

# Fast and cheap object recognition by linear combination of views

J erome Revaud  
LIRIS UMR 5205 CNRS,  
INSA-Lyon  
F-69621, France  
jerome.revaud@insa-  
lyon.fr

Yasuo Ariki  
Dept. of Computer and  
System Engineering, Kobe  
University  
1-1 Rokkodai, Kobe,  
657-8501, JAPAN  
ariki@kobe-u.ac.jp

Guillaume Lavou e  
LIRIS UMR 5205 CNRS,  
INSA-Lyon  
F-69621, France  
glavoue@liris.cnrs.fr

Atilla Baskurt  
LIRIS UMR 5205 CNRS,  
INSA-Lyon  
F-69621, France  
abaskurt@liris.cnrs.fr

## ABSTRACT

In this paper, we present a real-time algorithm for 3D object detection in images. Our method relies on the Ullman and Basri [13] theory which claims that the same object under different transformations can often be expressed as the linear combinations of a small number of its views. Thus, in our framework the 3D object is modeled by two 2D images associated with spatial relationships described by local-invariant feature points. The recognition is based on feature points detection and alignment with the model. Important theoretical optimizations have been introduced in order to speed up the original full alignment scheme and to reduce the model size in memory. The recognition process is based on a very fast recognition loop which quickly eliminates outliers. The proposed approach does not require a segmentation stage, and it is applicable to cluttered scenes. The small size of the model and the rapidity of the detection make this algorithm particularly suitable for real-time applications on mobile devices.

## Categories and Subject Descriptors

I.4.8 [Image Processing and Computer Vision]: Scene Analysis—*Object recognition, Shape*; I.4.7 [Image Processing and Computer Vision]: Feature Measurement; G.1.3 [Numerical Analysis]: Numerical Linear Algebra—*Linear systems (direct and iterative methods)*; I.5.1 [Pattern Recognition]: Models—*Structural*

## General Terms

Algorithms, Performance

## Keywords

Object Recognition, linear combination, 2D views

## 1. INTRODUCTION

Although the recognition of familiar objects in any kind of environment may be a simple task for human beings, that is still of huge difficulty for computers. Indeed, changes in light or movements in space make images of the same thing look totally different. On the other hand, the number of devices that are able to capture pictures from everyday life has exploded. As a result, visual object recognition has become a real challenge, particularly to be able to classify this huge amount of data. Although most approaches treat object recognition as a complex process which requires powerful computers to run, we focus on simple linear combinations. Indeed, we want our system to be able to run on mobile devices such as cell phones or embedded systems. As a consequence, our approach has to take in account the constraints of these devices: few memory resources and limited computational power.

Existing object recognition algorithms can be classified as follows: the global and the local ones. The global techniques aim at recognizing the object in its whole. To achieve this result a global approach generally learns, from a set of images, the object to recognize. Then it extracts global features and finally uses statistical classification techniques. For instance, Nayar et al. [11] present a fast method which can handle one hundred objects while still being effective, using PCA and nearest neighbour search. Some other works have been done in face recognition [14] with boosted cascade of simple classifiers. However, these global methods share some drawbacks: First of all, the amount of data needed for learning is usually huge, as well as the training time. Another issue is that these approaches can not deal with partial occlusions. Moreover, the exact 3D position and orientation of the object may be difficult to retrieve, if not impossible.

On the other hand, local approaches mainly focus on the so-called keypoints (i.e. local features). In this context, the general scheme for object recognition usually implies three

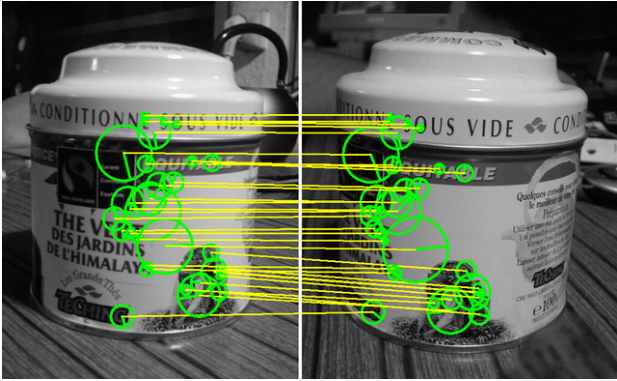


Figure 1: The model, a tea box, is composed with two pictures where the object slightly differs by its pose. The yellow lines represent keypoints matches and constitute the only information needed by our method (i.e. 2D positions of the keypoints and their descriptors).

steps: (1) first is the extraction and description of local features, in both test and model images. (2) The next step consists in selecting image features that match with the model ones. (3) The final step elects the best subset of keypoints (the inliers) that presents the highest correlation with the model. In that manner, the object position can be precisely extracted as well as the occlusion, provided that the model object is covered by a sufficient number of local features [3]. The keypoints detectors are generally fast and multi-scale. Another point that can explain why keypoints have become so popular these last few years is that the concept of decomposing an object into small interest patches is somehow familiar with the human visual system.

