obtained by combining a small number of views using algebraic functions. The image coordinates of the groups in the new views are then used to generate an index to a hash table where information about the model, the reference views, the group, and the sampled set of parameter values are stored. During recognition, groups of points are chosen from the scene and their image coordinates are used to generate an index. The entries stored in the indexed location indicate the model, reference views, model group, and parameter values that might have produced the scene group. Verification follows to reject/accept candidate matches. Figure 1 illustrates these steps.

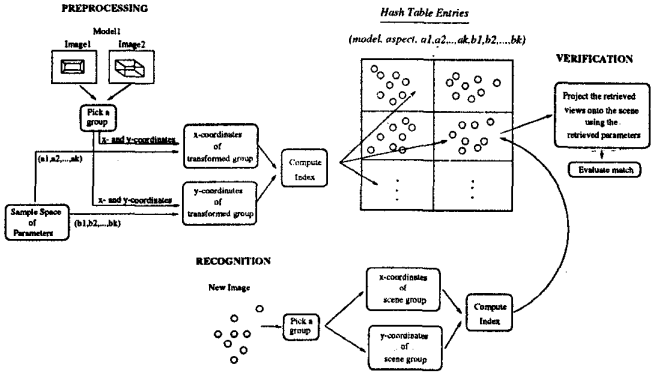

Figure 1. The steps involved during indexing.

4. Estimating the parameters' ranges of values

Let us assume that we have M models and that each model is represented by a number of aspects A_m, m=1,2,...,M. In the case of convex 3D objects, six aspects should be enough, while in the case of general 3D objects, more aspects are necessary to represent the object from different viewing directions. Each aspect is represented by V different views (reference views). Here, V=2 since we assume orthographic projection and 3D linear transformations. For each aspect, we assume a number of "interest" points $N_{m(a)}$, $m = 1, 2, \ldots, M$, $a = 1, 2, \ldots, A_m$ (i.e., corners or junctions), which are common in all the views associated with the aspect and in correspondence.

Under the assumption of orthographic projection, the following system of equations should be satisfied (see (3) and (4)):

$$\begin{bmatrix} x'_1 & y'_1 & x''_1 & 1 \\ x'_2 & y'_2 & x''_2 & 1 \\ \cdots & \cdots & \cdots & \cdots \\ x'_{N_{m(a)}} & y'_{N_{m(a)}} & x''_{N_{m(a)}} & 1 \end{bmatrix} \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \\ a_4 & b_4 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \cdots & \cdots \\ x_{N_{m(a)}} & y_{N_{m(a)}} \end{bmatrix} \quad (5)$$

Splitting the above system in two subsystems we have:

$$Pc_1 = p_x \quad (6)$$

$$Pc_2 = p_y \quad (7)$$

where P is the matrix formed by the x- and y-coordinates of the reference views (plus a column of 1's), c_1 and c_2 are vectors corresponding to the a_j and b_j parameters and p_x, p_y are vectors corresponding to the x- and y-coordinates of the new view. Since both (6) and (7) are overdetermined, they can be solved using a least-squares approach such as SVD [13]. Using SVD, $P = U_P W_P V_P^T$ where both U_P and V_P are

orthonormal matrices, while W_P is a diagonal matrix whose elements w_{ii}^P are called the singular values of P. The solution of the above two systems is $c_1 = P^+ p_x$ and $c_2 = P^+ p_y$ where P^+ is the pseudoinverse of P defined as $P^+ = V_P W_P^+ U_P^T$ (W_P^+ is also a diagonal matrix with elements $1/w_{ii}^P$ if w_{ii}^P greater than zero and zero otherwise). In specific, the solutions are given by the following equations [13]:

$$c_1 = \sum_{i=1}^{4} \left(\frac{u_i^P p_x}{w_{ii}^P}\right) v_i^P \tag{8}$$

$$c_2 = \sum_{i=1}^{4} \left(\frac{u_i^P p_y}{w_{ii}^P}\right) v_i^P \tag{9}$$

where u_i^P denotes the i-th column of matrix U_P, v_i^P denotes the i-th column of matrix V_P.

