A Deformation Tolerant Version of the Generalized Hough Transform for Image Retrieval

Marco Anelli[1] and **Alessandro Micarelli**[2] and **Enver Sangineto**[1]

Abstract. We propose a new version of the famous Ballard's Generalized Hough Transform (GHT) for image retrieval by shape similarity. Indeed, the GHT is a very powerful pattern recognition technique, robust to noise and occlusion situations, utilized in hundreds of different machine vision problems. Nevertheless, it is conceived for an *exact* matching between the model and the input image, while, in image retrieval, the user description of a figure is inherently abstract and approximate, thus *locally different* from each data base image. In this paper we present a version of the GHT locally tolerant to deformations which successfully fits image retrieval peculiarities without accuracy loss. The proposed method has been implemented and tested using images randomly chosen from the Web with very good experimental results.

1 MOTIVATIONS AND GOALS

Image retrieval is an emerging and interesting area of research and application that has grown very quickly in these last twelve years. It is based on the idea of retrieval of visual information (such as still images) by using a user visual query. In case of retrieval by shape similarity, the query usually is a stylized sketch drawn by the user to specify the shape features she/he is interested to find in the images of the system data base. In a recent survey on content based image retrieval [12], the authors give an overview of the main methods and techniques for image retrieval by shape similarity. Most of them can be classified in three main approaches: *statistic techniques*, *elastic template matching* and *multiscale representations*.

The former is a very common machine vision approach, consisting in the extraction from an image of a set of n pre-fixed statistical features. Each feature i ($i = 1, .., n$) is expressed by means of a real value (v_i), hence the whole image can be represented through a point $(v_1, ..., v_n)$ in a n-dimensional space. If we code both the user sketch and the data base images with a set of points in $\mathbb{R}^n$, shape similarity can easily be defined by means of a distance measure (for instance, the Euclidean distance). A lot of commercial and research prototype image retrieval systems are based on this principle, the most famous among them is QBIC [5].

The main advantage of this technique is the possibility of a quick indexing of large-size visual data bases, possibly organized exploiting the spatial proximity in $\mathbb{R}^n$ of their images' representations. Indeed, feature space representation is the only approach which allow data base indexing. All the other methods need to sequentially process all the repository images to match them with the user sketch. Unfortunately, statistical measures of an object bi-dimensional appearance (such as its *area*, *compactness*, *elongatedness* and so on) only allow for poorly accurate descriptions of the object shape; thus these techniques cannot be applied to domain independent repositories without loss in retrieval recall and precision rates. Moreover, they are occlusion sensitive, because partial occlusions (or adjacent objects) modify the silhouette of an object, and consequently all global statistical features defined on it.

A more accurate approach is elastic template matching. The idea is based on the progressive deformation of the user sketch until it matches the image. The sketch S is overlapped to each image I of the data base and it is iteratively locally modified trying to minimize the matching function $M_{I,S}(O, D)$, where O expresses the overlapping rate between S and I and D is the deformation energy. $M_{I,S}(O, D)$ gives the similarity measure between S and I: the larger is O, the larger is $M_{I,S}(O, D)$, but the larger is D, the smaller is $M_{I,S}(O, D)$. Jain et al. in [7] and Del Bimbo and Pala in [3] present some working examples exploiting this idea, with different formalizations of the function M.

Elastic template matching allows one to take into account the user sketch shape in a more accurate way with respect to its representation by means of a set of summarizing statistical features. Furthermore, it is robust to occlusions. Nevertheless, often the iterations needed to minimize M make the technique quite slow. In [7], for example, an image is processed up to 12 seconds. Otherwise, it is necessary to manually select the portion of I in which S is initially overlapped. Indeed, elastic template matching is not translation invariant (nor rotation or scale invariant)and the presence of a non-uniform background in I can bring S deformations to be attracted by background lines.