Nowadays, many authors have presented their own keypoint detector [5, 6, 8]. At the same time, the associated description problem (i.e. how to describe the neighbourhood of these points in an invariant way) has also been widely addressed by the scientific community [8, 2]. Lately, affine region descriptors have been heavily developed [12, 3] as they allow to recover the position of a 3D object from a single patches match. A recent state-of-the-art about affine region detectors can be found in [10]. Nevertheless, these methods based on affine descriptors are too slow for our real-time purpose. In our algorithm, we use SURF (Speed Up Robust Feature), a recent keypoint detector and descriptor, which has been developed by Bay et al [2]. This system is based on a similar approach than SIFT's [8] but seems to give lightly better matching results. Moreover, its vector size is half than SIFT's -as well as the comparison time between two keypoints-, and the whole detection-description process is also quite fast.

About recognition, recent advances have improved both robustness and detection speed. Lowe [9] combines multiple views of the same object in order to compute a set of *characteristic* views. Then their algorithm relies on a probabilistic model to determine whether the object is present in the scene or not, with respect to 2D similarity transforms of the model views. The drawback of such an approach is that it does not take into account the real 3D shape and the 3D transformations of the object and hence is unable to recover



Figure 2: Result of recognition over a cluttered image. Red crosses indicates the final assumed positions of the keypoints defined in the model. Despite the fact that the box is viewed from a different angle than in the model, the recognition achieves a nearly exact extrapolation of the model pose. The whole process spends about 1s for keypoints extraction and 0.01s for recognition.

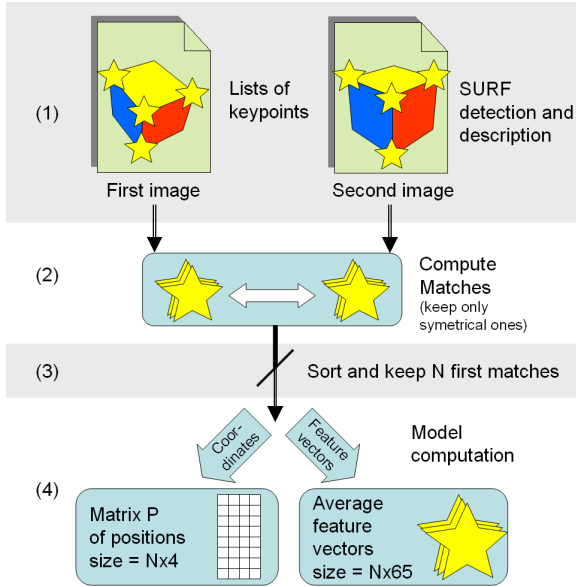
its precise spatial pose. In a different way, Rothganger et al. [12] consider affine keypoints to recover more efficiently the object pose from the matched feature patches. However, the system needs some time to detect an object. More recently, Lepetit et al. [7] present a real-time system: they obtain fast and robust keypoint classifiers based on randomized trees. Their solution is notably robust against changes in view point and illumination. Even if it gives good results, the training time is sizeable and the 3D geometric shape of the object has to be learned (one alternative is to consider the object to be locally planar). On the contrary, our method does not need to know the 3D geometric information, but the result is spatially constrained as if a 3D model was used. Moreover, no training time is required to achieve fast and efficient recognition.

Our method relies on the *linear combination* theory from Ullman and Basri [13], proposed in 1991: One may be able to recognize any 2D view of a 3D object from a limited number of its already known 2D views without storing its 3D geometric shape and even if its pose is different. The key idea is that the same object under different transformations can be expressed as the linear combination of a small number of its views. The interest of this approach is to obtain a real 3D shape recognition, but sparing the storing and computation cost of a 3D mesh and thus save time both in model construction and recognition. Considering this theory, our algorithm is the following: For a given object to index, we construct our model thanks to two of its views (differing by their viewpoints). These images are aligned using local invariant feature points. The model size is quite small and the recognition loop very fast. Finally, there is no training time, so the user can add new objects into the database in near real-time. We considered the SURF [2] keypoint detector to create the model but the presented framework could be generalized with other invariant feature points.

The paper is organized as follows: Section 2 presents how to build the model from two object images. The recognition is detailed in section 3 while results and conclusion are presented in the last sections.

## 2. MODEL CONSTRUCTION

The model construction includes two steps: first of all, we basically align two input images of the object by using the feature point detector SURF [2]. Then, the model feature vectors are computed based on the most representative matches. A match is a couple of keypoints  $(k, k')$  that refers to the same surface patch of the object, where  $k$  is belonging to the first image and  $k'$  to the second. Matches are computed thanks to the description vectors associated with feature points (see figure 3).



**Figure 3: Schema for model construction:** (1) key-points are detected and depicted by SURF, (2) they are matched symmetrically, (3) some matches are selected over their scale and strength, (4) the matrix containing positions  $X_1, Y_1, X_2$  and  $T$  is orthogonalized into  $P$ . Average feature vectors are also built from the matches vectors.

### 2.1 Keypoints alignment

Our method begins by computing the keypoints along with their descriptors on slightly different views of the same object. In our case, two images only are needed to represent an affinely invariant view of the object. After that keypoints are extracted, they are matched between the two images with the nearest neighbour ratio matching strategy (see [2]). Since we want the matching scheme to be symmetric, this strategy is applied from both sides. The final correspondence set is formed by the intersection of the two matching sets.

