Proceedings of the 5th WSEAS Int. Conf. on Signal Processing, Computational Geometry & Artificial Vision, Malta, September 15-17, 2005 (pp155-162)

3D Object Prototype-Based Recognition from 2D Images Using Features Net

RAQUEL CÉSAR, AGOSTINHO ROSA
Laseeb, ISR
Instituto Superior Técnico, Universidade Técnica de Lisboa
Av Rovisco Pais, 1, TN 6.21, 1049-001 Lisboa
PORTUGAL

Abstract: Object recognition is at the top of a visual task hierarchy. In its general form, this is a very difficult computational problem, which will probably play an important role in the eventual building of intelligent machines. A large number of psychological and neurophysiologic studies support the idea that humans represent three-dimensional objects internally as a small set of bidimensional images. In this work we present a scheme for recognition of 3D objects from 2D images. The proposed approach begins by identifying the class of the observed object and only then proceeds to determine its individual identity. In this way, we are able to reduce the computational costs of an exhaustive comparison with all known objects. The developed system has no previous knowledge about existing objects and builds the object basis as it operates on given images.

Key-Words: Computer Vision, Object Recognition, 3D Objects, Features Net

1 Introduction

A three-dimensional object can produce excitatory patterns in the retina that vary widely depending on the object's position relative to the observer. In spite of that, we are able of perceiving these different patterns as produced by the same object. This ability of constant recognition from such input signals is the result of the brain's ability of establishing internal representations of the objects. The nature of such viewpoint invariant representations and the way they can be acquired is still one of the major unsolved problems in neuroscience and computational vision.

There are several behavioural studies that support a view-based model for the three-dimensional object representation used by our visual system. If we present human subjects a set of unfamiliar views of an object previously seen at a limited range of attitudes, there is an increase in the recognition error rate with misorientation relative to the training attitude [11]. The effect is reduced if intermediate views are considered. The performance is not linearly dependent on the shortest angular distance in 3D to the best view but is significantly correlated with an image-plane feature by feature deformation distance between the presented view and the best (shortest response time and lowest error rate) view [2].

Therefore, measurement of image-plane similarity to a few feature patterns seams to be an appropriate model for the mechanism used by the human visual system to recognize objects across changes in their 3D orientation.

Several physiological studies with monkeys also provide evidence of view-based processing by the brain during object recognition. Measurements of the activity in the inferior temporal cortex (IT) of the monkey, which has long been known to play an essential role in visual object recognition, support the results obtained in behavioural studies. Populations of IT neurons were found that responded selectively to views of previously unfamiliar objects. The cells discharged maximally to one view of an object, and their response declined gradually as the object was rotated away from this preferred view [7].

In summary, we can say that the representation of objects in the form of linked unique views seams to be sufficient for a wide range of perception situations and tasks.

The present work describes an attempt of incorporating 3D object recognition from 2D images in the work presented in [14]. The adopted approach is based on the orthographic projection of 3D objects in 2D images and consists of two stages. In the first stage, the categorization stage, the image is compared to prototype objects. For each prototype, we determine the view that's closest to the image and, if that view is similar to the image, we classify the object as belonging to the class represented by the prototype. In the second stage, the identification stage, the observed object is compared with the individual models in the same class. Each class groups objects with similar forms. For each model, a view is searched that is coincident with the image. If such a view is found, the identity of the object is established.

This recognition process follows closely the scheme proposed by Ullman and Basri [8]. However, it distinguishes from the work presented there as we tried to develop a recognition method where the knowledge base is built in an incremental way, without previous construction/categorization of an image library. This has lead to several modifications to the scheme proposed in [8], some of which have not a theoretical justification and will need further research.

