Learning Orthographic Transformations for Object Recognition

George Bebis[†], Michael Georgiopoulos[‡], and Sanjiv Bhatia[†],

[†]Department of Mathematics & Computer Science, University of Missouri-St. Louis, St. Louis, MO 63121
[‡]Department of Electrical & Computer Engineering, University of Central Florida, Orlando, FL 32816

Abstract

In this paper, we consider the problem of learning to predict the correct pose of a 3D object, assuming orthographic projection and 3D linear transformations. A neural network is trained to learn the desired mapping. First, we consider the problem of predicting all possible views that an object can produce. This is performed by representing the object with a small number of reference views and using algebraic functions of views to construct the space of all possible views that the object can produce. Fundamental to this procedure is a methodology based on Singular Value Decomposition and Interval Arithmetic for estimating of the ranges of values that the parameters of algebraic functions can assume. Then, a neural network is trained using a number of views (training views) which are generated by sampling the space of views of the object. During learning, a training view is presented to the inputs of the network which is required to respond at its outputs with the parameters of the algebraic functions used to generate the view from the reference views. Compared to similar approaches in the literature, the proposed approach has the advantage that it does not require the 3D models of the objects or a large number of views, it is extendible to other types of projections, and it is more practical for object recognition.

1. Introduction

In a recent paper [1], we studied the problem of learning to predict the correct pose of a planar object, undergoing 2D affine transformations (i.e., unconstrained viewpoint). The idea was to train a single-layer neural network (SL-NN) with a number of affine transformed views of the object in order for it to learn to predict the parameters of the affine transformation between the training views and a reference view of the object. To demonstrate our approach, we performed a number of experiments using several objects. A separate neural network was assigned to each object and was trained with views of this object only. In this way, each network became specialized in the prediction of the pose from views of a specific object only (*object specific networks*). Our experimental results showed that training was extremely fast and that only a small number of training views was sufficient for the networks to generalize well. By generalization we mean the ability of the networks to predict the correct affine transformation even for views that were never exposed to them during training. We also considered issues related to the discrimination power and noise tolerance of the networks. Our results showed that the discrimination power of the networks was excellent. Their noise tolerance was not very good initially, however, it was dramatically improved by applying a preprocessing to the inputs based on Principal Components Analysis (PCA) [2].

In this paper, we are extending the above work in the case of 3D objects, assuming orthographic projection and 3D linear transformations. This extension is possible using the theory of algebraic functions of views [3][4]. Algebraic functions of views are simply functions which express a relationship among a number of views of the same object in terms of their image coordinates alone. For example, it has been shown that in the case of orthographic projection, the image coordinates of any three views of an object, undergoing 3D linear transformations, satisfy a linear function [3]. This means that novel views of an object can be expressed as a linear combination of two known (reference) views of the object. Our goal here is to train a neural network to predict the pose of 3D objects, in terms of the parameters (coefficients) of the algebraic functions of views.

A separate SL-NN was associated with each object in [1]. This is because a single view is enough to represent planar objects from any viewpoint. In the case of 3D objects, more than one views are needed to represent different aspects of the object. Two views correspond to the same aspect if they capture almost the same 3D features. As discussed above, two views are enough to represent each aspect. A separate neural network has been associated with different aspects of an object (*aspect specific networks*). SL-NNs are used again since the mapping to be approximated is linear. To train the aspect specific networks, a number of training views are generated. This is performed by sampling the space of transformed views associated with the aspect of interest. This space is constructed by combining the reference views associated with the aspect, using algebraic functions of views [3][4]. The values of the parameters used in the combination are obtained by sampling the ranges of values that the parameters can assume. To estimate the ranges, a methodology based on Singular Value Decomposition (SVD) [2] and Interval Arithmetic (IA) [5] is applied. During recognition, an unknown view is presented to the aspect specific networks which predict sets of values for the parameters of the algebraic functions. By combining the reference views associated with an aspect specific network, using the parameter values predicted, a view is predicted which is then compared with the unknown view.