Then, we elect a subset of the most representative matches by evaluating each couple from its attributes:

$$score = k.strength \cdot k.scale \cdot k'.strength \cdot k'.scale$$

This score represents the quality of the match. Indeed, the bigger is the scale, the farer the spot can be detected. The strength is also a determining factor to improve robustness over illumination changes. In the following, we keep only the  $N$  better correspondences regarding this score. Figure 1 illustrates such a match with  $N = 32$ .

Optionally, one can eliminate spurious matches at this step.

This operation can be realized manually or automatically, considering that the field of vectors linking matched keypoints must have some homogeneous characteristics. The presence of spurious matches has to be avoided since it can interfere with the recognition process. That is why we have to make sure that the model is clean before to continue to the next step.

Finally, we assume that the two sets of keypoints descriptors are nearly identical for both images 1 and 2. Therefore, only their average is preserved in the model dataset. For each couple  $(k, k')$ , the corresponding average  $M$  is defined as follows:  $\forall i \in [1, 64], M.V_i = (k.V_i^1 + k'.V_i^2)/2$ , with 64 the number of components defined in SURF. This operation is theoretically allowed by the descriptor since every component  $V_i$  is already a geometric sum of Haar wavelet coefficients in the neighbouring of the point. The  $N$  vectors of keypoint descriptors constitute the first part of our model.

### 2.2 Linear combination of views

As we said in the introduction, the construction of the model is basically processed according to the theory presented in [13]. The statement demonstrated by the authors is that the set of possible images of an object undergoing linear 3D transformations (rotation, scaling, etc.) is embedded into a linear space of a lower dimensionality and spanned by a small number of views. To understand this model, we need a different vision of images. Usually, images are interpreted as 2D matrix. On the contrary, the authors' vision of images is parametric. A picture is thus expressed as a vector containing the coordinates of every points of the image (or the object) sorted in an arbitrary way. In their original paper, the authors gave a lot of possible variations of their model. Because this is only a short recap, we will only focus on the most interesting method in our eyes: the minimal model.

The minimal model proposes to express every possible view of an object from only two others of its views, provided that the object is considered to be somehow translucent - no occlusion *at all*. Formerly, the proof is quite simple. Let  $O$  be a rigid object (*i.e.* an ordered collection of 3D points),  $P_1$  an image of  $O$ , and let  $P_2$  be the image of  $O$  following a rotation  $R$  (a  $3 \times 3$  matrix). We will denote by  $r_1, r_2, r_3$  the three rows of  $R$  and by  $e_1, e_2, e_3$  the three rows of the identity matrix. For a given 3D point  $\rho$  of  $O$ , its coordinates  $(x_1, y_1)$  in the first image  $P_1$  are given by  $x_1 = e_1.\rho, y_1 = e_2.\rho$ , and similarly  $x_2 = r_1.\rho, y_2 = r_2.\rho$  for  $P_2$ . Consider now any other view obtained by applying another  $3 \times 3$  matrix  $U$  to the points of  $O$ . The coordinates  $(\hat{x}, \hat{y})$  of  $\rho$  in this new view will be  $\hat{x} = u_1.\rho, \hat{y} = u_2.\rho$  (where  $u_1, u_2$  are the first two rows of  $U$ ). Assuming that  $e_1, e_2$  and  $r_1$  span  $\mathbb{R}^3$  (which is true unless  $R$  is a pure rotation around the line of sight), then

$$u_1 = a_1.e_1 + a_2.e_2 + a_3.r_1$$

for some scalars  $a_1, a_2, a_3$ . Therefore

$$\hat{x} = u_1.\rho = (a_1.e_1 + a_2.e_2 + a_3.r_1)\rho = a_1.x_1 + a_2.y_1 + a_3.x_2$$

Similarly, for the y coordinates

$$\hat{y} = u_2.\rho = (b_1.e_1 + b_2.e_2 + b_3.r_1)\rho = b_1.x_1 + b_2.y_1 + b_3.x_2$$

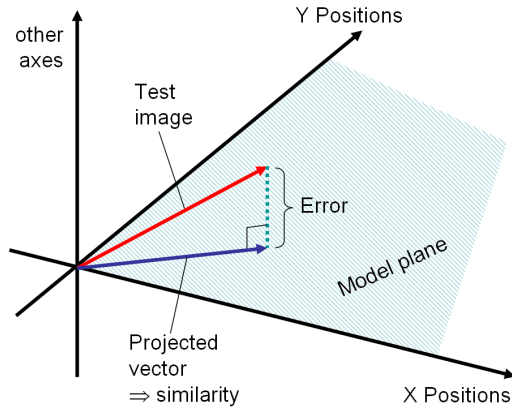
This equality holds for every point  $\rho$  from  $O$ . Moreover, for a given 3D position the scalar coefficients are the same for

every point from  $O$ . Let now  $X_1$  be the vector of all the  $x$  coordinates of the feature points in the first view,  $X_2$  in the second,  $\hat{X}$  in the third and  $Y_1$  the vector of all  $y$  coordinates in the first view. Then

$$\hat{X} = a_1.X_1 + a_2.Y_1 + a_3.X_2$$

$$\hat{Y} = b_1.X_1 + b_2.Y_1 + b_3.X_2$$

One can view the situation as follows: within a  $N$ -dimensional space (with  $N$  the dimension of the above vectors, *i.e.* the number of selected matches in our case),  $X_1, Y_1$  and  $X_2$  span a three-dimensional subspace. For all the image of the considered 3D object, both coordinates vectors  $\hat{X}$  and  $\hat{Y}$  theoretically reside within this three dimensional subspace. Figure 4 illustrates this mechanism for a 2D subspace. Let