The multiscale representation approach [4] is based on a multiple representation $M(s)$ of a given object O, in which O is represented by different scale values (s). The bigger is s the greater is the number of details of O represented in $M(s)$. By changing the scale factor we can change the abstraction level, and this seems to be an important pre-attentive human vision mechanism. $M(s)$ is usually obtained from O by extracting from its silhouette the curvilinear abscissa which is then progressively smoothed. This method can be very fast, because, at a given scale value, we have to match only few elements for two given images. Nevertheless, it is specially suitable for isolated objects (without occlusions and with uniform backgrounds) [9]. Indeed, curvilinear abscissa extraction needs a previous segmentation of O external contours from the processed image. From this point of view, the multiscale approach has some advantages and disadvantages in common with the statistical one, and indeed $M(s)$ is often represented by means of a feature vector.

In this paper we propose a modified version of the Generalized Hough Transform (GHT) for image retrieval by shape similarity is-

[1] Centro di Ricerca in Matematica Pura e Applicata (CRMPA), Sezione "Roma Tre". Via della Vasca Navale 79, 00146 Roma, Italia.
[2] Dipartimento di Informatica e Automazione (DIA), AI Lab, Università degli Studi "Roma Tre". Via della Vasca Navale 79, 00146 Roma, Italia.

sues. The main reason that induced us to consider the GHT for image retrieval is its robustness to occlusions, noise and non-uniform backgrounds. It does not need previous manual segmentation and can be applied to generic images without accuracy loss. The GHT was initially proposed by Ballard [1] as a generalization of the original Hough Transform (HT). It is a very powerful technique for general-purpose image identification and it can be essentially viewed as an efficient and elegant implementation of the (rigid) template matching, capable to identify the known model M of an object shape in an image I. It is very robust to partial occlusions, noise and non-uniform backgrounds: indeed, a *voting* mechanism allows to automatically select in I the most probable position for M. In [10] the HT is used to extract the main straight lines of an image in order to classify it by its orientation. Concerning the GHT, it has never been applied to image retrieval up to now, and the reason is that it performs an *exact* template matching between M and I. Conversely, in image retrieval the user sketch S and the image I usually do not exactly overlap because the user only has an approximate idea of what is looking for when draws S. Jain et al., in [7], state that:

> "The Hough Transform method can be viewed as template matching. However, it is a rigid scheme in that it is not capable of detecting a shape which is *different* from the template by transformations other than translation, rotation and scalability."

Hence, if we want to apply the GHT to image retrieval, the first problem we have to deal with is its extension to a *deformation tolerant* template matching. We propose a solution to this problem allowing the voting mechanism to increment a range of values in the Hough accumulator (A) instead of a single element. Conceptually, this means that the related positions of the points of the template M (the user sketch) can locally vary in the selected image I. Finally, we adopt a different way to select local maxima in the accumulator A in order to avoid false retrievals due to the augmented voting area.

The rest of this paper is organized as follows. In Section 2 we explain in details our proposed method, while in Section 3 we show the results of our experiments and we conclude in Section 4.

2 MODIFYING THE GHT

We give here a rapid overview of the GHT, referring to the original paper [1] for details and to [6, 8] for two exhaustive surveys on HT versions. Given a model M and a reference point p_r in the M Chartesian representation, the first step of the GHT is to construct the *R-Table* T. For each edge point p of M we store in T the vector $v = p_r - p$. For efficiency reasons, T is indexed by using p's gradient direction. In the second step, given an image I to recognize, for each point p of I we increment the Hough accumulator $A[x, y]$ if there exists a vector $v \in T$ s.t. $v = p_r - p$ and $p_r = (x, y)$. We look for v in T by utilizing p's gradient direction. The last step consists in finding local maxima of A: they are the most likely positions of M in I. As we can see, no a priori segmentation is needed, because the original reference point p_r is found in I looking for a pick in A given from the intersection of a consistent number of votes.