Our work has similarities with [6]. In specific, the problem of approximating a function that maps any perspective view of a 3D object to a standard object view was considered in [6]. This function was approximated by training a Generalized Radial Basis Functions Neural Network (GRBF-NN). The training views were obtained by sampling the viewing sphere, assuming that the 3D model of the object is available. Despite the fact that the two approaches

consider different types of projections, which we discuss later, there are a number of other differences between the two approaches. The first important difference is in the way that the training views are obtained. In [6], the training views can be obtained easily only when the 3D models of the objects are available. Since it is not always possible to assume this, an alternative way must be used to obtain the training views, for example, by taking pictures of the objects from different viewpoints. This approach, however, requires more effort and time: edges must be extracted, interest point must be detected, and point correspondences across the views must be established. On the other hand, our approach requires only a small number of views. Then, the training views can be generated by combining these views, using algebraic functions of views.

The second difference is in the kind of outputs that the neural networks have been trained to produce. In [6], the GRBF-NN predicts the coordinates of a standard view of the object. In a similar approach [3], a linear operator was built to distinguish between views of a specific object and views of other objects, assuming orthographic projection. This was done by mapping every view of the object to a vector which uniquely identifies the object. In our approach, however, the SL-NN predicts the values of the parameters of the algebraic functions of views. Although the above two approaches are mostly biological motivated, it is the practical value of the neural network approach which is of interest to us here. In particular, our objective is to benefit approaches which operate under the hypothesize and verify paradigm [7][8]. In this context, objects are hypothesized and verified during recognition by back-projecting them onto the unknown scene. In the case of recognition using algebraic functions of views [4][9], back-projection simply implies the combination of the reference views of the candidate object. Thus, a candidate set of parameter values must be computed for every hypothesis. We show that the accuracy of the neural network approach in predicting the correct parameters is as good as that of a traditional least-square scheme (SVD), however, the neural network approach has less computational requirements.

To simplify training, we show that it is not necessary to consider both the x- and y-coordinates of the views during training (views are represented here as a collection of points given by their location in the image). In fact, training the networks using only one of the two is enough. This simplification has many benefits (smaller networks, faster training) and adds only a minor cost during pose prediction (pose must now be predicted in two steps). The current version of our method deals with orthographic projection only. However, orthographic approximates perspective quite well when the camera is not very close to the object [10]. Since orthographic projection is linear, linear networks (SL-NN) are required to learn the mapping as opposed to nonlinear networks (GRBF-NN) used in the case of perspective [6]. On the other hand, extending the current approach to other types of projections is possible due to the fact that algebraic functions of views have been shown to exist in the case of paraperspective [12] as well perspective projection [4].

The organization of the paper is as follows: Section 2 presents a brief overview of the theory of algebraic functions of views. The procedure for estimating the range of values that the parameters of the algebraic functions can assume is presented in section 3. In Section 4, we describe the methodology for obtaining the training views and the procedure for training the aspect specific neural networks. Our experimental results are given in Section 5. Finally, section 6 contains our conclusions.

2. Background on algebraic functions of views

Algebraic functions of views were first introduced, in the case of scaled orthographic projection (weak perspective), by Ullman and Basri [3]. They showed that if we let an object to undergo 3D rigid transformations, namely, rotations and translations in space, and we assume that the images of the object are obtained by orthographic projection followed by a uniform scaling, then any novel view of the object can be expressed as a linear combination of three other views of the object. In specific, 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 novel 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. It should be mentioned that the parameters follow certain functional restrictions [3]. 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:

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

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

where the parameters a_j, b_j, $j = 1, \ldots, 4$, are the same for all the points which are in correspondence across the three views. It should be noted that not all the information from the second reference view is used but only "half" of it (only the x-coordinates). Of course, (3) and (4) can be rewritten using the y-coordinates of the second reference view instead.