**Figure 4: The Ullman-Basri theory for recognition, adapted into a 2 vectors model for representation. The X and Y positions axes figures the  $X_1$  and  $Y_1$  model keypoints’s coordinates. The third axe symbolises every axes that are perpendicular to the first ones. the ratio of the projection error norm to the image norm allows to test the similarity between a test image and the model.**

us cluster these three vectors inside a matrix  $P = [X_1, Y_1, X_2]$ . The recognition process will then use  $P$  to match between keypoints 2D position and model 3D coordinates. However, this method is still unable to deal with object translation. This is caused by the fact that the three vectors are implicitly defined as centered on the object rotation point defined between both views. To solve this issue, we add a fourth vector  $T = [1, 1, \dots, 1]$ . By this mean, the translation becomes possible without modifying the theory. Finally, we orthogonalize  $P$  for optimization purposes (see section 3.3). This last operation does not modify the subspace spanned by the vectors.

One can object that this approach only considers the orthogonal projection of an object over a plan, and thus neglects perspective effects. Despite this approximation, the theory gives good results even in close-up conditions (see for instance figure 6).

To summarize (see figure 3), we construct a very small model in terms of memory cost. The total size of the model is  $68N$  floating point values, composed of  $65N$  values for the SURF vectors and  $3N$  values for the positions matrix (the fourth vector is dispensable). The model takes less than one second

to be built as the only complex process is to widely match keypoints, which is in  $O(M^2)$ ,  $M$  being the average number of keypoints per image ( $M \approx 700$  in our case). We are now going to study how the recognition is processed from this dataset.

### 3. RECOGNITION STAGE

In their paper, Ullman and Basri [13] proposed three general schemes for recognition. We have only focused on the second one, the full alignment. This choice was motivated by the extra-possibilities it offers, such as handling partial object occlusion. First, we present how the problem was handled by Ullman and Basri. Then, we detail our optimization scheme to speed up the alignment process. Finally, our recognition loop is presented; it provides a quick convergence toward the solution while efficiently eliminating outliers.

#### 3.1 Full alignment scheme

We saw above that the  $P$  matrix is defined from only four vectors. To compute full alignment in a vectorial context, we build a matrix  $L$  which maps every model vector into a defined vector  $Q$  and every other perpendicular vectors into itself. To build this matrix, we transform  $P$  into a square matrix by filling the other columns such as  $\bar{P}$ ’s columns are all linearly independent. Then, we build  $\bar{P}$  as equal to  $P$  in all respects, except for the four first columns. These are substituted by the same vector  $Q$ . Then, we require that

$$LP = \bar{P}$$

Therefore,

$$L = \bar{P}P^{-1}$$

Note that  $P$  is composed of linearly independent vectors, so  $P$ ’s inverse always exists. A simple choice for  $Q$  is the null vector. This way,  $L.P_i = 0$  for  $i \in [1, 4]$  with  $P_i$  the  $i^{th}$  column of  $P$ . In other words,  $L.P_i = 0$  for  $P_i \in \{X_1, Y_1, X_2, T\}$ . As a consequence, any other view  $V$  which is a linear combination of these four vectors gives the null vector:  $L.V = L(\sum_{i=1}^4 a_i.P_i) = \sum_{i=1}^4 a_i.L.P_i = 0$ . Inversely, in the pure-noise condition, the output would be  $L.V = V$ . In fact, the view  $V$  is composed with two vectors  $X$  and  $Y$ , however in all the formula we consider  $V$  as  $X$  or  $Y$ . Taking a decision whether  $V$  is a view of the object or not can be simply based on the comparison of  $\|L.V\|$  with  $\|V\|$ . If  $V$  is indeed a view of the 3D object, this ratio will be small (0 in the noise-free condition). On the contrary a score close to 1 indicates a high difference with respect to the model (see Figure 4) :

$$score = \sqrt{\frac{\|L.X\|^2 + \|L.Y\|^2}{\|X\|^2 + \|Y\|^2}} \quad (1)$$

#### 3.2 Occlusion problem

During the alignment process, some points may remain unmatched because of occlusions. This issue was also addressed by Ullman and Basri [13] in the case of full alignment. Indeed, the authors proposed a method to recover the coordinates of the missing points using the  $L$  matrix.

Considering that the first  $k$  coordinates of  $V$  are unknown, the problem is then to minimize  $\|L.V\|$ . These first  $k$  components are initially equal to 0. We construct a new vector  $\hat{V}$  from  $V$  by supplementing the missing coordinates:

$$\hat{V} = V + U.A,$$

where  $U$  is the span of unit vectors along the first  $k$  coordinates and  $A$  the vector of corresponding unknown values. Then, solving  $\min_A (\|L.V + L.U.A\|)$  comes to the solution  $A = -[L.U]^+ L.V$  (where  $[L.U]^+$  denotes the pseudo-inverse matrix). Hence we are able to retrieve 2D coordinates of the missing keypoints, which are interpolated from the known ones using the  $L$  matrix.

