Available online at www.sciencedirect.com

ScienceDirect

Image and Vision Computing 27 (2009) 87–98

www.elsevier.com/locate/imavis

A hierarchical non-parametric method for capturing non-rigid deformations ☆

Ady Ecker [a,*], Shimon Ullman [b]

[a] *University of Toronto, Canada*
[b] *Weizmann Institute of Science, Israel*

Received 10 November 2005; received in revised form 30 September 2006; accepted 20 October 2006

Abstract

We present a novel approach for measuring image similarity based on the composition of parts. The measure identifies common sub-regions between the images at multiple sizes, and evaluates the amount of deformation required to align the common regions. The scheme allows complex, non-rigid deformation of the images, and penalizes irregular deformations more than coherent shifts of larger sub-parts. The measure is implemented by an algorithm which is a variant of dynamic programming, extended to multi-dimensions, and is using scores measured on a relative scale. The similarity measure is shown to be robust to non-rigid deformations of parts at various positions and scales, and to capture basic characteristics of human similarity judgments.
© 2008 Published by Elsevier B.V.

Keywords: Image similarity; Non-rigid deformations; Relative dynamic programming; Overlapping patches

1. Introduction

Current approaches for measuring images similarity usually consist of two components. The first is based on feature similarity, using features extracted from both images. The features can range from simple templates to complex and more abstract features such as SIFT features [3] or shape context [4]. The second component is based on the amount of distortion required to bring the two images into alignment. This is usually done using a linear transformations or more complicated parametric transformations [4]. In this paper we deal with the second component, in particular, the question of how to capture non-rigid deformations between images and how the deformations affect the similarity judgment. Fig. 1 illustrates the sort of transformations we would like to be able to extract and explain. At the bottom image of Fig. 1, (a) stays in the same place, (b) moves rightwards and part (c) upwards. The similarity between the top and bottom images is apparent for humans. With more subdivisions of the image, this type of transformations can be made arbitrarily complex, since different sub-images undergo different displacements. This cannot be adequately captured by a parametric transformation. Although the last example is artificial, it is clear that current methods do not cover the range of image pairs that human would consider structurally similar. For instance, a hand with some longer or shorter fingers will still be considered a hand by humans. One can deal with such variability by building specialized hand-detector. However, no matter how strong the feature detectors are, there will always be cases where the similarity is established from the spatial layout of the features. While many methods are designed to be robust to small non-rigid deformations encountered in practice, we are interested in a method that can explain in a principled manner a much wider range of non-rigid deformations of image patches.

1.1. Related work

The method developed in this work is rooted in dynamic programming methods for matching strings, such as edit

☆ This work is based on a M.Sc. thesis [1]. A conference version of the paper appeared in [2].
* Corresponding author.
E-mail addresses: adyecker@cs.toronto.edu (A. Ecker), shimon.ullman@weizmann.ac.il (S. Ullman).

0262-8856/$ - see front matter © 2008 Published by Elsevier B.V.
doi:10.1016/j.imavis.2006.10.006