A locally deformation tolerant version of the GHT (DTGHT from now on) for image retrieval is composed of the following steps.

First of all, we use the user sketch S as a model to build the R-Table T. We do not index T by means of the gradient direction of the points in S. Indeed, due to local dissimilarities between S and a generic image I of the data base, we usually expect that a point p in S and a corresponding point p' in I are differently oriented. Thus, T is built according to the following algorithm:

```
R-Table Construction(S)
1 Compute the centroid $p_r$ of $S$.
2 For each point $p_i$ ($i = 1, ..., m$) of $S$, set: $T[i] := p_r - p_i$.
```

The second step of the DTGHT consists of a series of pre-processing algorithms applied to each image I of the data base. These low-level modifications of I are independent by S and can be done off-line, when I is included in the data base. In the first pre-processing step we apply the Crimmins filter [2] to the gray value image. This is a filter used to reduce noise. Performing a few iterations with this filter we enhance the contour pixels weakening the texture ones. Afterwards we perform a standard edge extraction and thinning process by using the Canny edge detector with Sobel 3×3 masks [11] (see Figure 1 and 2). From now on, we indicate with I the edge map of a generic image of the system data base.

After this standard pre-processing, we propose two *texture filters* to erase those thick textured areas which deteriorate the retrieval process. The two filters work in this way. From each edge pixel p of I let $C(p)$ be a squared mask of $n_1 \times n_1$ pixels centered in p. Furthermore, let N be the number of edge pixels in $C(p)$, $\phi(p)$ the orientation of the edge pixel p ($\phi(p) \in [0, 2\pi[$) and σ^2 the variance of all the edge pixels' orientation in $C(p)$:

$$\sigma^2 = (\sum_{p' \in C(p)} \frac{\phi(p')^2}{N}) - (\sum_{p' \in C(p)} \frac{\phi(p')}{N})^2. \tag{1}$$

Then, we cancel the edge pixel p from the image I if:

$$N > th_1 \wedge \sigma^2 > th_2, \tag{2}$$

where th_1 and th_2 are two pre-fixed thresholds ($th_1 \in [0, n_1^2]$, $th_2 \in [0, 4\pi^2[$). Therefore, if, for a given edge pixel p, we have that in its $n_1 \times n_1$ neighbor $C(p)$ there are more then th_1 edge pixels with considerable orientation entropy (σ^2), we consider p as part of a (disordered) textured area and we delete it.

For the second filter we define: N' as the number of edge pixels $p' \in D(p)_{n_2 \times n_2}$ s.t. $\phi(p') = \phi(p)$ and we cancel p from I if $N' > th_3$, where $th_3 \in [0, n_2^2]$ is an empirical threshold. This second filter aims to delete those edge pixels p s.t. in a $n_2 \times n_2$ neighbor $D(p)$ of p there are more then th_3 edge pixels with the same orientation as p. This is the case of shadows or particular ordered textures which produce in edge extraction multiple parallel lines that do not add further shape information.

Presently we empirically set: $n_1 = 40$, $n_2 = 20$, $th_1 = 260$, $th_2 = 0.165$, $th_3 = 120$. Figures 1, 2 and 3 show an example of an image before and after the pre-processing phases.

The third step of the DTGHT is the voting phase. Given an image I and a user sketch S represented by means of the offset vectors of the R-Table T, we want to find in I the most likely positions of S allowing local deformations for the S shape. To achieve this objective, we have to modify the GHT voting mechanism incrementing all those accumulator's positions $A[x, y]$ in a range centered in p_r, for each edge point p of I and each vector $v = p - p_r$ in T. This is efficiently achieved by means of the following algorithm, in which we use two accumulators A and B both of the same dimensions of the image I.

```
Voting Procedure(I,T)
1 The accumulators $A$ and $B$ are set to 0.
2 For each edge point $p$ ($p = (x, y)$) of $I$ do:
3     For each $i$ ($i = 1, ..., m$) do:
```