The system has no initial knowledge and the object classes and models are built as new objects are seen. When the system is presented with the first 2D image, it uses it to build its first object. This object will initially be described by a single view, namely that first seen image. It will also be the single element of a new class, for which it will also be the prototype. After that, when given an object's image *I*, the system proceeds as follows. First, as explained above, it determines, for each known prototype, the closest view to the image and, if that view is similar to the image, the object is classified in the class represented by the prototype. If no similar views are found, the system uses the image to build a new class and object, using the same procedure that was described for the first image. If the object is successfully classified in an existing class, then the system tries to identify the object by comparing the given image to aligned views of all the objects already known to belong to the same class. During this stage, three things can happen. If a view is found that is coincident with the image, then the object is identified as the object represented by the model that originated such view. If no model can produce a view similar to the image, then image *I* is used to build a new model for the class. As a third possibility, the system can find a model with a view that is very close to *I*. In this case, the system rebuilds that model's representation adding to it the new view *I*. If more than one model are found on the same conditions, those models are merged in a single model.

2 Linear Combination Model

The representation scheme we adopted results from the linear combination model for 3D objects proposed by Ullman and Basri [12]. In that paper it's demonstrated that the set of all possible images obtained by orthographic projection of a 3D object that is subject to rigid transformations and scaling between images belongs to a linear space spanned by a small number of 2D images of the same object.

An object is modelled by a matrix *M*, with dimension $n \times k$, where n is the number of characteristic points and *k,* the number of columns in *M*, is related to the number of degrees of freedom the object has.

Let *O* be a 3D object that contains *n* characteristic points $(X_i, Y_i, Z_i), 1 \le i \le n$. Under weak perspective projection, the object's position in the image, after rotation *R*, translation $\vec{t}$ and scaling *s*, is given by

$$\begin{aligned} x_i &= sr_{11}X_i + sr_{12}Y_i + sr_{13}Z_i + st_x \\ y_i &= sr_{21}X_i + sr_{22}Y_i + sr_{23}Z_i + st_y \end{aligned} \quad (1)$$

where r_{ij} are the rotation matrix components, t_x and t_y are the horizontal and vertical components of the translation vector $\vec{t}$, respectively, and *s* is the scaling factor.

Denote by $\vec{X}, \vec{Y}, \vec{Z}, \vec{x}, \vec{y} \in \Re^n$ the vectors composed of the coordinates X_i, Y_i, Z_i, x_i and y_i, respectively, and define $\vec{1} = (1,...,1) \in \Re^n$. Then, we can rewrite (1) using vector form

$$\begin{aligned} x_i &= sr_{11}X_i + sr_{12}Y_i + sr_{13}Z_i + st_x \\ y_i &= sr_{21}X_i + sr_{22}Y_i + sr_{23}Z_i + st_y \end{aligned} \quad (2)$$

where

$a_1 = sr_{11}$	$b_1 = sr_{21}$
$a_2 = sr_{12}$	$b_2 = sr_{22}$
$a_3 = sr_{13}$	$b_3 = sr_{23}$
$a_4 = st_x$	$b_4 = st_y$

Hence,

$$\vec{x}, \vec{y} \in span\{\vec{X}, \vec{Y}, \vec{Z}, \vec{1}\}$$

Notice that the translation component can be ignored if the centroids of the points (X_i, Y_i, Z_i) and (x_i, y_i) are moved to the origin, that is, if we translate the object's and image points in such a way that

$$\sum_{i=1}^{n} (X_i, Y_i, Z_i) = (0,0,0),$$

$$\sum_{i=1}^{n} (x_i, y_i) = (0,0)$$

Therefore, all the views of the rigid object *O* are contained in a linear space 3D (ignoring the translation). The idea, now, is to use images of the object to build a base for this space. It can be shown that, in general, two views are sufficient [12].