The extension of algebraic functions of views in the case of perspective projection has been carried out by Shashua [4] and by Faugeras and Robert [11]. In particular, it was shown that three perspective views of an object satisfy a trilinear function. Moreover, Shashua [4] has shown that a simpler and more practical pair of algebraic functions exist when the reference views are orthographic. This is useful for realistic object recognition applications. Here, we consider the case of orthographic projection and 3D linear transformations only.

3. Estimating the ranges of the parameters

We start by introducing some terminology and making certain assumptions. We assume that each object m is represented by a number of aspects A_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 which we call, reference views. The number of reference views V will be equal to two here, since we have assumed the case of linear transformations. For each aspect, we assume a number of "interest" points $N_{m(a)}$, $a = 1, 2, \ldots, A_m$ (e.g., corners or junctions) which are common in all the views associated with the aspect. We also assume that the point correspondences across the views have been established.

Under the assumption of orthographic projection, two reference views V_1 and V_2 must be combined in order to obtain a new view V, as Eqs. (3) and (4) illustrate. Given the point correspondences across the three views, the following system of equations must be satisfied:

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

where (x_1', y_1'), $(x_2', y_2'), \cdots (x_{N_{m(a)}}', y_{N_{m(a)}}')$ and (x_1'', y_1''), $(x_2'', y_2''), \cdots (x_{N_{m(a)}}'', y_{N_{m(a)}}'')$ are the coordinates of the points of the reference views V_1 and V_2 respectively, and $(x_1, y_1), (x_2, y_2), \cdots (x_{N_{m(a)}}, y_{N_m}(a))$ are the coordinates of the points of the novel view V. Instead of the x-coordinates of the second reference view V_2, its y-coordinates could have been used. The above system can now be split into two subsystems, one involving the a_j parameters and one involving the b_j parameters. Using matrix notation, they can be written as follows:

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

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

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 of the algebraic functions 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, we can solve them using a least-squares approach such as SVD [2]. Using SVD, we can factorize the matrix P as $P = U_P W_P V_P^T$ where both U_P and V_P are orthonormal matrices, while W_P is a diagonal matrix whose elements w_{ii}^P are always non-negative and 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. Assuming that P has been factorized, its pseudoinverse is $P^+ = V_P W_P^+ U_P^T$ where W_P^+ is also a diagonal matrix with elements $1/w_{ii}^P$ if w_{ii}^P greater than zero (or a very small threshold in practice) and zero otherwise. In specific, the solutions of (6) and (7) are given by the following equations [2]:

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

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

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 and $k = 4$.

To determine the range of values for c_1 and c_2, we assume that the image of the unknown view has been scaled so that its x- and y-coordinates belong to a specific interval. This can be done, for example, by mapping the image of the unknown view to the unit square. In this way, its x- and y-image coordinates are mapped in the interval [0, 1]. To determine the range of values for c_1 and c_2, we need to consider all possible solutions of (6) and (7), assuming that the p_x and p_y belong to [0,1]. To solve this problem, we have used IA [5]. 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 [5]:

$$t + r = [t_1 + r_1, t_2 + r_2]$$

$$t * r = [min(t_1 r_1, t_1 r_2, t_2 r_1, t_2 r_2), max(t_1 r_1, t_1 r_2, t_2 r_1, t_2 r_2)]$$

Applying the interval arithmetic operators to (8) and (9) instead of standard arithmetic operators, we can compute interval solutions for c_1 and c_2 by setting $p_{x'}$=[0,1] and $p_{y'}$=[0,1]. 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. 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 $c_2^I = [c_2 : Pc_2 = p_y,\ p_y \in p_y^I]$. It should be mentioned that since both (8) and (9) involve the same matrix P and p_x, p_y assume values in the same interval, the interval solutions c_1^I and c_2^I will be the same.