However if less than 4 keypoints are initially correct, whatever their positions, we do not obtain relevant interpolated positions for the missing ones. Since the subspace is spanned by four vectors, the resulting vector  $\hat{V}$  is inevitably in the model subspace (see figure 4).

To conclude with this issue, unmatched keypoints are not a problem since their coordinates can be interpolated from other matches without lowering the recognition score. However this technique has a computational cost which is quite high. We will see below that this issue can also be solved.

### 3.3 Optimization

The original approach from Ullman and Basri requires a pseudo-inverse computation of the  $L.U$  matrix, whose dimension could theoretically be up to  $N \times N$ , at each model-image matching operation. With respect to our implementation, this would be a problem since this step is processed at each iteration of our solving loop. Nevertheless, it is possible to use less memory and to increase speed in the same time. Indeed, the  $L$  matrix becomes obsolete for score computing if  $P$  is orthogonalized. In that case, the score is simply the correlation between the vector to test and the subspace defined by  $P$ , which is now reduced to the original 4 columns matrix  $\{X_1, Y_1, X_2, T\}$  (see figure 4). Indeed, our algorithm directly projects the vector  $V$  on  $P$  with a scalar product. Because the orthogonal projection gives the minimal distance toward a subspace, this method can also retrieve both coefficients  $A = [a_1, a_2, a_3, a_4]^T$ ,  $B = [b_1, b_2, b_3, b_4]^T$  and subspace coordinates  $(\hat{X}, \hat{Y})$ .  $\hat{X}$  and  $\hat{Y}$  correspond to the extrapolation (i.e predicted position) of the model keypoints:

$$A = P^T X \Rightarrow \hat{X} = PA$$

$$B = P^T Y \Rightarrow \hat{Y} = PB$$

However, the  $L$  matrix was still helpful to interpolate missing points coordinates, but it can also be spared. If there exist occlusions, some rows from  $P$  and  $V$  are set to 0. Then the main problem is that  $P$  is no longer orthogonal when deprived from some coefficients (the missing ones). Let's call this matrix  $P'$  which is formally

$$P' = U.P = [P'_1 \ P'_2 \ P'_3 \ P'_4]$$

where  $U$  is the  $N \times N$  identity matrix deprived of the rows corresponding to missing keypoints (that is, not suppressed but set to 0).  $P$  and  $P'$  are  $N \times 4$  matrices. We formally search for  $A$  coefficients so that  $U.\hat{X} - U.X = P'.A - U.X$  is minimal (and similarly for  $B$  and  $Y$ ). This minimum is reached for

$$\begin{aligned} A &= P'^+ U X \\ &= [P'^T P']^{-1} P'^T U X \\ &= [P'^T P']^{-1} (U P)^T U X \\ &= [P'^T P']^{-1} P^T (U X) \\ &= [(U P)^T (U P)]^{-1} P^T (U X) \end{aligned}$$

Thanks to the fact that  $P$  is an orthogonalized matrix, one can prove that  $(U P)^T (U P) = I_4 - [(I_n - U) P]^T [(I_n - U) P]$ . Depending on the ratio of  $k$  to  $N$  ( $k$  being the number of missing keypoints), the user can compute the  $[P'^T P]$  matrix in two manners. If  $k < N/2$ , the expression  $I_4 - [(I_n - U) P]^T [(I_n - U) P]$  should be preferred to  $(P'^T U P)$ , because faster to compute ( $O(k)$  instead of  $O(N - k)$ ).

Concretely, this whole optimization reduces the memory cost from  $N^2 + 4N$  ( $P$  and  $L$ ,  $N \times N$  matrix) to  $4N$  ( $P$  only), which represents a gain of  $1 + N/4$ . The speed of the alignment scheme is also boosted from  $O(k^3 + N^2)$  to  $O(N)$ .

### 3.4 Recognition loop

*Alignment.* The keypoints matching is processed in the perspective of a full alignment scheme. As a consequence, we basically try to find a correspondent in the test image for each model's keypoint. The Euclidean distance (between descriptors) is then computed  $N \times M$  times, with  $N$  the model's number of keypoints and  $M$  the image's. For each model keypoint  $i$ , we keep a list  $L_i$  of every keypoint in the image that presents a distance (between descriptors) inferior to a given threshold  $T_{match}$ . In each list, the top ranked keypoint coordinates constitutes the  $X$  and  $Y$  vectors that initially feed the detection loop.

As we saw in section 3.3, we can compare the vectors  $X$  and  $Y$  with the  $P$  matrix. The score defined in equation 1 gives an idea of the colinearity between the vector and the model. However, this value is impaired by the wrongly aligned or missing keypoints. Moreover, it can happen that the model does not suit at all the image but still gives a high score. To answer this issue, we have developed a loop which converge quickly toward the solution considering the spatial coherency of the model (see figure 6). The loop comprises three steps.