Let $p_1 = (\vec{x}_1, \vec{y}_1)$ be a 2D image of *O* and let $p_2 = (\vec{x}_2, \vec{y}_2)$ be the image of *O* obtained after a rotation R (a 3×3 matrix). Consider, then, a new view of O, $p_3 = (\vec{x}_3, \vec{y}_3)$, obtained by applying a new rotation to O. We will have:

$$\begin{aligned}\vec{x}_3 &= a_1\vec{x}_1 + a_2\vec{y}_1 + a_3\vec{x}_2 \\ \vec{y}_3 &= b_1\vec{x}_1 + b_2\vec{y}_1 + b_3\vec{x}_2\end{aligned} \quad (3)$$

as long as the two images p_1 e p_2 don't differ only by a pure rotation along the line of sight [12].
The described linear combination of two views is applicable to general linear transformations. To impose rigidity (with possible scaling), we need to impose some restrictions on the coefficients *(a_1, a_2, a_3, b_1, b_2, b_3)*. In our implementation we ignored those restrictions with the risk of having some false positive misidentifications. However, the likelihood of such misidentifications is negligible if the objects contain a sufficient number of points.

Resuming, following the exposed scheme, an object is represented by a matrix M whose columns are built from views of the object, translated so that the centroid is at the origin, forming a basis for the 3D space.

The object's views can be built as follows

$$\begin{aligned}\vec{x} &= M\vec{a}, \\ \vec{y} &= M\vec{b},\end{aligned} \quad (4)$$

where $\vec{a}, \vec{b} \in \Re^k$ are the vectors formed with the coefficients in equation (3). Note that the two linear systems can be merged in a single system by constructing a modified model matrix,

$$\begin{bmatrix}\vec{x} \\ \vec{y}\end{bmatrix} = \begin{bmatrix}M0 \\ 0M\end{bmatrix}\begin{bmatrix}\vec{a} \\ \vec{b}\end{bmatrix}$$

The recognition process involves computing the transform vectors $\vec{a}$, $\vec{b}$.

Note that this scheme of linear combination of images assumes that the same object's points are visible in different views. When the views are sufficiently different this approach may no longer be valid because of auto occlusion. To represent an object from every possible direction (for example, seen from the front and behind), we will need several different models of this kind.

3 Categorization Stage

Recognizing an object implies, as a first step, determining the object's category by comparing it with prototype objects that are typical exemplars of their classes. For a given prototype, we compute the view that is most similar with the image. That view is compared with the actual image and the result of this comparison determines the identity of the object's class.

A class of objects is a pair $C = (P, \{M_1, M_2, ..., M_l\})$, where P is a prototype object for the class and $M_1, M_2,... M_l$ are model objects. The prototype and models are represented by matrices of point locations, according to the description made in the previous section.

Objects in the same class roughly share the same topology and there is a "natural" correspondence between them. This correspondence is made explicit by the order of the row vectors in the models. Specifically, given a prototype P and models M_1, M_2,... M_l, we order the lines in these models in such a way that the first characteristic point in P corresponds to the first characteristic point in each model M_1, M_2,... M_l, the second characteristic point in P corresponds to the second characteristic point in each model M_1, M_2,... M_l, and so forth. The importance of this ordering will become evident in the following sections.

To categorize an object observed in an image we need to align the prototype objects with the image so we can compare them. Therefore, for each prototype, we solve, first, the correspondence between the prototype and the image. Then, using the computed correspondence, the nearest prototype's view is determined.

Given a prototype P and an image I, we generate a vector $\vec{v}$ from the image with the location of the image's characteristic points ordered in correspondence with the prototype points: the first point in $\vec{v}$ corresponds to the first point in P and so forth. The transform vector $\vec{a}$ that brings the prototype points as close as possible to the corresponding image points is the vector that minimizes the Euclidian distance between the prototype and image points

$$\min_{\vec{a}'} \left\| P\vec{a}' - \vec{v} \right\|$$

If P is an over determined matrix, that is, if P has dimension $n \times k$, with $n > k$ and verifies $rank(P) = k$, then the solution of the above equation is given by

$$\vec{a} = P^{+}\vec{v} \quad (5)$$

where $P^{+} = \left(P^T P\right)^{-1} P^T$ denotes the pseudo-inverse of P. Then, the nearest prototype's view, $\vec{p}$, is obtained by applying P to $\vec{a}$

$$\vec{p} = P\vec{a} = PP^{+}\vec{v}$$

View $\vec{p}$ is then compared with the image and its similitude determines the object's classification. The matching quality between prototype and image is given by