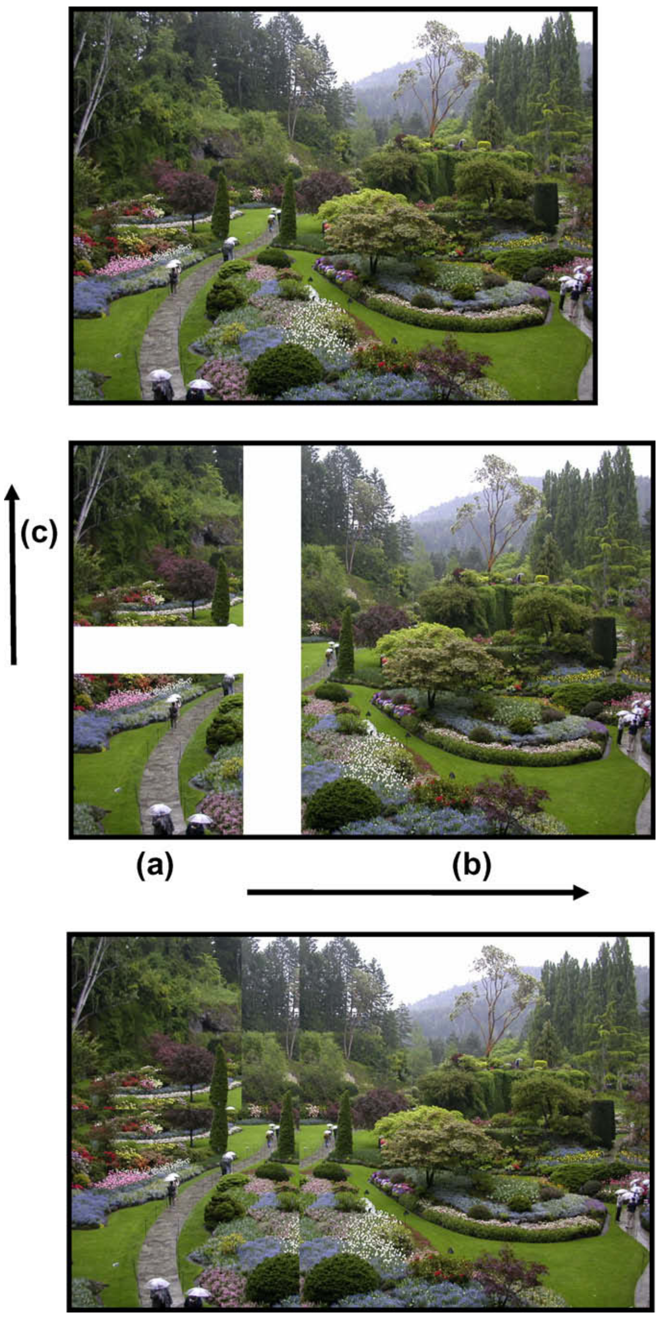


Fig. 1. The middle image was obtained by shifting patches (b) (rightwards) and (c) (upwards). The bottom image was obtained by overlaying the middle image on the top one. While the bottom and top images are similar, no parametric transformation can explain this.

distance [5]. The main difficulty in applying the dynamic programming paradigm directly to the problem at hand is that two dimensional images have no natural scanning direction. For example, it will not be useful to match the two images in Fig. 1 by matching them in a fixed one-dimensional scan, by rows or by columns. There have been several previous attempts to use dynamic programming to evaluate similarity of images. In [6], the longest common subsequence algorithm was used over the columns of a matched template. In [7] and [8], shape contours were aligned using dynamic programming. In [9], shapes were represented by shock graphs, and a specialized edit distance algorithm was developed to align two skeleton-trees in minimum cost of tree-transition operations. In [10], a polygon triangulation was matched by dynamic programming using an elimination order for the triangles. In [11], the problem of matching parts-based model to an image is considered. The cost of a match is based both on the similarity between the parts to the image and on pairwise geometric relationships between the parts. An approximation to a dynamic programming procedure is developed to make the problem tractable. An exact solution is possible in case the graph representing the relationships between the parts is a tree [12].

Our method differs substantially from the graph-based approaches mentioned in that it works entirely on a two dimensional grid. To do so, it became necessary to abandon a key property of the dynamic programming paradigm, the definition of a global cost function. Instead, we apply iterative steps that optimize a hierarchical objective function. In order to interpret the similarity scores produced by the algorithm as it compares different image patches, we normalize this score, using the empirical distribution of scores generated by the algorithm for a class of images.

Our algorithm is based on matching sub-patches of the two images. Intuitively, two similar images consist of similar subparts. As noted in [13], the converse also holds: if two images have many similar overlapping shape fragments, then with high probability the images are similar. It is highly unlikely that overlapping patches will be arranged in two configurations that are totally dissimilar. The consequence is that relations among the parts within each image can be ignored, since they are captured indirectly by comparing overlapping patches. Similar ideas appear in [14]. Related work in computer graphics (e.g. [15]) demonstrated patch-based method to be highly effective for texture synthesis. Recently, there have been several successful demonstrations of patch-based methods for segmentation and recognition employing statistical learning from examples [16,17]. However, the problem of systematically identifying the transformations of similar sub-patches in images has not been studied previously.

2. The relative dynamic programming algorithm

In this section we present our similarity-measuring algorithm, which we call RDP (Relative Dynamic Programming) for reasons explained below. We describe an implementation of RDP that measures the similarity between grayscale images. Our aim is to identify patches from the two images as similar if they contain many overlapping similar sub-patches. The question whether subparts are similar is therefore recursive. This leads to an iterative, bottom-to-top construction. The algorithm starts by matching small patches, using a basic similarity measure, and computes the similarity of larger patches based on the similarity of the smaller ones contained in them. The similarity values between patches are stored in a table. At each phase, the algorithm fills the table iteratively, based on the table computed in the previous phase. The algorithm is a dynamic programming algorithm in the sense that it re-uses optimal solutions already obtained for sub-problems.