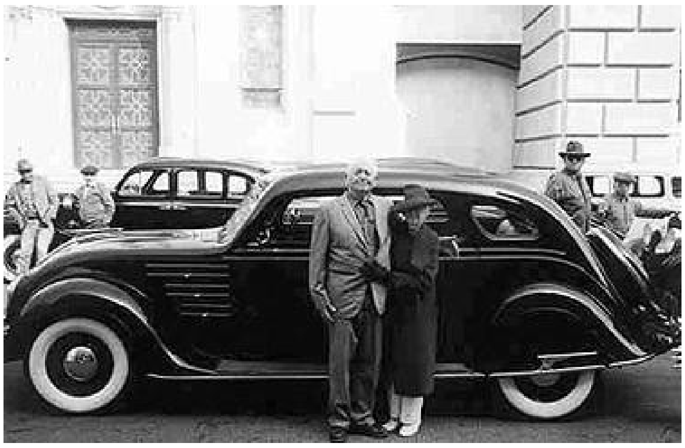

Figure 1. A gray-level image of a car.

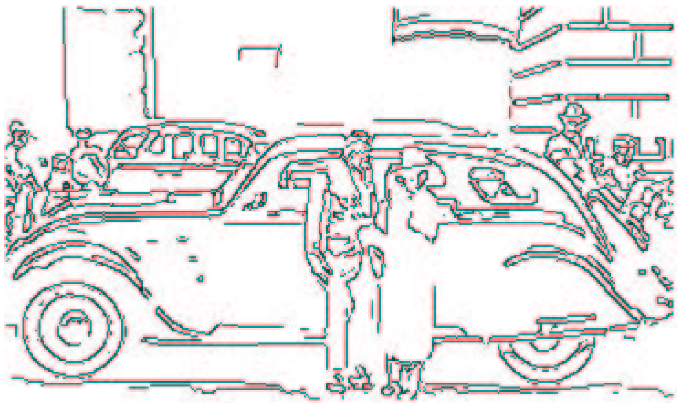

Figure 2. The edge map extracted from Figure 1.

```
4         Let $T[i] = (x_i, y_i)$.
          Then $A[x + x_i, y + y_i] := A[x + x_i, y + y_i] + 1$.
5 For each point $(x, y)$ of $B$ do:
6     For each $(x_1, y_1)$ s.t. $|x - x_1| <= l$, $|y - y_1| <= l$, do:
7         $B[x, y] := B[x, y] + A[x_1, y_1]$.
```

Let us call Steps 1-4 of the above procedure the *exact voting* phase and Steps 5-7 the *spreading* phase. The main difference between the GHT and our proposed DTGHT is given by the spreading phase. A point $p = (x, y)$ in the final accumulator B (and thus also the maximum value in B) receives votes by all those edge points $p' = (x', y')$ in I s.t., for some $i \in [1, m]$, $T[i] = (x_i, y_i)$, and:

$$|x + x_i - x'| \leq l, \tag{3}$$

$$|y + y_i - y'| \leq l, \tag{4}$$

where l is a pre-fixed value indicating the half side of a squared tolerance window centered in $(x + x_i, y + y_i)$. This tolerance window allows S to be locally deformed by means of non-rigid translations of its points. Presently, we have $l = 10$.

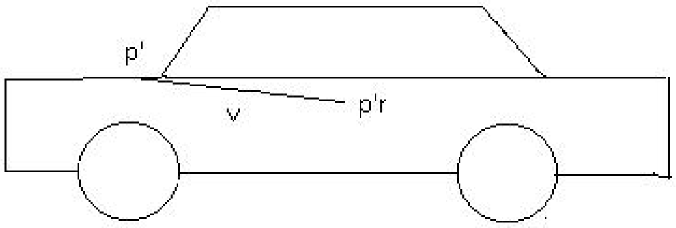

Figure 3. A user sketch of a car and an element (v) of the R-Table.