Proceedings of the 5th WSEAS Int. Conf. on Signal Processing, Computational Geometry & Artificial Vision, Malta, September 15-17, 2005 (pp155-162)

$$D\left(P,\vec{v}\right)=\frac{\left\|\vec{p}-\vec{v}\right\|}{\left\|\vec{v}\right\|}=\frac{\left\|\left(PP^{+}-I\right)\vec{v}\right\|}{\left\|\vec{v}\right\|} \quad (6)$$

where I represents the identity matrix. The division by the norm of $\vec{v}$ normalizes the metrics (6) eliminating effects due to eventual scaling.

If the object belongs to the class represented by P, then the function defined by (6) will attain its minimal value when $\vec{v}$ is ordered in correspondence with P. Any other ordering of the points will increase the function value. Therefore, function D can be used as an objective function for determining the correspondence between prototype and image. Formally, denoting by π a permutation matrix, we define:

$$\hat{D}\left(P,\vec{v}\right)=\min_{\pi} D\left(P,\pi\vec{v}\right) \quad (7)$$

If we now define the cost of matching the point p_i in the image $\vec{p}$ with the poin q_j in image $\vec{v}$ as

$$Cij\left(p_i,q_j\right)=\left(p_i-q_j\right)^2$$

then minimizing (7) is equivalent to minimizing the function

$$H\left(\pi\right)=\sum_{i=1}^{n} C_{ij}\left(p_i,q_{\pi(i)}\right) \quad (8)$$

subject to the restriction that the matching is one-to-one, that is, π is a permutation. This is an instance of the square assignment problem (or weighted bipartite matching), which can be solved in $O(n^3)$ time using the Hungarian method. In our implementation we used the more efficient method of [9]. The input to the attribution problem is a square cost matrix C_{ij} and the output is a permutation π that minimizes (8).

In order to have robust handling of outliers, we add "dummy" points to each points set with a constant matching cost of ε_d. A point will be matched with a "dummy" whenever there is no real match available at smaller cost than ε_d. Thus, ε_d can be regarded as a threshold parameter for outlier detection. In a similar way, when the number of sample points on two sets is not equal, the cost matrix can be made square by adding "dummy" points to the smaller points set.

An object seen by view $\vec{v}$ belongs to the class represented by prototype P if

$$\hat{D}\left(P,\vec{v}\right)<\varepsilon$$

for a certain constant $\varepsilon > 0$.

In summary, given a prototype P and an image I, the correspondence between P and I is solved by minimizing the metrics (7) over all the possible permutations of $\vec{v}$ and, if the obtained minimum is below threshold ε, then the object's class is determined.

Although the general classification scheme defined doesn't depend on the specific distance metrics chosen, the metrics affects the division of the models in classes as well as the selection of optimal prototypes for those classes. We will show in section 5 how we can chose the optimal prototypes using the metrics (7).

In the following section we will explain how the transform vector can be reused to align the image with the specific models. Hence, after the categorization, the cost of comparing the image with each specific model is substantially reduced, as the complex part of recovering the transformation that relates the models with the image is applied only to the prototype objects.

4 Identification Stage

After categorizing the object, we seek determining its individual identity. In this stage, the image is compared with all the models belonging to the class obtained in the categorization process. For each model, the transformation that aligns the model with the image is determined, if it exists, using the information obtained in the categorization stage.

Let $\vec{v}$ be a view of the model object M_i, verifying

$$\vec{v}=M_i\vec{b} \quad (9)$$

for a certain transform vector $\vec{b}$. Then, it can be shown that

$$\vec{b}=A_i\vec{a} \quad (10)$$

where $\vec{a}$ is the transform vector for the prototype, given by (5), and $A_i=\left(P^{+}M_i\right)^{-1}$, assuming $\det\left(P^{+}M_i\right)\neq 0$.