In contrast to standard dynamic programming algorithms, the presented algorithm is extendible to more than one dimension. This does not simply mean that we use a multidimensional table. The crucial distinction is that in our method image blocks can move in multidimensional directions during the alignment process. The key that enables flexible movements in multiple dimensions is the fact that our algorithm builds the tables starting from many initial points simultaneously, whereas standard dynamic programming algorithms start from a single point and require a scan order. This advantage comes at the price that the objective function computed by the algorithm is hierarchical, i.e. cannot be explicitly written down as a function of the two images in a short way. We don't start by stating a global objective function and formulating the iterative steps to optimize this function. We start by formulating the iterative steps in a way that the resulting hierarchical function the algorithm computes will be robust to small movements of subparts.

As a result, similarity scores have to be interpreted relatively. The meaning of a score comes from comparison to other scores produced by the same algorithm on other inputs. Specifically, the score is the probability to obtain lower values than the value computed by the similarity computation. Note that even for standard dynamic programming algorithms, such as edit distance [5], the interpretation of the score depends on some assumptions on the distribution of these scores. For example, suppose we compare two strings and find that 5 edit operations are required. The number "5" is not informative unless we know how frequent it is in the domain of these strings.

2.1. Algorithm details

For expository reasons, we start by describing a simplified version of the algorithm in one dimension. The input consists of two rows of N pixels, $I_1(0..N-1)$, $I_2(0..N-1)$, where each pixel has grayscale value between 0 and 1. For simplicity assume that N is a power of 2. A hierarchical alignment of the sort we consider is shown in Fig. 2. At level h, where $h = 1,2,\ldots,\log(N)$, the alignment matches sub-patches of size 2^h taken from both images. The patches are equally spaced 2^{h-1} pixels apart, i.e. the distance between patches is doubled as the level increases. We allow sub-patches to shift one position (2^{h-1} pixels) with respect to their parent patch. In Fig. 2, patches of two pixels can match by moving up to one pixel with respect to the four-pixel patch that contains them, and patches of four pixels can move in steps of two pixels. With each hierarchical alignment we associate a hierarchical

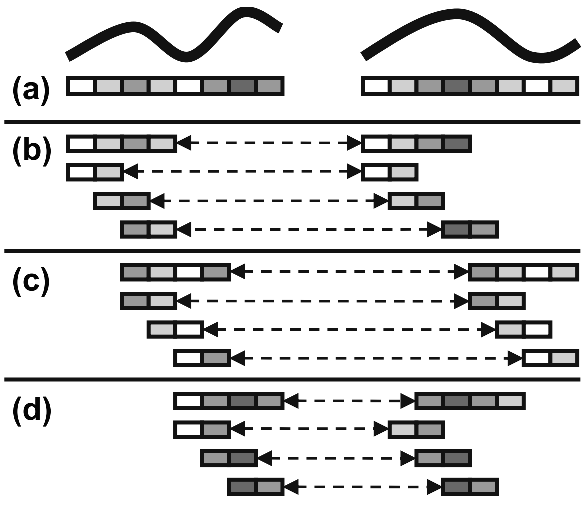


Fig. 2. Hierarchical alignment of 1D images. (a) The two 8-pixels images being compared. (b, c, and d) The best matches for the 4-pixels sub-patches of the left image based on the movements of the 2-pixels sub-patches.

score. The smallest patches are compared using some basic similarity function. At higher levels patches are compared by combining the similarity scores between their sub-patches. A penalty is added whenever the alignment chooses to shift a sub-patch relatively to its parent patch. The goal is to compute the minimal hierarchical score of hierarchical alignments of this type. Note that the hierarchical score is not a simple global objective function of the two images, but has to be computed from the hierarchical alignment. However, the hierarchical score is useful for ranking similarities of images.

Next we describe the algorithm in more detail. The algorithm proceeds in phases. In phase h, the algorithm computes the similarity score between sub-patches of size 2^h. The final output is the similarity score of phase $\log(N)$, when the patch size equals N.