To determine the range of values for c_1 and c_2, first we assume that the views to be recognized have been scaled such that their x, y-coordinates lie in [0,1]. Then, we compute all possible solutions of (6) and (7) assuming that p_x, p_y belong to [0,1]: The idea is to use Interval arithmetic (IA) [14]. In IA, each variable is represented as an interval of possible values. Given two interval variables $t = [t_1, t_2]$ and $r = [r_1, r_2]$, then the sum and the product of these two interval variables is defined as follows [14]:

$$t + r = [t_1 + r_1, t_2 + r_2] \tag{10}$$

$$t * r = [min(t_1 r_1, t_1 r_2, t_2 r_1, t_2 r_2), max(t_1 r_1, t_1 r_2, t_2 r_1, t_2 r_2)] \tag{11}$$

In interval notation, we want to solve the systems $Pc_1 = p_x^I$ and $Pc_2 = p_y^I$, where the superscript I denotes an interval vector, given that $p_x^I = p_y^I = [0,1]$. The solutions c_1^I and c_2^I should be understood to mean $c_1^I = [c_1: Pc_1 = p_x, p_x \in p_x^I]$ and $c_2^I = [c_2: Pc_2 = p_y, p_y \in p_y^I]$. As an example, let us consider the 3D objects shown in Figure 2. Table 1 shows the range of values computed for c_1.

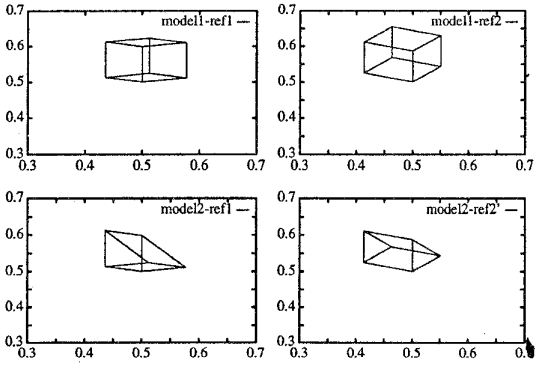

Figure 2. Some test 3D objects.

Table 1. The computed ranges for the original views.

	Ranges of values			
	range of a1	range of a2	range of a3	range of a4
model1	[-25.321 25.321]	[-10.154 10.154]	[-23.173 23.173]	[-5.943 6.943]
model2	[-27.771 27.771]	[-10.154 10.154]	[-24.328 24.328]	[-8.496 9.496]

There are two issues to be emphasized at this point. First, since both (6) and (7) involve the same matrix P and p_x, p_y are equal, then $c_1^I = c_2^I$. Second, not every solution in c_1^I and c_2^I satisfies the interval system of equations. In other words, not every solution in c_1^I and c_2^I corresponds to p_x and p_y that belong to p_x^I and p_y^I [15]. Thus, $p_x^I \subseteq Pc_1^I$ and $p_y^I \subseteq Pc_2^I$. We call these "invalid solutions". Clearly, views corresponding to invalid solutions can be easily detected and rejected by testing whether the image coordinates of a transformed group lie inside the unit square.

5. Preconditioning the reference views

As Table 1 illustrates, the width of the range of values varies from parameter to parameter. Wide ranges are not desirable because more sets of values must be considered. In this section, we present a methodology to "precondition" the original reference views. By "preconditioning" we imply a transformation that will transform the original reference views to new reference views, yielding very narrow ranges. We define the "condition" of a reference view as the ratio of the maximum to minimum singular values of matrix P in (6) or (7) (this ratio can be regarded as the condition number of P [16]). By performing a number of experiments, we have found that large condition numbers yield wide ranges of values for c_1 and c_2 [17]. This should not be considered coincidental since it is well known that the relative error in the solution of a linear system of equations, like (6) or (7), depends on the condition number of the coefficients' matrix (P) [16]. Using (5), we can write the preconditioning transformation as follows:

$$\begin{bmatrix} x_1' & y_1' & x_1'' & 1 \\ x_2' & y_2' & x_2'' & 1 \\ \cdots & \cdots & \cdots & \cdots \\ x_{N_{m(a)}}' & y_{N_{m(a)}}' & x_{N_{m(a)}}'' & 1 \end{bmatrix} \begin{bmatrix} a_1 & b_1 & a_5 & 0 \\ a_2 & b_2 & a_6 & 0 \\ a_3 & b_3 & a_7 & 0 \\ a_4 & b_4 & a_8 & 1 \end{bmatrix} = \begin{bmatrix} x_1'^n & y_1'^n & x_1''^n \\ x_2'^n & y_2'^n & x_2''^n \\ \cdots & \cdots & \cdots \\ x_{N_{m(a)}}'^n & y_{N_{m(a)}}'^n & x_{N_{m(a)}}''^n \end{bmatrix}$$

$$or \quad PC = P^n \tag{12}$$

where C is the desired transformation matrix, P is the matrix corresponding to the old reference views, and P^n is the matrix corresponding to the new reference views. Let us consider the singular value decomposition of P, C and P^n: $P = U_P W_P V_P^T$, $C = U_C W_C V_C^T$, and $P^n = U_{Pn} W_{Pn} V_{Pn}^T$. Substituting these expressions in (12) we have:

$$(U_P W_P V_P^T)(U_C W_C V_C^T) = (U_{Pn} W_{Pn} V_{Pn}^T) \tag{13}$$

Since there is some freedom in choosing the elements of C, (13) can be simplified if we choose $U_C = V_P$. Then, (13) can be written as $(U_P W_P W_C V_C^T) = (U_{Pn} W_{Pn} V_{Pn}^T)$, since $V_P^T V_P = I$. This implies that $W_{Pn} = W_P W_C$. The key is to choose the singular values of C in a way such that the singular values of P^n are all equal. Obviously, W_C should be chosen as follows:

$$W_C = \lambda W_P^{-1} \tag{14}$$

where λ is a positive constant. As a result, $W_{Pn} = \lambda I$. Since the last column of C is equal to $[0\ 0\ 0\ 1]^T$, the following equation should be satisfied: $V_P W_C (v_4^C)^T = [0\ 0\ 0\ 1]^T$. We proceed by splitting the above problem into two subproblems:

$$W_C (v_4^C)^T = z, \tag{15}$$

$$V_P z = [0\ 0\ 0\ 1]^T \tag{16}$$

V_P is known from the SVD analysis of P. The procedure is to solve for z first (Eq. (16)) and then to solve for $(v_4^C)^T$ (Eq. (15)). In solving (15), we need to consider an additional

constraint: the magnitude of the solution vector $(v_4^C)^T$ must be equal to one (a direct consequence of the orthonormality of V_C). Assuming that the elements of W_C are w_{ii}^C, $i = 1,2,3,4$, the solutions of (15) must satisfy the following condition:

$$(\frac{z_1}{w_{11}^C})^2 + (\frac{z_2}{w_{22}^C})^2 + (\frac{z_4}{w_{43}^C})^2 + (\frac{z_4}{w_{44}^C})^2 = 1 \quad or$$

$$(z_1 \prod_{i\neq 1} w_{ii}^C)^2 + (z_2 \prod_{i\neq 2} w_{ii}^C)^2 + (z_3 \prod_{i\neq 3} w_{ii}^C)^2 + (z_4 \prod_{i\neq 4} w_{ii}^C)^2 = (\prod_{i=1}^{4} w_{ii}^C)^2$$

From (14), $w_{ii}^C = \lambda / w_{ii}^P$, $i = 1,2,3,4$. Substituting w_{ii}^C in the above equation and solving for λ we have:

$$\lambda = (\prod_{i=1}^{4} w_{ii}^P) \sqrt{(z_1 / \prod_{i\neq 1} w_{ii}^P)^2 + (z_2 / \prod_{i\neq 2} w_{ii}^P)^2 + (z_3 / \prod_{i\neq 3} w_{ii}^P)^2 + (z_4 / \prod_{i\neq 4} w_{ii}^P)^2}$$

where only the positive λ value has been considered since the sign of λ determines the sign of the singular values. Next, the rest elements of V_C need to be determined. The details can be found in [17].