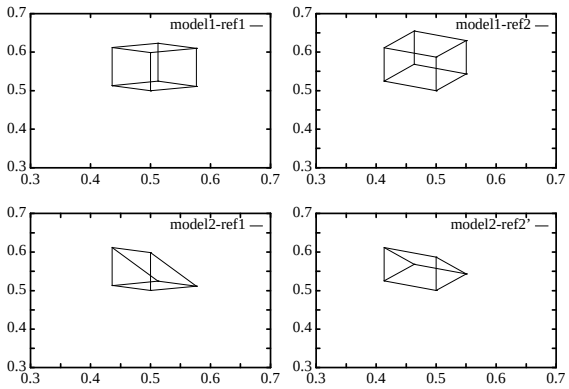


Figure 1. Some test 3D objects.

As an example, let us consider the 3D objects shown in Figure 1. Two different reference views per object are used (we assume that the objects are transparent). Table 1 shows the range of values computed for c_1. It should be mentioned at this point that when interval solutions are computed, 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 [13]. Thus, $p_x^I \subseteq Pc_1^I$ and $p_y^I \subseteq Pc_2^I$. In the context of our approach, if we generate new views by choosing the parameters from the interval solutions obtained, then we might

generate views that do not lie entirely in the unit square. We call these solutions, "invalid solutions". Clearly, predicted views based on invalid solutions should be rejected. By testing whether the coordinates of a predicted view lie inside the unit square, we can easily reject invalid solutions.

Table 1. The computed ranges using the original views.

Ranges of values				
	range of a1	range of a2	range of a3	range of a4
model1	[-25.32 25.32]	[-10.15 10.15]	[-23.17 23.17]	[-5.94 6.94]
model2	[-27.77 27.77]	[-10.15 10.15]	[-24.32 24.32]	[-8.49 9.49]

It can be seen from Table 1 that the width of the ranges varies from parameter to parameter. It can be shown that the width of the ranges depends on the condition of the matrix P (see Eqs. (6) and (7)) and that the original reference views can be "preconditioned" to narrow the ranges of values [14]. By preconditioning we imply a transformation that can transform the original reference views to new reference views, yielding tighter ranges of values. Tighter ranges have the advantage that they yield less invalid views during the generation of the training views. We have applied the preconditioning procedure on the views shown in Figure 1. Table 2 shows the new, tighter, ranges of values.

Table 2. The computed ranges using the preconditioned views.

Ranges of values				
	range of a1	range of a2	range of a3	range of a4
model1	[-0.45 0.45]	[-0.42 0.42]	[-0.39 0.39]	[0.0 1.0]
model2	[-0.44 0.44]	[-0.41 0.41]	[-0.42 0.42]	[0.0 1.0]

4. Learning the mapping

First, the training views must be generated for every object. This is performed by sampling the space of views that the object can produce. This is actually performed by sampling the ranges of values that the parameters of algebraic functions can assume. The sampling procedure is straightforward: first, we pick a sampling step and then we sample the range of values associated with each parameter. Then, we pick a sampled value for each parameter and we form sets of sampled values. Each set of sampled values is then used to generate a new view by combining the reference views using the algebraic functions. If a view is invalid, then we reject it as discussed in section 3. During learning, each training view is presented to the inputs of the network (i.e., the coordinates of the points representing the view) which is required to respond at its outputs with the sampled set of values used for the generation of the view. Figure 2(a) shows the neural network scheme.

As discussed in section 3, both a_j and b_j assume values from the same ranges. Taking also into consideration that the same vector is involved in the computation of the x- and y-coordinates of the training views (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 force the network to learn the same transformation twice, we perform training using only one of the two coordinates (the x-coordinates here). This is shown in Figure 2(b). This simplification has only a minor cost in the prediction of the pose. The parameters of the algebraic function must now be predicted separately: first, we predict a_j's by presenting to the network the x-coordinates of the unknown the view and second, we predict b_j's by presenting to the network the y-coordinates of the unknown view.