This result is valid because the characteristic points in the prototype and in the models are aligned.

The linear transformation defined by matrix A_i is independent of the particular view $\vec{v}$ considered. That is, for any view of the object, the same transformation maps the prototype's transformation corresponding to the view in the correct model transformation. This means that the transformation A_i can be computed ahead and stored together with the model. Furthermore, the transformation A_i allows recovering the model's transformation independently of the matching quality between the prototype and the image. Even when the prototype aligns badly with the image, the transformation that aligns the model with the image is determined correctly.

As mentioned above, A_i exists if P^+M_i is invertible. This condition is equivalent to require that the two column spaces of P and M_i aren't orthogonal in any direction. This condition is verified, in general, as long as the two objects are relatively similar.

Denote $M_i^{'} = M_i A_i$ the model M_i aligned with the prototype P. $M_i^{'}$ model the same object as M_i, since the column vectors of both matrices span the same space. Moreover, the aligned model $M_i^{'}$ is brought by the prototype transform $\vec{a}$ to a perfect alignment with the image. Indeed, we can rewrite (10) as

$$\vec{v} = M_i^{'}\vec{a} \qquad (11)$$

Therefore, if the models are aligned with the prototype, the transformation computed in the categorization stage can be used for identification without further manipulations. This result allows simplifying the identification process. The models $M_1, ..., M_l$ are aligned with the prototype P applying the corresponding transformations $A_1, ..., A_l$. In the recognition, the prototype transform $\vec{a} = P^+\vec{v}$ is applied to the aligned models $M_1^{'},...,M_l^{'}$.

In the above description it is assumed that there is a total correspondence between prototype and image. However, this assumption is not mandatory. If the correspondence isn't total, the previous results are still valid if we eliminate, in matrices P e M, the lines corresponding to points with no correspondence in the image.

Also, note that the models in a class can have different degrees of freedom. Let k_i be the width of model M_i in a certain class $C = (P, \{M_1, M_2, ..., M_l\})$, $1 \le i \le l$. Then, the prototype for this class, P, will have width

$$k_p = \max\{k_1, \ldots, k_l\}$$

In this case, the prototype-to-model transform A_i will be

$$A_i = \left(P^+ M_i\right)^+$$

and A_i will have dimension $k_i \times k_p$.

5 Constructing the optimal Prototypes

In this section we will show how it is possible to determine the optimal prototype for a given class under metrics (7).

Given a class of objects, the optimal prototype is the object that most resembles the class's objects. In our formulation, such object must share the maximal number of characteristic points with the objects in the same class. The positions of those points in the prototype must be as near as possible to their positions in the objects and the prototype-to-model transformations must be as stable as possible. The prototype can be computed then using a principal component analysis.

The optimal prototype for a certain class is defined as the object P that minimizes the cost function

$$E(P) = \sum_{i=1}^{n} \int_{\|\vec{v}_i\|=1} \left\| \left(PP^T - I\right)\vec{v}_i \right\| d\vec{v}_i \qquad (12)$$

which corresponds to the sum, for every model in the class, of the distance $D(P, \vec{v}_i)$ between the prototype and all the model's possible views, with unitary norm.

In [8] it is proven that the prototype that minimizes equation (12) can be obtained by the following algorithm:

1 – Verify that the column vectors of each model's matrix M_i $(1 \le i \le l)$ are orthogonals. If that's not the case, apply the Gram-Schmidt orthogonalization method.

2 – Build the symmetric matrix n x n:

$$F = \sum_{i=1}^{l} M_i M_i^T \qquad (13)$$

3 – Find the k eigenvectors of F corresponding to the dominant eigenvalues. The optimal matrix P is built using these vectors.

The prototype determined by this process is independent on the chosen basis for the models. This implies that, in order to build the prototype, it's not required that the model objects $M_1, ..., M_l$ are aligned.