1. First comes the extrapolation of the model from the matches. Starting from  $X$  and  $Y$  (white circles in fig. 6), we compute the predicted model compliant positions  $\hat{X}$  and  $\hat{Y}$  (red crosses in fig. 6) using equations from section 3.3. Even missing key points are predicted.
2. We eliminate the keypoint that is the farthest from its predicted position. To that aim, the distance (represented as a blue segment in figure 6) is computed in an Euclidean way :

$$Dist = (X - \hat{X})^2 + (Y - \hat{Y})^2 - S$$

$Dist$ , a  $N$  rows vector, represents the square value of the distance between each keypoint and its predicted position. This distance is corrected by  $S = [scale_1, \dots, scale_N]^T$ , the vector of the scales of the current image keypoints. We realize this correction because the larger is the keypoint, the more imprecise is its location. Our assumption is that the imprecision is roughly linear with the square root of the keypoint scale. Our experiments have confirmed that this approximation is accurate enough to correctly eliminate outliers. Finally, the image keypoint  $K'_i$  associated

to the model keypoint  $K_i$  with  $i = \text{argmax}_i(\text{Dist})$  is disconnected from  $K_i$ . As a result, this suppresses a match and thus adds a “missing keypoint” (*i.e.* its position is interpolated from the others matches in the following iteration).

The loop exits at this point if one of the two following conditions is reached:

- The distance  $\text{distModel} = \max_i(\text{Dist})$  is below a threshold  $T_{\text{dist}}$ .
  - The number of remaining matches is less than  $T_{\text{remaining}}$ .
3. The last step attempts to recover an hypothetical keypoint that would fit to the actual extrapolated model. Over every lonely model keypoint  $K_i$ , we select the image keypoint  $K'_i$  in the corresponding list  $L_i$  which presents the smallest spatial distance to its predicted position. The distance has to be inferior to  $\sqrt{\text{distModel}}$  in order to keep the loop converging. This keypoint is reintegrated as a match into  $X$  and  $Y$ .

The vectors  $X$ ,  $Y$ ,  $\hat{X}$  and  $\hat{Y}$  thus lightly evolve at each iteration (see figure 6). The third step allows the algorithm to recover from its mistakes along the convergence. Theoretically, this loop offers no exit guarantee. Indeed, at each step, one keypoint is eliminated whereas another can be reintegrated. To solve that issue, we arbitrarily restricted the maximum number of iterations to  $8N$ . We have not reached yet this limit in practice.

Figure 5 illustrates the whole recognition strategy. At the end, we have to evaluate the final alignment with the model. We define a score measure as function of the exit parameters. It is far more reliable than the score defined in [13]. Indeed, the score from eq. 1 of the remaining keypoints after the last iteration is often extremely low (*i.e.* good), even in case of bad alignment. That is coming from the fact that with a small number of remaining keypoints, the method is always able to find a subset of the initial points that match the model. To overcome this issue, we base upon the exit conditions to evaluate the recognition:

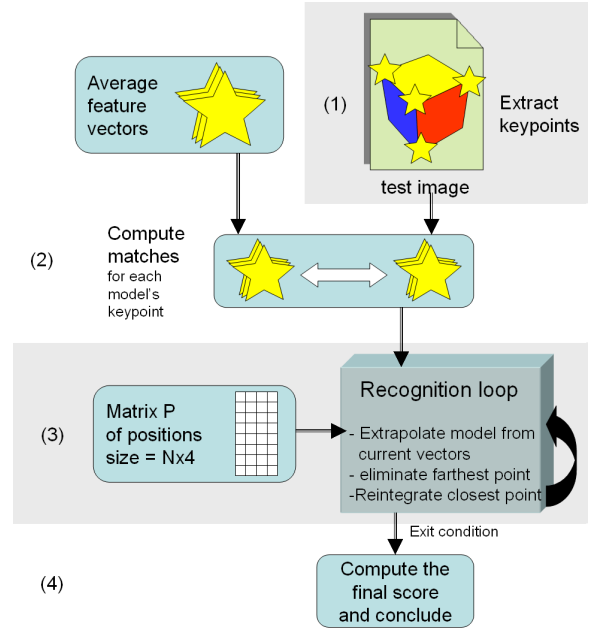
$$\text{score} = \frac{T_{\text{dist}}}{\text{distModel}} \cdot \frac{\text{nbRemaining}}{T_{\text{remaining}}}$$

In case of perfect matching, the score is infinite. On the contrary, it tends toward 0 for bad alignments. In our experiments, we observed that a good threshold to evaluate the recognition decision is 2.

## 4. RESULTS

We have tested our algorithm on various objects. The use of the SURF algorithm for keypoint detection makes our algorithm particularly suitable for textured objects. In this paper, we give some examples of recognition with a tea box and a roller skate.