Intermediate scores are stored in a similarity table for use in the subsequent phase. We assume that each pixel in I_1 cannot move more than Δ pixels to its "true location" in I_2. The similarity table $T(i,\delta)$ stores all possible comparisons between patches whose centers are no more than Δ pixels apart. More specifically, $T(i,\delta)$ is the similarity value between the patches whose left corners are $I_1(i)$ and $I_2(i+\delta)$. T has pixel index i ranging from 0 to $N-1$ and displacement (offset) index δ ranging from $-\Delta$ to Δ (indexing based on the relative offset between patches is a common compact way to store matrices of small bandwidth. This representation is extendible to multidimensional arrays). The structure of the table is illustrated in Fig. 3. Some of the entries in the similarity table are empty due to boundary overflows. From here on we will ignore such details which are handled in the implementation.

The first phase is executed using a basic similarity function D applied to the intensity values of two 2-pixels patches. Here D is a distance (or dissimilarity) function, meaning that values close to zero reflect higher similarity.

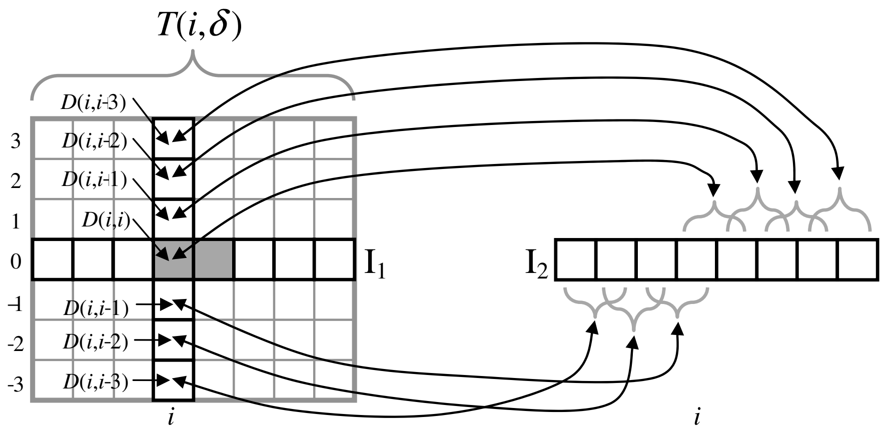


Fig. 3. The similarity table for one-dimensional images. Each entry in the table T stores the similarity score between sub-patches of I_1 and I_2. The first index denotes the patch's corner in I_1, and the second index denotes the relative offset of the patch in I_2.

$$T(i,\delta) = D([I_1(i), I_1(i+1)], [I_2(i+\delta), I_2(i,\delta+1)]) \quad (1)$$

The basic similarity function could range from simple template matching such as L_2 to more complicated functions such as comparing features over two larger windows centered at the locations of these patches. The choice and evaluation of basic similarity functions are beyond the scope of this paper. The function we used in our implementation is described in Appendix A.

We next compare the two inputs in a manner that will be robust to small distortions between the images being compared, and make the measurement symmetric with respect to the two inputs. To allow for small distortions, we maintain two tables M_1, M_2, of the same dimensions as T. These two tables are introduced for efficiency reasons, since the algorithm will reuse the values stored in them several times. M_1 stores the minimum of the comparisons of patches from I_1 to three possible patches from I_2:

$$M_1(i,\delta) = \min(T(i,\delta-1)+c_1, T(i,\delta), T(i,\delta+1)+c_1) \quad (2)$$

M_2 stores the minimum of the comparisons of patches of I_2 to three possible patches in I_1. The offsets are adjusted so that the three patches in I_1 will point to the same patch of I_2:

$$M_2(i,\delta) = \min(T(i-1,\delta+1)+c_1, T(i,\delta), T(i+1,\delta-1)+c_1) \quad (3)$$

The minima are weighted in such a way that a local displacement has a fixed cost c_1. This cost was introduced to give preference to coherent displacements, which are captured in the offset index, over irregular movements, which are captured in the minimum operation. Coherent movement of a large sub-patch results in a single penalty at some stage of the hierarchical framework, while many incoherent movements of sub-patches in various directions result in multiple penalties during the process.

The similarity comparisons used by the algorithm are symmetric in the input images. To see why it is advantageous to perform such bidirectional comparison, consider a case where $I_1 = (0,1,0,1,0,1,\ldots), I_2 = (0,1,1,0,1,1,\ldots)$. Each sub-patch of two pixels in I_1 can find an exact mate in I_2 by moving no more than a single position, but the "11" sub-patches of I_2 have no corresponding mates in I_1, and we would like such an odd patch to affect the similarity measure.