Figure 3 shows the preconditioned views corresponding to the views shown in Figure 2 (only one preconditioned reference view is shown since only the x-coordinates of the second reference view were used in preconditioning). Table 2 shows the computed ranges of values for the parameters in this case.

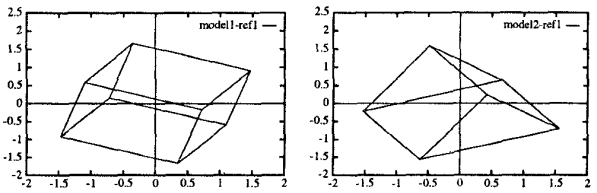

Figure 3. The preconditioned views.

Table 2. The computed ranges for the preconditioned views.

	Ranges of values			
	range of a1	range of a2	range of a3	range of a4
model1	[-0.454 0.454]	[-0.417 0.417]	[-0.392 0.392]	[0.0 1.0]
model2	[-0.439 0.439]	[-0.413 0.413]	[-0.423 0.423]	[0.0 1.0]

6. Decoupling the image coordinates

As discussed in section 4, both a_j and b_j assume values from the same ranges. Taking also into consideration that the same basis vector is involved in the computation of both x- and y-coordinates (i.e., (x', y', x'')), it turns out that the transformation which generates the x-coordinates is exactly the same to the transformation which generates the y-coordinates. Since it is not necessary to represent the same transformation twice over the hash table, only one of the two coordinates (the x-coordinates here) are used during indexing. This simplification offers great computational and memory savings, however, it makes recognition slightly more complicated. The hash table needs to be accessed twice now: first, the x-coordinates of the scene group are used to give rise to hypotheses which predict a_js and second, the y-coordinates of the scene group are used to give rise to hypotheses which predict b_js. Then, the intersection of the hypotheses needs to be considered. We further discuss this in section 9.

7. Predicting the parameters

Given a scene group, the goal of recognition is to predict the model group and the parameters that have produced the scene group. As discussed in the previous section, the prediction of the parameters is performed in two steps. However, it is important to understand that there will be some error in the prediction of the parameters. This is because the hash table was built using a finite number of images per model group (obtained by sampling the space of parameters). As a result, if a scene group is not very similar to one of the transformed groups used during preprocessing, then the predicted parameters will not be very accurate.

In a recent work [18], we studied the problem of learning to predict the parameters of the affine transformation between known and unknown views of a planar object. A neural network was used to learn the mapping. This work has been extended in the case of 3D objects, assuming orthographic projection and 3D linear transformations [19]. First, a number of training views is generated using algebraic functions of a small number of reference views. Then, these views are used to train a neural network to predict the parameters of the algebraic functions used to generate the training views from the reference views. Motivated from these ideas, we have assigned a different neural network to each model group (*group specific neural networks*). Then, when an entry is made to the hash table during preprocessing, instead of storing the parameters of the algebraic functions we store a pointer to the neural neural network associated with the model group. It should be noted that the neural network approach has lower space and time requirements compared to other least-square approaches [17][18].

8. Group size, condition, and ordering

An important issue when we consider groups of points is how to choose the number of points G in the group. Obviously, in order for the groups to be useful for matching, G must be chosen in a way such that every scene group may have been produced only by one model group. In the case of orthographic projection and 3D linear transformations, the algebraic functions of views involve eight parameters. This means that we need to match at least four scene points to four model points in order to determine the parameters. Thus, the minimum group size which provides discrimination is five.