Each model is built from two 600x500 images where are selected the  $N = 32$  top-rated keypoints (see figures 1 and 8). This number is a compromise between having enough points to tolerate occlusions while not having too much small-scaled keypoints. However, the value  $N = 32$  is not an optimal value insofar as we did not focus on optimizing



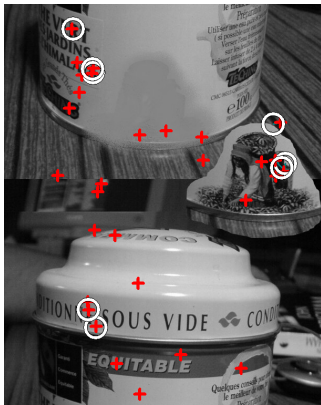
**Figure 5: The recognition process.** (1) Keypoints are extracted from the image. (2) The process retains every image-model match whose distance is less than a threshold. (3) The recognition loop extrapolates the model subspace from the keypoints coordinates matched previously. The process iterates by deleting the farthest keypoint with respect to the extrapolated model. (4) When one of the loop’s exit conditions is reached, the process compute the final score and conclude on the presence of the object.

it. Recognition seems to give similar results for values of  $N$  between 25 and 60. However, the choice of  $N$  has a direct influence on the model size. In that condition, the file size of a model is about 17 Ko only. This potentially allows to load many models simultaneously in memory. The recognition tests are performed on 1600x1200 images where the object, when it is present, has a size of about 300x200. Every threshold was set experimentally without focusing much on optimization. The maximal distance for keypoint matching is set to  $T_{\text{match}} = 0.25$ . The thresholds for loop exit worth  $T_{\text{dist}} = 45$  and  $T_{\text{remaining}} = 8$ . Figure 6 illustrates an example of recognition with a score of 2.4 on an image that is artificially blurred, resized smaller and presents a difference of angle of  $45^\circ$  with the model. The figure 7 demonstrates the consistency of our spatially constrained technique since the recognition do not detect the object manually disrupted.

The PC used for the experiments is an Athlon 1.85GHz with 512 Mo RAM running under Windows. This can not compare with small devices like cell phone, but it still gives an idea about the computational power needed by our system. The average recognition performance are specified in table 1 for two size of test images: 800x600 and 1600x1200. The slowest part of our algorithm is the SURF keypoints detection and description which represent 99% of the processing time. In a 1600x1200 image, the average number of detected keypoints is about 2000. We used the online available imple-



**Figure 6: Progression of the recognition loop over a test image at iterations  $i = \{1, 3, 6\}$  with  $N = 16$  keypoints. The white circles represent keypoints that are considered as inliers. Red crosses refer to positions that are predicted by the model. Lonely red crosses refers to missing matches caused by occlusion or keypoints detection failure. They are interpolated from the previous ones in a 3D manner. The test image was shot about  $45^\circ$  from the model image’s line of sight, resized smaller and blurred. The final recognition score is about 2.4 on that image.**



**Figure 7: Another tea box image which has been manually disrupted. The recognition achieves a low score of 0.09 on this image because of the lack of spatial consistency. The resulting extrapolation clearly demonstrates the failure of recognition.**

mentation of SURF for Windows (1.0.7) [2]. However, our algorithm achieves real-time performances; in particular the solving loop is almost instantaneous (less than one millisecond). Practically, our algorithm is particularly suited for real-time applications on embedded systems, provided that keypoints detection and description is fast enough.

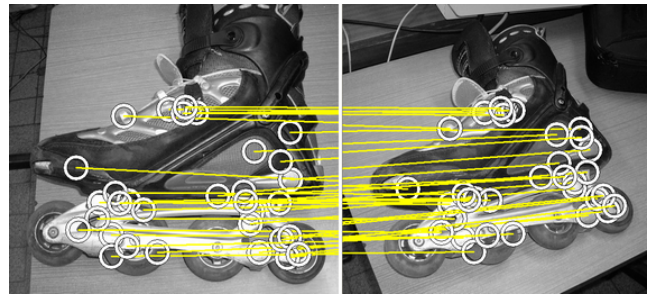
Scores that are between 0.5 and 2 often indicates partial recognition: the object was found but the system did not manage to eliminate some outliers. This happens in images of poor conditions. The algorithm’s convergence speed does not seem to be a function of the length of the lists  $L_i$ . Concretely, the loop exit is reached after about 40 iterations for  $N = 32$  keypoints. The initial keypoint repartition in a test image can reach 60% of outliers at the initial step without harming the recognition. That is thanks to the keypoint reintegration which progressively enforce the inliers assumption. However, in the case of two same objects in one image,

**Table 1: Average recognition time**

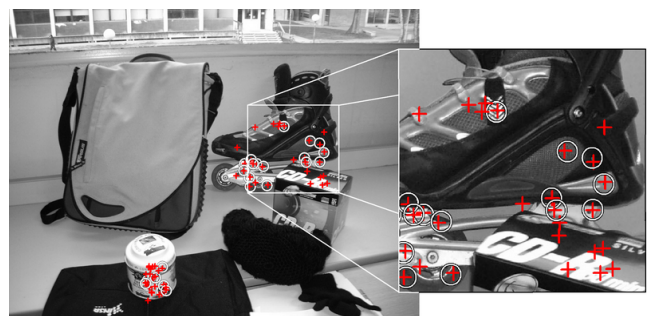
Stage	Time (ms) ( $800 \times 600$ )	Time (ms) ( $1600 \times 1200$ )
Keypoints detection	125	750
Keypoints description	328	1200
Keypoints matching	1	20
Solving loop	$\ll 1$	$\ll 1$
Total	454	1971

our technique is not able to differentiate them. This is still an issue that needs to be settled.