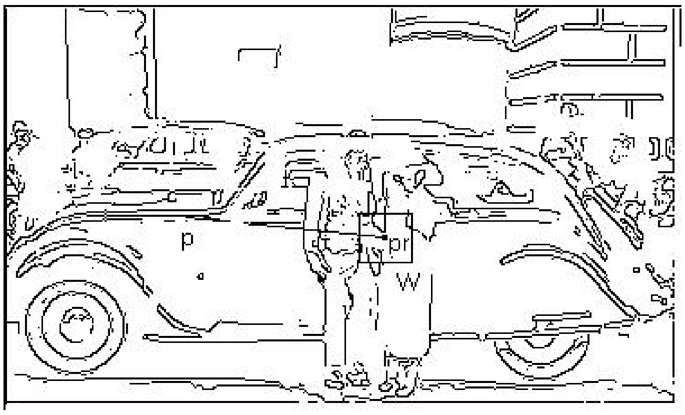

Figure 4. The edge map of Figure 2 after the texture filters' application and an example of voting window (W).

In Figure 3 a point p' on the user sketch S contributes to the R-Table construction with the vector v (for some i, we pose: $T[i] = v$). In Figure 4 an edge point p of I, by using the ith element of T, votes in p_r in A accumulator and in the whole window $W(p_r)$ in B final accumulator. We call $W(p_r)$ the *voting window* centered in p_r and with fixed side length $2l + 1$.

It is worth noting that the operations performed by the spreading

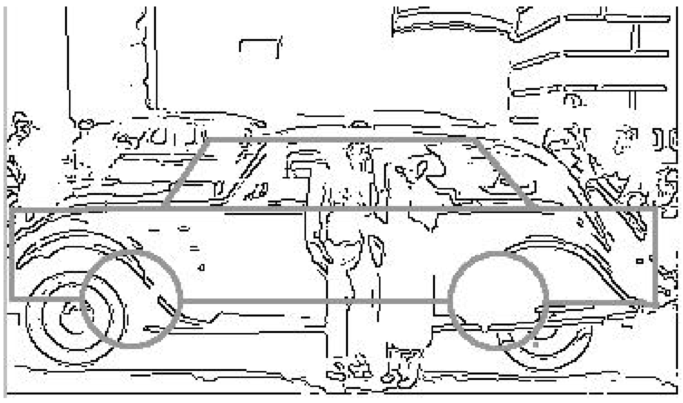

Figure 5. The user sketch superimposed to the edge map after the global maximum detection.

phase in the above algorithm could conceptually be done in Step 4 directly incrementing all the elements $A[p_1]$ s.t. $p_1 \in W(p_r)$ ($p_r = p+T[i]$). Nevertheless, Step 4 is nested in Step 2 and Step 3 loops and this second solution would make dramatically worse computational complexity and execution times.

The final step of the DTGHT concerns maxima detection in B accumulator and data base images' ranking according to the user sketch. We have modified the traditional HT local maxima detection taking into account the new voting mechanism. Let M be an array of the same dimensions as B. If p is a point in B accumulator and $W(p)$ the voting window centered in p, we define $W'(p)$ to be a mask of $(2l'+1) \times (2l'+1)$ elements of B centered in p and with $l' > l$ (presently we empirically set $l' = 50$) and $F(p) = W'(p) - W(p)$. Finally, if N is the number of elements of $W(p)$ ($N = |W(p)|$) and N' is the number of elements of $F(p)$ ($N' = |F(p)|$), we define the average voting value in p ($av(p)$) and the average voting value in p's neighbor ($nav(p)$) respectively by:

$$av(p) = \frac{\sum_{p' \in W(p)} B[p']}{N}. \tag{5}$$

$$nav(p) = \frac{\sum_{p' \in F(p)} B[p']}{N'}. \tag{6}$$

With the above definitions, we associate to each image I of the system data base a *similarity degree* $DTGHT(S, I)$ with the user sketch S computed by using the following algorithm.

```
Maxima Detection(B)
1 For each element $p$ of $B$ do:
2     Compute $av(p)$ by means of (5).
3     Compute $nav(p)$ by means of (6).
4     $M[p] := av(p) - nav(p)$.
5 $DTGHT(S,I)$ is the maximum value in $M$.
```

Images $I_1, I_2, ...$ of the data base are ranked according to their similarity degree with respect to S and then proposed to the user.