Assume we know the empirical distribution of the values in M_1 and M_2 (the way this distribution is estimated is discussed later). We normalize every value in the tables M_1 and M_2 by this distribution. The normalization gives us better control over the numbers processed by the algorithm since we learn from the distributions how frequent a similarity score is in a particular domain of patches. After normalization, the entries are expected to be distributed uniformly between 0 and 1.

In the iterative step, the algorithm computes the similarity of double-size patches at doubled distances from each other. The similarity score $T(i,\delta)$ of the next phase is based on six similarity values between sub-patches that were computed and normalized in the previous phase:

$$v_1 = M_1(2i, 2\delta), v_2 = M_1(2i+1, 2\delta), v_3 = M_1(2i+2, 2\delta),$$
$$v_4 = M_2(2i, 2\delta), v_5 = M_2(2i+1, 2\delta), v_6 = M_2(2i+2, 2\delta) \quad (4)$$

Since patches double in each round, the range of indexes is halved ($0 \leqslant i < N/2^h$, $-\Delta/2^h \leqslant \delta \leqslant \Delta/2^h$). $T(i,\delta)$ is a weighted geometric mean of $v_1,\ldots,v_6$ normalized by its empirical distribution. The rational for taking the geometric mean is that we would like evidences of high similarity (values close to 0) to have high influence on the combined score. In the implementation we actually multiply the values of $\max(v_i,\varepsilon)$, to prevent the possibility of zero product. A product of zero would propagate through all the stages of the algorithm and will result in perfect match at the last phase. This situation may happen in practice due to image discretization or inaccurate measure of the distribution by which we normalize.

We used weights to put more emphasis on sub-patches that contain salient features such as boundaries. Following Moravec [18], for each point i we compute

$$W(i) = \sum_{j=-2}^{2} (I(i+j) - I(i))^2 \tag{5}$$

(for two dimensional images the sum is over a square window). This weight gives higher emphasis to edge points with a large intensity gradient. We assign a weight w for each patch by averaging $W(i)$ over the points i in that patch (we used a minimal weight value to insure that uniform patches get a small positive weight). To compute the combined measure we define:

$$L = \left(\sum_{i=1}^{6} w_i \log(v_i)\right) \Bigg/ \left(\sum_{i=1}^{6} w_i\right) \tag{6}$$

The weighted geometric mean is then $T(i, \delta) = 2^L$. The representation 2^L is more efficient because it uses a single raise-to-power operation. The computation of logarithms in L is avoided by normalizing v_i directly to the logarithm of its tabulated distribution.

This process is then iterated over with increased patch sizes, as illustrated in Fig. 4. Each phase involves computing the minima tables M_1 and M_2, normalizing their values by an empirical distribution, and storing the normalized weighted geometric means in the T table.

Our implementation uses separate distributions for each phase. It is possible to sample the distribution of phase $h+1$ only after the distribution of phase h had been computed. To build the distributions, we pick a set of image pairs, and build the distributions in phases. At phase h we compute similarity scores for patches of size 2^{h-1} from the tables of all pairs of images simultaneously. Note that there is no need to sample the similarity scores of full-size images, because normalizing the final score will not change the relative order of the final scores.

We end the description of the algorithm with a cross-resolution extension, designed to deal with scale changes in the images. Scale changes need not be homogenous over the images: some parts of the image may expand while other parts may shrink. It might happen that within an expanding part some subparts will in fact shrink in size, and this property of non-uniform scaling is therefore recursive.

To deal with scale variations, we add an additional dimension to all of our data structures. First, we construct a pyramid for the images. Denote by $I(r)$ the image I at resolution r. Our aim is to compare sub-patches of $I_1(r)$ to sub-patches of I_2 at resolutions $r-1$, r, $r+1$ and store these values in the similarity table $T(r,z,i,\delta)$. $z \in \{-1,0,1\}$ is the resolution offset. The corresponding resolution of I_2 is $r+z$. Given two patches, we compute the weighted geometric mean 2^L of the evidences of their parts as before. Two additional cross-resolution terms are computed. If the two patches are at the same resolution (i.e. $z=0$), the central sub-patch from I_1 is tested to fit the low-resolution version of the whole patch from I_2, and vice versa:

$$V_{\text{high-low}} = \min(T'(r,1,2i+1,\delta), T'(r+1,-1,i,2\delta+1)) + c_2 \tag{7}$$

where T' denotes the normalized similarity table of the previous phase. The term is taken with a cost c_2 in order to prevent possible "drift" to base the comparison on repeated low-resolutions of both-patches. The last term to compute is the similarity between the two low-resolution versions of both patches:

$$V_{\text{low-low}} = T'(r+1, z+1, i, \delta) + c_3 \tag{8}$$

This term is important because it allows the algorithm to ignore non-similar details at high resolution. Finally, the similarity score of the two patches is:

$$T(r,z,i,\delta) = \min(2^L, V_{\text{high-low}}, V_{\text{low-low}}) \tag{9}$$

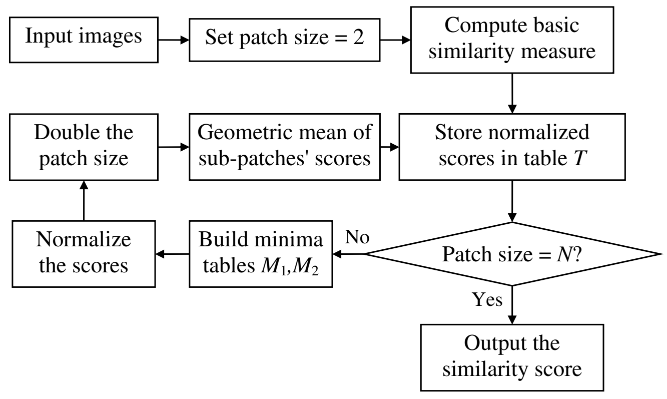

Input images
Set patch size = 2
Compute basic similarity measure
Double the patch size
Geometric mean of sub-patches' scores
Store normalized scores in table T
Normalize the scores
Build minima tables M_1,M_2
No
Patch size = N?
Yes
Output the similarity score

Fig. 4. High-level flowchart of the algorithm.

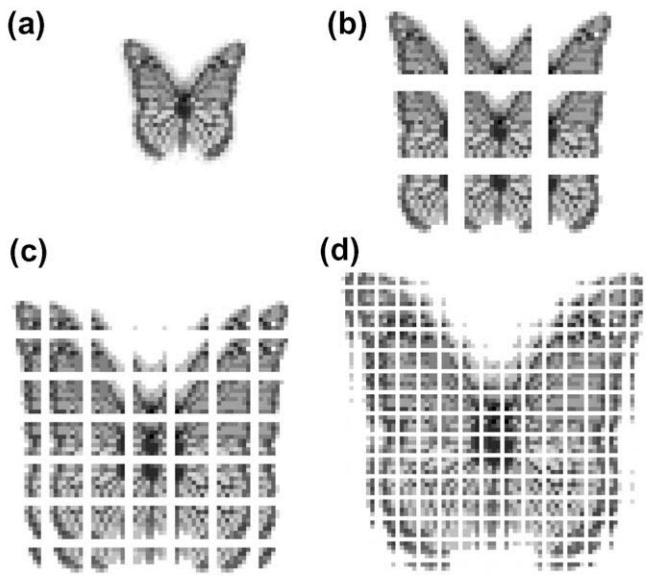


Fig. 5. Arrays of overlapping patches. (a) Original 32×32 image. (b,c), and (d) – Three levels of overlapping patches of sizes 16, 8, 4, where each patch shares pixels with its neighbors.

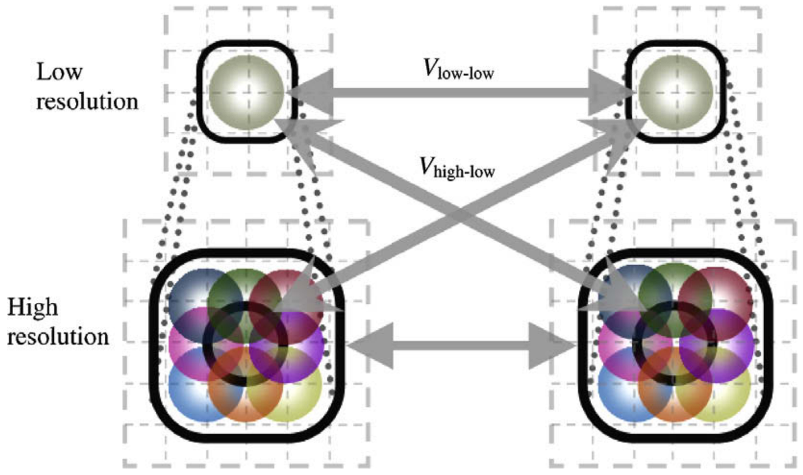


Fig. 6. The similarity score between two patches in two dimensions is computed from the sub-patches, the low-resolution and the cross-resolution similarities.