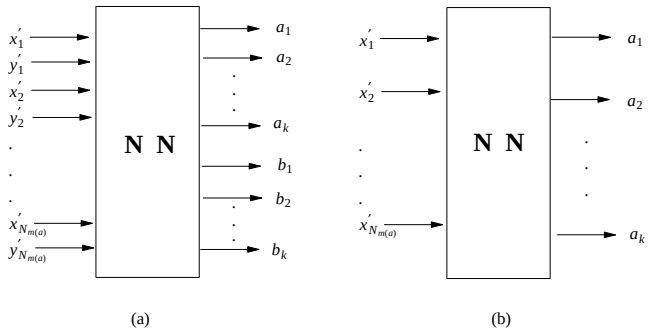


Figure 2. The neural network scheme.

Evaluating neural networks' noise tolerance in [1], we found that it was dramatically improved by preprocessing the inputs using PCA [2]. We have also adopted this idea here. PCA is a multivariate technique which transforms a number of correlated variables to a smaller set of uncorrelated variables. In specific, PCA works as follows: first, we compute the covariance matrix associated with our correlated variables and we find the eigenvalues of this matrix. Then, we sort them and we form a new matrix whose columns consist of the eigenvectors corresponding to the largest eigenvalues. Deciding how many eigenvalues are significant depends on the problem at hand. The matrix formed by the eigenvectors corresponds to the transformation which is applied on the correlated variables to yield the new uncorrelated variables. Here, we think of each view as a vector with components the x-coordinates of the points in the view (the y-coordinates are not used for training) and we apply the procedure as described above.

Despite the fact that PCA improves noise tolerance, it has also two other important benefits: first, it reduces the dimensionality of the input vectors and as a result, smaller size networks are needed. Second, it can guide us in choosing a sufficient number of training views so that the networks learn a good mapping. Performing experiments to evaluate the noise tolerance of the networks using different number of training views, we verified the same results as in [1]. In particular, we found that in cases where the performance of the networks was very poor, the number of non-zero eigenvalues associated with the covariance matrix of the training views was consistently less than four. More training views did not improve the results, as long as the number of non-zero eigenvalues remained less than four. By including more training views which increased the number of non-zero eigenvalues to four, a dramatic improvement was observed. Adding more training views after this point neither improved the results nor increased the number of non-zero eigenvalues. It seems thus that the number of non-zero eigenvalues of the covariance matrix of the training views plays an important role in deciding how many views to select for training. In fact, we think that there is a simple reason that the number of non-zero eigenvalues never exceeded four, given in the form of a theorem in [9]: "the

views of a rigid object are contained in a four-dimensional linear space".

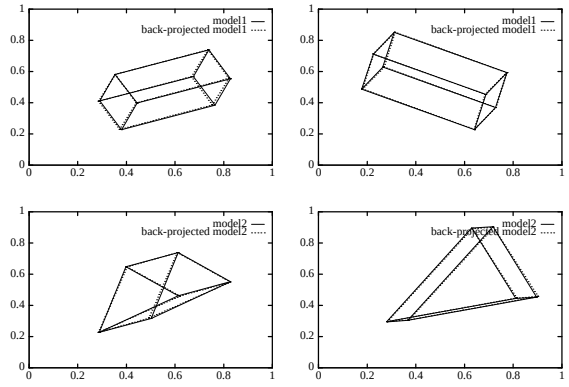


Figure 3. Recognition results.

Table 3. Actual and predicted parameters.