Figure 5 shows the system output with respect to the image of Figure 1. Figure 6, instead, shows an example in which the system proposes to the user the first ten ranked images it retrieved according to the given sketch (in this case, the profile of a quadruped).

The above presented maximum detection can be viewed as a sort of extension of the traditional local maxima detection in Hough accumulators. Indeed, in HT techniques, a mask C is overlapped to the accumulator to look for the points m s.t., for each point $p \in C$, $B[m] \geq B[p]$. Conversely, we use two masks, W and W' because our voting area is now augmented (from a point to a squared window W). Indeed, initial experimental trials in which we simply posed $DTGHT(S,I) = B[m]$ (m being the global maximum in B) brought to a greater number of false retrievals with respect to the present evaluation method. The reason is that some photos with objects containing large areas with dense noise or textures (such as bushes, trees with thick leaves, ...) can produce a great number of false votes with a global maximum in B not corresponding to the user sketch shape.

This situation can be avoided (or at least strongly limited) in two ways. The first is by using some pre-processing filters to delete thick texture areas after the edge extraction and thinning phase. We described these filters in the second step of the DTGHT.

The second way is to evaluate the results of the accumulator B not only with respect to the value of its maximum $B[m]$ but also taking into account the average value of the vote in the neighborhood of m outside the voting window. Indeed, if in I there is a thick texture or noise area, it will produce in B a corresponding area E with high votes. But the entropy of the noise will produce a rather uniform distribution of the votes in E, so that we can not isolate in it a point considerable higher than its neighboring. On the other hand, a correct presence of S in I will concentrate its votes in a restrict area (not much greater than $W(m)$), and this can be detected by subtracting $nav(m)$ from $av(m)$.

Before concluding this section, we remark some features about the DTGHT. It is translation invariant and does not need segmentation. Moreover, it can tolerate deformations fixed by the parameter l (from (3) and (4)). However, like the original GHT, it is not rotation nor scale invariant. To cope rotation and scale changing factors we have to iterate Steps 3 and 4 of the transform for discrete values of rotation and scale. Nevertheless, it must be noted that the deformation tolerance allows scale invariance for differences less or equal to l, and, for foreground searched objects, few iterations (may be 4-5) cope almost all situations. Regarding rotation, we are studying the possibility to use the edge orientation information to make the method rotation invariant. Nevertheless, in an image retrieval domain, we can suppose the user usually draws its sketch with the right object orientation (an "horizontal" car, and so on).

3 EFFICIENCY AND RESULTS

In Step 2, 3 and 4 of the DTGHT we use squared masks to compute some values from an original bi-dimensional array. This operation can be computational optimized by computing, for each row of the array, only the first mask and then deriving the others by the last one just computed with a dynamic programming technique. For example, regarding the spreading phase, we observe that:

$$B[x,y] = \sum_{p_1 \in W(x,y)} A[p_1], \tag{7}$$

if $x = l$ (we suppose to compute the spreading phase for only those points in B far at least l from B's boarders), and:

$$B[x,y] = B[x-1,y] - \sum_{p_1 \in W_1(x-1,y)} A[p_1] + \sum_{p_1 \in W_{x+l}(x,y)} A[p_1], \tag{8}$$

if $x > l$; where $W_i(p)$ means the i-th column of the voting window $W(p)$.