Considering all possible model groups of a given size during preprocessing is not practical since this would require a lot of space. Here, we consider only *well-conditioned* model groups. The definition of the condition of a model group is similar to the definition of the condition of a model view: it is the condition of the matrix formed by the x- and y-coordinates of the group, across the reference views, plus a column of 1's (see Eq. (5)). Obviously, if we assume noise in the location of the points of a group (right hand-side of (5)), then the error in the solution of (5) will proportional to the condition of the group (matrix P) [16]. Thus, although a scene group might have been matched correctly to a model group, verification might not be able to verify this if the parameters of the algebraic functions have not been recovered accurately. To avoid such hypothetical matches from the beginning, unstable model groups are dis-

qualified during preprocessing. This is performed by applying a simple thresholding on the condition of the groups. We must ensure, however, that most model points are represented in the groups chosen and that the same model point does not appear in every model group to allow tolerance to occlusions.

Another important issue is the order of the points in the group. If we do not make any assumptions about the order, either all possible orderings must be considered during preprocessing or all possible orderings must be considered during recognition. Since the second approach will increase recognition time, the first approach is considered only. To avoid considering all possible orderings during preprocessing, we apply a canonical ordering to the points of the model groups. The canonical ordering procedure utilized here is very simple: we order the points by sorting the coordinates (x or y) of the group in an increasing order. During recognition, the same canonical ordering is applied to the scene groups. Information about the ordering of the model group is stored in the hash table during preprocessing.

9. Evaluating hypotheses

Considering only well-conditioned groups during preprocessing restricts ourselves to a small set of model groups. As a result, many invalid matches are expected to be established during recognition. To reject as many invalid matches as possible without having to apply the expensive verification step first, each hypothetical match is evaluated using a number of easy to test conditions. As mentioned earlier, each hypothesis is formed by combining entries retrieved by the x-coordinates of a scene group with entries retrieved by the y-coordinates of the scene group. However, before a hypothesis is verified, it has to satisfy a number of conditions. These conditions ensure that (i) both x- and y-coordinates predict the same model, (ii) the same aspect, (iii) the same model group, (v) the parameters predicted by the neural network belong to the ranges computed during preprocessing, and (iv) the predicted model group is well-conditioned.

10. Recognizing 3D objects

In this section, we demonstrate the proposed approach through a number of experiments. The group size chosen in these experiments is G=5. A 5-dimensional hash table of size 10 x 10 x 10 x 10 x 10 was utilized. First, we performed a number of experiments using the artificial objects shown in Figure 2. For each object, we generated a number of random views and we added some random noise in the location of the model points to simulate sensor noise. In all cases, recognition was successful and the parameters of the algebraic functions were predicted accurately. The number of hypotheses verified in four different cases are shown in Table 3. To demonstrate the significance of the tests discussed in section 9, Table 3 shows the number of hypotheses verified with (second column) and without testing (first column) these conditions.

Next, we performed a number of experiments using the real 3D objects shown in Figure 4. We have considered only one aspect per object in these experiments. The views used to model these aspects are shown in Figure 4 ((a)-(f)). A corner detector [20] was used to extract a number of interest points. Then, we manually selected the most important interest points, which were also common in both views, and we established their correspondences. Figure 4 ((g)-(l)) shows the points chosen (the lines have been drawn just for visualization).

Table 3. Verified hypotheses.

Hypotheses	
without using (i)-(iv)	using (i)-(iv)
12888	45
50907	65
107603	186
20257	61

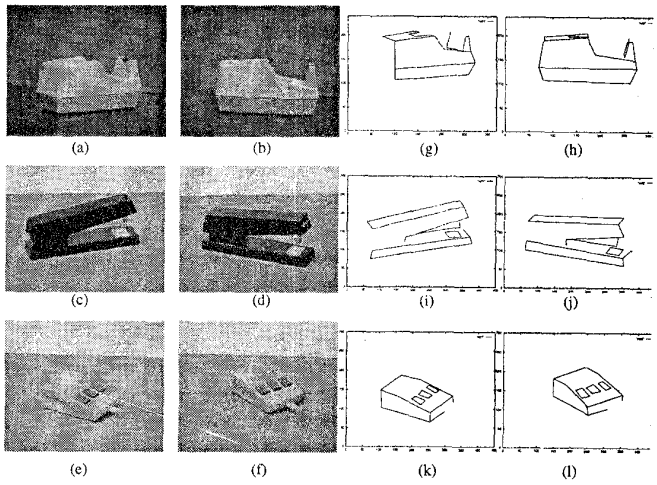

Figure 4. The real objects.