Parameters		
	Actual parameters (*Figure 3(a)*)	Predicted parameters (*Figure 3(a)*)
a_1,a_2,a_3,a_4	-0.08248 -0.15374 0.09463 0.55224	-0.08222 -0.15323 0.09412 0.55223
b_1,b_2,b_3,b_4	-0.05775 0.02241 0.13408 0.48342	-0.05732 0.02222 0.13413 0.48313
	Actual parameters (*Figure 3(b)*)	Predicted parameters (*Figure 3(b)*)
a_1,a_2,a_3,a_4	-0.01866 -0.14224 0.00280 0.46931	-0.01863 -0.14223 0.00280 0.46932
b_1,b_2,b_3,b_4	0.07291 0.04679 0.19096 0.44305	0.07288 0.04675 0.19093 0.44299
	Actual parameters (*Figure 3(c)*)	Predicted parameters (*Figure 3(c)*)
a_1,a_2,a_3,a_4	-0.06960 -0.13060 0.08588 0.53691	-0.07512 -0.13366 0.08039 0.53807
b_1,b_2,b_3,b_4	-0.06665 0.02818 0.16177 0.49026	-0.06666 0.02814 0.16176 0.49025
	Actual parameters (*Figure 3(d)*)	Predicted parameters (*Figure 3(d)*)
a_1,a_2,a_3,a_4	-0.14990 0.08872 0.14231 0.61792	-0.14990 0.08868 0.14231 0.61792
b_1,b_2,b_3,b_4	-0.21708 -0.12707 0.01205 0.55004	-0.21713 -0.12714 0.01200 0.54999

5. Experiments

First we performed a number of experiments using the artificial objects shown in Figure 1. The interest points used were the points corresponding to the corners of each object. For each object, we generated a number of training views and we trained a SL-NN to learn the desired mapping. Back-propagation with momentum was used for the training of the networks [15]. The learning rate used was 0.2 and the momentum term was 0.4. The networks assumed to have converged when the sum of squared errors between the desired and actual outputs was less than 0.0001. To evaluate the quality of the mapping found by the networks, we generated a number of test views per object by combining linearly the reference views of the object, choosing the parameters of the linear combination randomly. To ensure that the x- and y-coordinates of the test views were in [0,1], we chose a random subsquare within the unit square and we mapped the square enclosing the view of the object to the randomly chosen subsquare. We also added some random noise in the location of the points to simulate sensor noise. To find how accurate the predictions of the networks were, we compared the predicted parameters with the actual parameters which were computed using SVD. Also, we back-projected the candidate view onto the test view to evaluate the match visually. Figure 3 shows some examples while Table 3 shows the actual and predicted parameters.

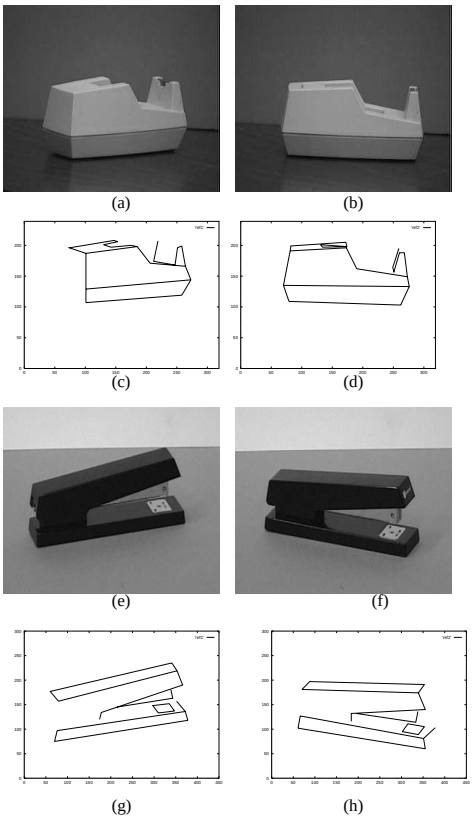


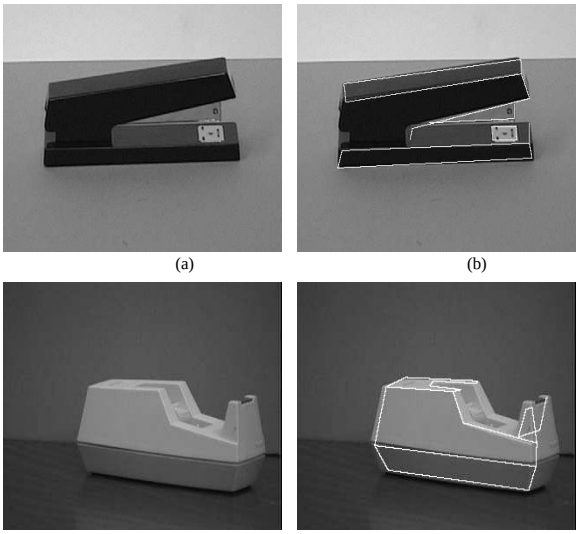
Figure 4. The real objects.