If we suppose N_1 to be the size of the array ($|I| = |S| = N_1$) and l_1 the side of the mask, the above algorithm is $O(N_1 l_1)$. To compute each DTGHT step computational cost, in the following we will use this result together with the assumption that n is the number of edge pixels of I and m the number of edge pixels of S.

The second Step of the DTGHT is composed of low-level filters. The standard ones are $O(n)$. Texture filters are $O(N_1 n_1)$, n_1 being the side of the used masks. Moreover, they can be applied off-line, at each image acquisition. As a consequence, the whole computational (on-line) cost of the DTGHT algorithm is given by the sum of Steps 1, 3 and 4.

Step 1 is linear with the number of edge pixels of S ($O(m)$). In Step 3 the exact voting phase is similar to the original GHT and has the same computational cost $O(nm)$, given by its two nested loops. The spreading phase, instead, has a computational cost of $O(N_1 n_2)$, where $n_2 = 2l+1$ is the size of the voting window. Finally, Step 4 is $O(N_1 n_3)$, where $n_3 = 2l'+1$. Therefore, the whole computational cost of the DTGHT is given by:

$$O(m + nm + N_1 n_2 + N_1 n_3) = O(nm + N_1 n_3). \tag{9}$$

Typical values for n, m, N_1 are: $n = 7000$, $m = 1500$, $N_1 = 90000$, which make $O(nm)$ and $O(N_1 n_3)$ of the same magnitudo order. From this and from (9) follows that the DTGHT complexity order is:

$$O(nm). \tag{10}$$

This result shows that the original GHT and our modified version DT-GHT have the same computational cost. Indeed, the spreading phase allows one to deal with non-rigid transformations of the template in the image but does not affect the whole consuming time.

We implemented our method in Java 1.3 and tested it on a Pentium III, 850 Mhz, 192 MB RAM. All our trials (a few hundreds) has been executed in less than 1 second (0.3 seconds on average) with images from 200×200 up to 380×350 pixels. The system data base is composed of 225 images randomly taken by the Web. They show a great variety of subjects: beside the ones shown in Table 1, we have included crucifixes (10), vases (10), animals, faces, landscapes, airplanes, boats, fruits, mushrooms, various cups, a picture of the Tour Eiffel, an ice cream photo,... and so on. No simplifying assumption was made about images: they have non-uniform backgrounds and often the retrieved objects are occluded by other objects. Also lighting conditions and noise degree are not fixed: we simply took our images randomly browsing the Web.

The user sketches used for experimentation (such as the car and the quadruped shown in this paper) have been drawn by a person not aware of the data base images' shapes: we only manually fix the sketches' approximate scale because the method is not scale invariant and we still not perform iterations to cope different scale values. Table 1 summarizes the results regarding the first 10 retrieved images for each input sketch. The second column shows the number of objects of a given type (first column) in the data base, while the third the correct retrievals in the first 10 positions.

Table 1. Correct retrievals in the first 10 positions.

Object type	Items in the data base	Ranked in 1-10
Car	24	10
Guitar	17	10
Saxophone	14	9
Horse	18	8
Pistol	10	10
Tennis racket	10	8
Bottle	11	7
Watch	12	9

As future work we are planning a more systematic testing of the DTGHT using benchmark data bases such as the Columbia COIL data set or the MPEG-7 shape silhouette data base.

4 CONCLUSIONS

In this paper we have presented a modified version (DTGHT) of the GHT for deformable template matching. We have shown that is possible to augment the GHT voting area without loss in accuracy nor in computational complexity. The larger voting area allows the user sketch to change in the compared image, thus the method is specially suitable for image retrieval by shape similarity issues.

With respect to statistic and multiscale methods (Section 1), our proposed DTGHT is not suitable for data base indexing. Nevertheless, it does not need a previous (often not automatic) segmentation of the objects' silhouettes. Conversely, with respect to elastic template matching techniques, the DTGHT also takes into account the user sketch shape in an accurate way but does not need initial setting of the template on the object appearance in the image, nor needs time-consuming iterations to minimize the matching function (see Section 1).