Some results are presented in the figures 2, 9, 10 and 11. The tea box is detected in figure 2 with a score of 5.23, despite a different orientation (about  $40^\circ$  from the model pose) and a cluttered background. In this figure, only the model extrapolation is presented (red crosses). The roller skate model is presented in figure 8 (yellow vectors symbolize keypoint matches between one model image and the other). Figures 9 and 10 illustrate correct recognitions of this object, despite an important occlusion (bottom right part) in the first figure and the cluttered background in the second one with final scores of 3.6 and 7.4 respectively. Finally, the figure 11 shows an example of recognition attempt on a cluttered image where the roller is absent, with the same graphical convention as above. The score for this image clearly bears out that fact with a value of 0.03.



**Figure 8: The roller skate model. White circles indicate the positions of keypoints used in the model. Yellow vectors symbolize matches between those keypoints.**



**Figure 9: The recognition over two different object in the same image. Each object was correctly localized with a score of 3.6 and 2.4 for the roller skate and the tea box respectively.**

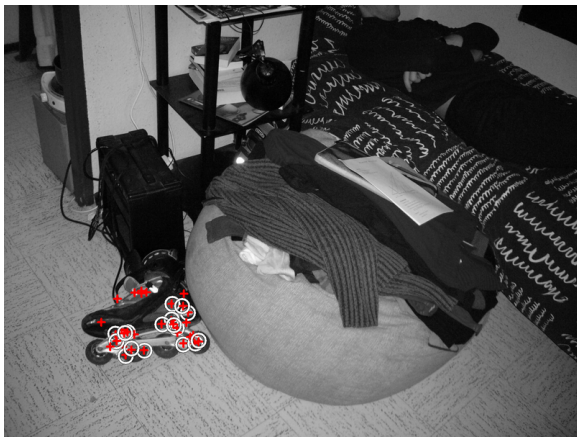


Figure 10: The roller is correctly localized in a highly cluttered picture with a score of 7.4.



Figure 11: An example of “empty” image: since the roller skate is absent from the picture, the algorithm converges toward an irrelevant solution. The score bears out that result with a value of 0.03.

## 5. CONCLUSION

We have presented an efficient method for 3D object recognition. Our scheme is faster and smaller than most of existing ones. The main interest of our approach results in minimizing the storing cost of the model and in quickly and accurately detecting the object inside a cluttered background. Our whole system is near real-time and simple to implement. To be fully real-time, we should use a faster keypoint matching mechanism. If so, it could outfit robots with a limited computational power.

One other point is that we can afford large object databases. Indeed, the slow keypoint extraction needs to be realized only once per image, whereas our recognition process can afford many executions.

Finally, another advantage of our approach is that we can deduce the exact positions and orientation of the object. Pose estimation can be obtained from the final state variable  $A$  and  $B$  if the model spatial conditions are known (or if a third model picture is available).

To some extents, our system is robust to object deformation provided that it still spans the subspace defined by the model, that is, for affine 3D deformations. We plan to still enhance the robustness of our solving algorithm. Some heuristics could be used to improve the convergence process. For instance, we do not take advantage of the keypoint orientation and scale. We also plan to take into account the model rigidity: this could also provide some useful rules for

accelerating convergence.

The efficiency of our recognition framework depends on the considered feature point detector. Whereas SURF is rather adapted for textured objects, we could consider other features in order to improve the capability range, like the plane curves described in [4] or the spread edges of [1].

## 6. REFERENCES

- [1] Y. Amit, D. Geman, and X. Fan. A coarse-to-fine strategy for multiclass shape detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(12):1606–1621, 2004.
- [2] H. Bay, T. Tuytelaars, and L. J. Van Gool. SURF: Speeded up robust features. In *European Conference on Computer Vision*, pages 404–417, 2006.
- [3] V. Ferrari, T. Tuytelaars, and L. Gool. Simultaneous object recognition and segmentation from single or multiple model views. *Int. J. Comput. Vision*, 67(2):159–188, 2006.
- [4] D. Forsyth, J. L. Mundy, A. Zisserman, C. Coelho, A. Heller, and C. Rothwell. Invariant descriptors for 3d object recognition and pose. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(10):971–991, 1991.
- [5] C. Harris and M. Stephens. A combined corner and edge detection. In *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151, 1988.
- [6] N. K. J. Fauqueur and R. Anderson. Multiscale keypoint detection using the dual-tree complex wavelet transform. In *IEEE International Conference on Image Processing (ICIP'2006)*, 2006.
- [7] V. Lepetit, P. Laguerre, and P. Fua. Randomized trees for real-time keypoint recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 775–781, 2005.
- [8] D. G. Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision*, pages 1150–1157, 1999.
- [9] D. G. Lowe. Local feature view clustering for 3d object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume I, pages 682–688, 2001.
- [10] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool. A comparison of affine region detectors. *Int. J. Comput. Vision*, 65(1-2):43–72, 2005.
- [11] S. K. Nayar, M. Watanabe, and M. Noguchi. Real-time focus range sensor. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(12):1186–1198, 1996.
- [12] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. 3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. *Int. J. Comput. Vision*, 66(3):231–259, 2006.
- [13] S. Ullman and R. Basri. Recognition by linear combinations of models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(10):992–1006, October 1991.
- [14] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 511–518, 2001.