Next, we performed a number of experiments using the real objects shown in Figure 4. A single aspect was considered and the reference views used to represent it are shown in Figures 4(a),(b) and 4(e),(f). To detect a number of interest points, we applied a corner and junction detector [16]. Then we manually picked sets of points which were common in both views. Figures 4(c),(d) and 4(g),(h) show the interest points chosen (the lines connecting the points have been drawn only for visualization purposes). The reference views were preconditioned and the ranges computed are shown in Table 4. Figure 5(a,c) shows some novel views. Note that none of them is exactly the same to any of the reference views. To interest points in the novel views were detected using the same corner detector ([16]). Then, we picked manually the points present in the reference views and we fed them to the corresponding networks. Table 5 shows the actual and predicted parameters. Figure 5(b,d) shows the predicted views, back-projected on novel views. It is important to mention at this point that the order in which the data are presented to the networks is important. This is because the networks are not invariant to shifts in the input sequence. This was also the case in [3] and [6]. Depending on the context of the application, however, the order might be available. For example, we have incorporated the neural network scheme in an indexing-based object recognition system [14]. In this system, groups of points (for occlusion tolerance) are chosen from the unknown view and are used to retrieve hypotheses from a

table. Each hypothesis contains information about a model group as well as information about the order of the points in the group. This information can be used to place the points in the correct order before they are fed to the network.

Table 4. The computed ranges (preconditioned views).

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]



(a) (b) (c) (d)

Figure 5. Recognition results.

Table 5. Actual and predicted parameters (real data).

Parameters		
	Actual parameters (**Figure 5(b)**)	Predicted parameters (**Figure 5(b)**)
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(d)**)	Predicted parameters (**Figure 5(d)**)
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

To compare the computational requirements of the SVD approach with that of the neural network approach, let us assume that SVD decomposition takes place off-line as is the case with the training of the neural networks. Then, assuming that the average number of points per view is N and that the number of parameters is 2k (k=4 here), the neural network approach requires $2kN$ multiplications and $2kN$ additions while SVD requires $2k(N+2k)$ multiplications, $2kN$ divisions, and $2k(N+2k)$ additions (see Eqs. (8) and (9)). Given that these computations must be repeated hundreds of times during recognition, the neural network approach has obviously lower computational requirements.

6. Conclusions

The problem of predicting the pose of a 3D object, assuming orthographic projection and 3D linear transformations was considered in this study. The proposed approach has the advantage that it does not require the 3D models of the objects and it is more practical for object recognition. Extensions to perspective projection are currently being explored.

References

[1] 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.

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

[3] 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.

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

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

[6] T. Poggio and S. Edelman, "A network that learns to recognize three-dimensional objects", *Nature*, vol. 343, January 1990.

[7] 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.

[8] D. Huttenlocher and S. Ullman, "Recognizing solid objects by alignment with an image", *International Journal of Computer Vision*, vol. 5, no. 2, pp. 195-212, 1990.

[9] R. Basri, "Recognition by combination of model views: alignment and invariance", in *Applications of Invariance in Computer Vision*, Lecture Notes in Computer Science, vol. 825, pp. 435-450, 1994.

[10] D. Thompson and J. Mundy, "Three dimensional model matching from an unconstrained viewpoint", *Proceedings of the IEEE Conference on Robotics and Automation*, pp. 208-220, 1987.

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

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

[15] J. Hertz, A. Krogh, and R. Palmer, *Introduction to the theory of neural computation*, Addison Wesley, 1991.

[16] 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.