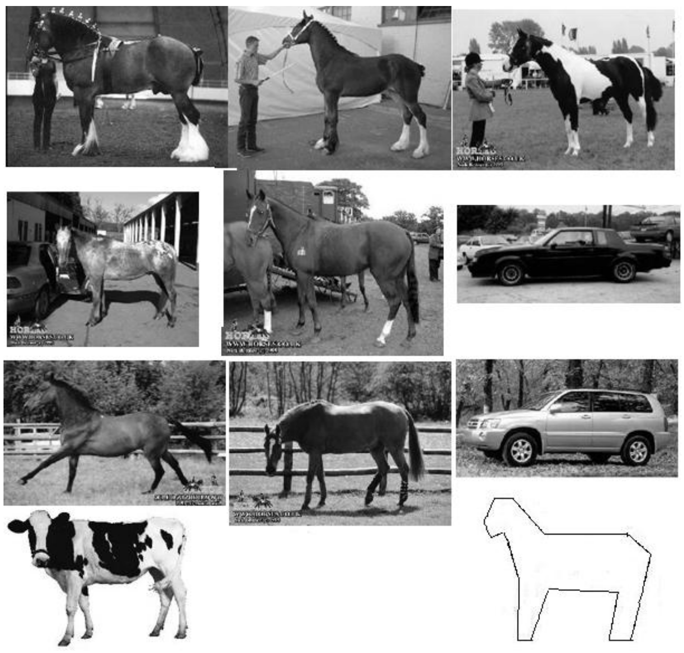
Figure 6. The first 10 images ranked by the system according to the sketch of a quadruped (in the bottom-right corner).

ACKNOWLEDGEMENTS

We want to thank Virginio Cantoni for the interesting and stimulating discussions about Hough Transform techniques.

REFERENCES

[1] D H Ballard, 'Generalizing the Hough Transform to detect arbitrary shapes', *Pattern Recognition*, **13. No. 2**, 111–122, (1981).
[2] T Crimmins, 'The geometric filter for speckle reduction', *Applied optics*, **24. No. 10**, (1985).
[3] A Del Bimbo and P Pala, 'Visual image retrieval by elastic matching of user sketches', *IEEE Trans. on PAMI*, **19. No. 2**, 121–132, (1997).
[4] A Del Bimbo and P Pala, 'Shape indexing by multi-scale representation', *Image and Vision Computing*, **17**, 245–261, (1999).
[5] M. Flickner et al., 'Query by image and video content: the QBIC system', *IEEE Computer*, **28. No. 9**, 23–32, (1995).
[6] J Illingworth and J Kittler, 'A survey of the hough transform', *Computer Vision, Graphics and Image Processing*, **44**, 87–116, (1988).
[7] A. K. Jain, Y. Zhong, and S. Lakshmanan, 'Object matching using deformable templates', *IEEE Trans. on PAMI*, **18**, 267–278, (1996).
[8] V F Leavers, 'Survey: which hough transform?', *Computer Vision, Graphics and Image Processing*, **58**, 250–264, (1993).
[9] F Mokhtarian, 'Silhouette-based isolated object recognition through curvature scale-space', *IEEE Trans. on PAMI*, **17. No 5**, 539–544, (1995).
[10] E Di Sciascio and A Celentano, 'Similarity evaluation in image retrieval using the Hough Transform', *Journal of Computing and Information Technology*, **4. No. 3**, (1996).
[11] L Shapiro and G. Stockman, *Computer Vision*, Prantice hall, 2001.
[12] A W M Smeulders, M Worring, S Santini, A Gupta, and R Jain, 'Content-based image retrieval at the end of the early years', *IEEE Trans. on PAMI*, **22. No 12**, 1349–1380, (2000).