6 Implementation

The implementation of the above described processing is quite trivial. The implemented algorithm is described next:

1 – Given an image I, apply the algorithm developed in [14] to identify the objects present in the image. For each object found, construct the vector $\vec{v}$ with the locations of the image's characteristic points and proceed as follows.

2 – Translate vector $\vec{v}$ so that its centroid is brought to the origin, obtaining the new vector $\vec{v}\,'$. This is done by translating every point $v_i = (x_i, y_i)$ in $\vec{v}$ as follows

$$v_i^{'} = v_i - \sum_{i=1}^{n} \frac{v_i}{n}$$

where n is the total number of points in $\vec{v}$. Normalize $\vec{v}\,'$.

3 – Let P be the set of all prototypes and let Cl be the set of all classes. If $P = \varnothing$, proceed to 7.1.

Proceedings of the 5th WSEAS Int. Conf. on Signal Processing, Computational Geometry & Artificial Vision, Malta, September 15-17, 2005 (pp155-162)

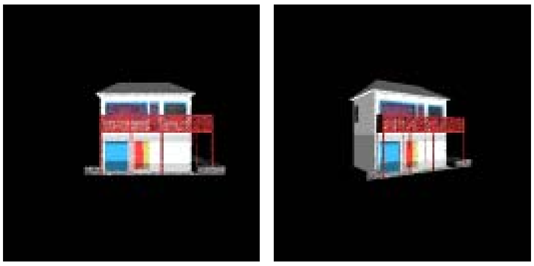

Fig. 1 – First and tenth view of the object House.

4 – For each prototype $P_j \in P$ determine the distance $\hat{D}\left(P_j,\vec{v}\,'\right)=\min_{\pi} D\left(P_j,\pi\vec{v}\,'\right)$, given by (7).

5 – Let $\pi_j = \arg\min_{\pi} D\left(P_j,\pi\vec{v}\,'\right)$ be the permutation that minimizes distance (7) for prototype P_j. Determine

$$d=\min_{j\in\{j:P_j\in P\}} \hat{D}\left(P_j,\vec{v}\,'\right)=\min_{j\in\{j:P_j\in P\}} D\left(P_j,\pi_j\vec{v}\,'\right).$$

6 – If $d < \varepsilon$

6.1 – For each prototype $P_j \in P$ such that $\hat{D}\left(P_j,\vec{v}\,'\right)<\varepsilon$

6.1.1 – Let M be the set of all models in the class represented by prototype P_j. Determine $d'=\min_{M_i\in M}\left\|M_iP_j^+\pi_j\vec{v}\,'-\pi_j\vec{v}\,'\right\|$.

6.1.2 – If $d' < \varepsilon'$, let $A_j=\left\{M_i\in M:\left\|M_iP_j^+\pi_j\vec{v}\,'-\pi_j\vec{v}\,'\right\|<\varepsilon'\right\}$.

6.1.2.1 – If $d' < \varepsilon''$, take $M'=M-A_j\cup\left\{\left[M_i\right]_{M_i\in A_j}\right\}$, where $\left[M_i\right]_{M_i\in A_j}$ represents the matrix of all the columns of all the matrices in A_j.

6.1.2.2 – If $\geq \varepsilon''$, take $M'=M-A_j\cup\left\{\left[M_i \quad \pi_j\vec{v}\,'\right]_{M_i\in A_j}\right\}$, where $\left[M_i \quad \pi_j\vec{v}\,'\right]_{M_i\in A_j}$ represents the matrix of all the columns of all the matrices in A_j plus the new column $\pi_j\vec{v}\,'$.

6.1.3 – If $d' \geq \varepsilon'$, take $M'=M\cup\left\{\pi_j\vec{v}\,'\right\}$.

6.1.4 – Make $Cl:=Cl-\left(P_j,M\right)\cup\left(P_j',M''\right)$, where P_j' is the optimal prototype for set M' and $M''=\left\{M_i\left(\left(P_j'\right)^+M_i\right)^{-1}:M_i\in M'\right\}$.