Table 4. The computed ranges for the real objects.

	Ranges of values			
	range of a1	range of a2	range of a3	range of a4
model1	[-0.41933 0.41933]	[-0.36234 0.36234]	[-0.42926 0.42926]	[0.0 1.0]
model2	[-0.44177 0.44177]	[-0.45138 0.45138]	[-0.43368 0.43368]	[0.0 1.0]
model3	[-0.42321 0.42321]	[-0.41114 0.41114]	[-0.37975 0.37975]	[0.0 1.0]

Then, the reference views were preconditioned, the range of values of the parameters of the algebraic functions were estimated, and the hash table was built. Table 4 shows the ranges computed for each object. Some of the scenes used in our recognition experiments are shown in Figure 5. The interest points were detected using the same corner detector. Since we have not tried to optimize the selection of promising scene groups during recognition, for example by using some kind of grouping, we manually removed interest points that were not corresponding to strong corners to limit the number of groups to be examined during recognition. Then, groups of scene points were formed randomly, using the remaining points, and hypotheses were formed. The recognition results are shown in Figure 5 (recognized objects have been superimposed on the scene). Table 6 shows the actual and predicted parameters in each case.

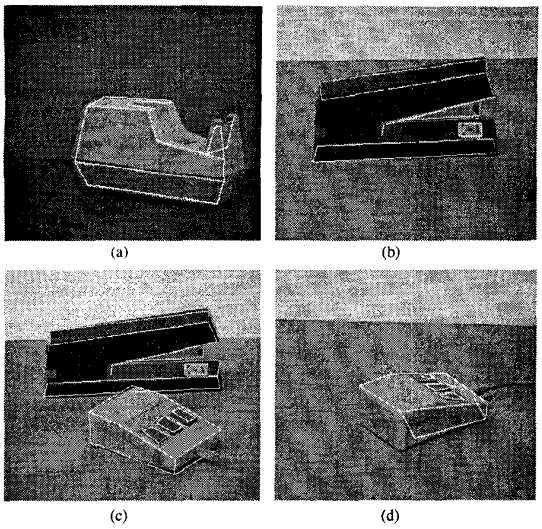

Figure 5. Recognition results.

Table 6. Actual and predicted parameters.

	Parameters	
	Actual parameters (**Figure 5(a)**)	Predicted parameters (**Figure 5(a)**)
a_1,a_2,a_3,a_4	0.03704 0.19696 0.04488 0.63449	0.03700 0.19692 0.04485 0.63446
b_1,b_2,b_3,b_4	-0.12358 0.05752 0.01046 0.53638	-0.12353 0.05757 0.01051 0.53644
	Actual parameters (**Figure 5(b)**)	Predicted parameters (**Figure 5(b)**)
a_1,a_2,a_3,a_4	-0.041474 0.22793 0.02362 0.57797	-0.04145 0.22798 0.02365 0.57802
b_1,b_2,b_3,b_4	0.123646 -0.05775 -0.00653 0.50611	0.12360 -0.05781 -0.00658 0.50606
	Actual parameters (**Figure 5(c)**)	Predicted parameters (**Figure 5(c)**)
a_1,a_2,a_3,a_4	0.01682 0.13225 -0.00602 0.62491	0.01682 0.13228 -0.00601 0.62492
b_1,b_2,b_3,b_4	-0.08168 0.03142 0.000150 0.68023	-0.08168 0.03146 0.00018 0.68026
	Actual parameters (**Figure 5(d)**)	Predicted parameters (**Figure 5(d)**)
a_1,a_2,a_3,a_4	-0.08481 0.06358 -0.03732 0.61571	-0.08480 0.06357 -0.03733 0.61572
b_1,b_2,b_3,b_4	-0.05666 -0.04054 0.01670 0.52448	-0.05660 -0.04049 0.01673 0.52454

11. Conclusions

In this paper, we introduced a new indexing-based object recognition approach based on algebraic functions of views. The proposed approach has the advantage that it requires only a small number of views per object to build the hash table. Thus, it is more practical than other approaches which either require too many views or the 3D structure of the object. For future research, we are planning to extend the proposed approach in the case of perspective projection. We are also exploring ways to build the hash table using analytical formulas, like in [7], without having to sample the parameter space.