7 – If $d \geq \varepsilon$

7.1 – The object in the image doesn't belong to a known class. Therefore, add a new class to the model.

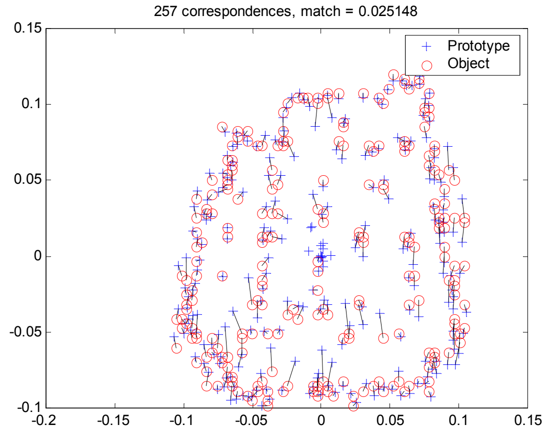

Fig. 2 – Comparison of the object house with the prototype (match = 0.025148).

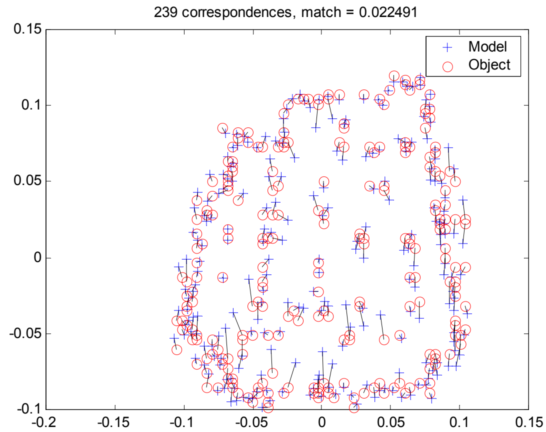

Fig. 3 – Comparison of the object house with the model (match = 0.022491).

This new class will have only one object, which will have only one view, $\vec{v}\,'$. The new object will be used as the prototype for the new class: $Cl:=Cl\cup\left(\vec{v}\,',\left\{\vec{v}\,'\right\}\right)$.

Threshold ε', in step 6.1.2, is the equivalent, for the models, to the threshold ε used in categorization. Threshold ε'', in step 6.1.2.1, is used to restrict the inclusion of new views in the models. The new view is not included in the model unless it differs from the model's aligned view by a value greater than ε''. In all simulations here reported we took $\varepsilon = 0.25$, $\varepsilon' = 0.15$ e $\varepsilon'' = 0.01$ (obviously, we should always have $\varepsilon'' \leq \varepsilon' \leq \varepsilon$).

7 Results

To test the system's ability of recognizing the same object seen from different viewpoints, we presented it

Fig. 4 – Image of the dog's 3D model.

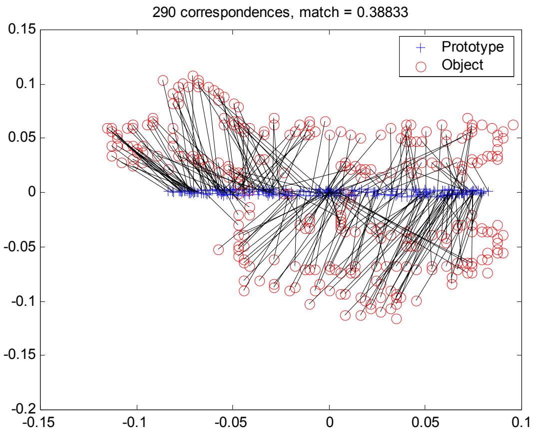

Fig. 5 – Comparison of the dog's image with the house prototype (match = 0.38833).

a set of ten images of a 3D model of a house (figure 1) obtained by successive rotations of 3.6º around the vertical axis. In this way, the first and last images presented were separated by a rotation in the horizontal plane of 36º. All the images were recognized as corresponding to the same object. The results obtained when comparing the tenth view with the prototype and the only model are shown in figures 2 e 3, respectively. The match value shown in those figures is the value obtained for metrics (7) and, therefore, smaller values correspond to better matches. Note also that the figures are rotated relative to the model's 2D image in figure 1.

Next, we presented an image of a different object to see if the system would be able of distinguishing the two objects. This time we used a 3D model of a dog (figure 4).

The results are shown in figure 5. The system was able to recognize it was in the presence of a new object.

8 Discussion

The results obtained, although in a very small number, are encouraging. Testing the validity of the proposed model will require defining a more demanding battery of tests. The model is very simple and attractive from the mathematical and computational perspectives. The threshold parameters used were chosen to produce the adequate results in the tests done and further research on the model will require carefully tuning of these parameters.

Aknowledgments: Raquel César is recipient of grant SFRH/BD/14359/2003 from FCT.

References:

[1] C. M. Cyr, B. B. Kimia, "3d object recognition using shape similarity-based aspect graph", In ICCV, 2001.

[2] F. Cutzu, S. Edelman, "Canonical Views in Object Representation and Recognition", Vision Research, 34:3037-3056, 1994.

[3] G. Peters, "Theories of Three-Dimensional Object Perception - A Survey", *Recent Research Developments in Pattern Recognition*, Vol. 1, pp. 179-197, (Part-I), *Transworld Research Network*, rivandrum, Kerala, India, 2000.

[4] G. Peters, Christoph von der Malsburg, "View Reconstruction by Linear Combination of Sample Views", *Proceedings of the 12th British Machine Vision Conference* (BMVC:2001), edited by Tim Cootes and Chris Taylor, University of Manchester, Vol. 1, pp. 223-232, Manchester, UK, September 10-13, 2001.

[5] G. Peters, C. von der Malsburg, "Learning Sparse Representations of Three-Dimensional Objects", *Proceedings of the 10th European Symposium on Artificial Neural Networks* (ESANN 2002), edited by Michel Verleysen, d-side, pp. 245-250, Bruges, Belgium, April 24-26, 2002.

[6] H. M. Gomes, R. B. Fisher, "Structural Learning from Iconic Representations", IBERAMIA-SBIA 2000, pp.399-408, 2000.

[7] N. K. Logothetis, J. Pauls, H.H. Bulthof, T. Poggio, "Shape Representation in the Inferior Temporal Cortex of Monkeys", Current Biology, 5(5): 552-563, 1995.

[8] R. Basri, "Recognition by Prototypes", *International Journal of Computer Vision*, 19(2): 147-168, 1996.

[9] R. Jonker, A. Volgenant, "A Shortest Augmenting Path Algorithm for Dense and Sparse Linear Assignment Problems", Computing, 38:325-340, 1987.

[10] S. Belongie, J. Malik, J. Puzicha, "Shape Matching and Object Recognition Using Shape Contexts", Vol. 24, No. 4, 2002.

[11] S. Edelman, H.H. Bülthoff, "Orientation dependence in the recognition of familiar and novel views of Three-Dimensional Objects", Vision Research 32(12):2385-2400, 1992.

[12] S. Ullman, R Basri, “Recognition by linear combinations of models”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):pp. 992–1006, 1991.

[13] S. Z. Li, J. Yan, X.W. Hou, Z.Y. Li, and H.J. Zhang, "Learning Low Dimensional Invariant Signature of 3-D Object under Varying View and Illumination from 2-D Appearances", In *Proceedings of 8th IEEE International Conference on Computer Vision*. Vancouver, Canada. July 9-12, 2001.

[14] T. Silva, “Reconhecimento Visual de Objectos por Coerência Estrutural de Características”, Tese de Doutoramento, IST, 2001.

[15] Z.Q. Zhang, L. Zhu, S.Z. Li, H.J. Zhang, "Real-Time Multi-view Face Detection", *Proceedings de 5th International Conference on Automatic Face and Gesture Recognition*. Washington, DC, USA. 20-21 May, 2002.