References

[1] Y. Lamdan, J. Schwartz, and H. Wolfson, "Affine invariant model-based object recognition", *IEEE Trans. on Robotics and Automation*, vol. 6, no. 5, pp. 578-589, October 1990.

[2] 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*, pp. 1407-1413, 1988.

[3] C. Rothwell, A. Zisserman, D. Forsyth, and J. Mundy, "Planar object recognition using projective shape representation", *International Journal of Computer Vision*, vol. 16, pp. 57-99, 1995.

[4] D. Weinshall, "Model-based invariants for 3D vision", *International Journal of Computer Vision*, vol. 10, no. 1, pp. 27-42, 1993.

[5] C. Olsen, "Probabilistic indexing for object recognition", *IEEE Pattern Analysis and Machine Intelligence*, vol. 17, no. 5, pp. 518-522, 1995.

[6] D. Clemens and D. Jacobs, "Space and time bounds on indexing 3D models from 2D images", *IEEE Pattern Analysis and Machine Intelligence*, vol. 13, no. 10, pp. 1007-1017, 1991.

[7] D. Jacobs, *Recognizing 3D objects using 2D images*, International Journal of Computer Vision, vol. 21, no 1/2, 1997.

[8] S. Ullman and R. Basri, "Recognition by linear combination of models", *IEEE Pattern Analysis and Machine Intelligence*, vol. 13, no. 10, pp. 992-1006, October 1991.

[9] R. Basri, "Recognition by combinations of model views: alignment and invariance", in *Applications of Invariance in Computer Vision*, J. Mundy, A. Zisserman, and D. Forsyth (Eds.), pp. 435-450, 1994.

[10] A. Shashua, "Algebraic Functions for Recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 779-789, 1995.

[11] O. Faugeras and L. Robert, "What can two images tell us about a third one ?", *Third European Conference on Computer Vision (ECCV'94)*, pp. 485-492, 1994.

[12] R. Basri, "Paraperspective ≡ Affine", *International Journal of Computer Vision*, vol. 19, no. 2, pp. 169-179, 1996.

[13] W. Press et. al *Numerical recipes in C: the art of scientific programming*, Cambridge University Press, 1990.

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

[15] E. Hansen and R. Smith, "Interval arithmetic in matrix computations: Part II", *SIAM Journal of Numerical Analysis*, vol. 4, no. 1, 1967.

[16] G. Forsythe, M. Malcolm, and C. Moler, "*Computer methods for mathematical computations*, (chapter 9), Prentice-Hall, 1977.

[17] G. Bebis, M. Georgiopoulos, M. Shah, and N. La Vitoria Lobo, "Indexing based on algebraic functions of views", submitted for publication.

[18] G. Bebis, M. Georgiopoulos, N. da Vitoria Lobo, and M. Shah, "Learning affine transformations of the plane for model based object recognition", *13th International Conference on Pattern Recognition (ICPR-96)*, vol. IV, pp. 60-64, Vienna, Austria, August 1996.

[19] G. Bebis, M. Georgiopoulos, and S. Bhatia "Learning orthographic transformations for object recognition", *IEEE International Conference on Systems, Man, and Cybernetics*, October 12-15, 1997, Orlando, FL (to appear).

[20] S. Smith and J. Brady, "SUSAN: A new approach to low level image processing", *DRA Technical Report TR95SMS1*, Department of Engineering Science, Oxford University, 1995.

[21] D. Jacobs, "Robust and efficient detection of convex groups", *IEEE Conf. on Computer Vision and Pattern Recognition*, pp.770--771, 1993.