The scheme carries on naturally to two dimensions. The sub-patches compared at phase h are squares of size $2^h \times 2^h$, as illustrated in Fig. 5. The data structures are extended by additional two indices. $T(r, z, i_1, i_2, \delta_1, \delta_2)$ represents the similarity value between two sub-patches of $I_1(r)$ and $I_2(r + z)$. i_1 and i_2 are the array coordinates of the patch corner in I_1. δ_1 and δ_2 are the displacement of the patch of I_2 relative to the projection of i_1 and i_2 on $I_2(r + z)$. Each patch is covered by nine sub-patches. When the minima tables are computed, each sub-patch is allowed to move one array position left, right, up, down or diagonally. The terms involved in the similarity score are shown

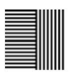
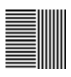
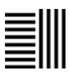
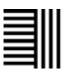
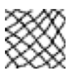
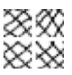
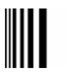
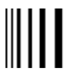
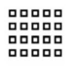
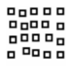
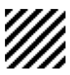
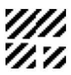
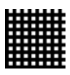
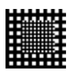
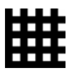
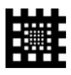
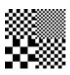
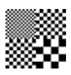
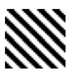
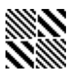
	A	B	
1			Bars of one pixel width. The left part is the same. The right part is shifted one pixel.
2			Bars of two pixels width. The left part is shifted two pixels upwards and the right part two pixels leftwards.
3			Diagonal pattern. Each quarter moves one pixel in opposite directions.
4			Uneven displacement of horizontal bars. The black bars are shifted non-uniformly by 0 to 8 pixels.
5			Random displacement of squares.
6			Diagonal stripes. Each quadrant movers differently.
7			The outer part of B stays at the same resolution, while the inner part is in low resolution.
8			B has the original resolution, low resolution and doubled low resolution.
9			Each quadrant in B is either low or high resolution version of the quarters of A.
10			Quadrants of diagonal stripes in different resolutions plus slight movement.

Fig. 7. Test 1: synthetic patterns undergoing non-rigid deformations.

in Fig. 6. The complexity of the algorithm, as well as the storage requirement, is $O(N^2\Delta^2)$.

The algorithm can also recover the optimal aligning transformation between the images. This could be done by marking at each minimum operation where this minimum has come from. By keeping track of intermediate data-structures one can identify the parts in the images that mostly contributed to the similarity score (e.g. in case of partial occlusion).

We conclude this section by a brief summary of the main principles used in the similarity measure, independently from the specific implementation details. The similarity measure compares images recursively based on the similarities of sub-structures. The composition of sub-structures is flexible, allowing small distortions, shifts and non-uniform changes of scale within the sub-parts. The computed transformation is a composition of a large number of intermediate transformations overlaid one on top of the other. When these transformations are coherent the score is small since the transformation is captured at a higher level of the computation (larger patch size or lower resolution). The output is optimal with respect to the hierarchical score we defined, in the sense that among all hierarchical alignments that we consider it has the smallest hierarchical score. Unlike standard dynamic programming algorithms, the hierarchical score is not a simple global function of the images that is being optimized. The hierarchical score depends on the distributions of similarity scores in the particular domain of patches (which depend on the algorithm that measures these scores). The hope is that if the recurrence is designed properly, the global solution will have the desired properties.

3. Results

We evaluated the performance of the similarity measure described above by applying it to several image test sets of size 32×32 pixels. Four test set are described in Figs. 7–12. The first two test sets are synthetic patterns designed to evaluate and demonstrate the properties of the RDP similarity measure. Fig. 7 describes the first test set in detail. It is made of ten image pairs, labeled (A) and (B). In each pair, the image labeled (B) was derived from (A) by some transformation, such as local shifts and deformations of image parts. In all the examples, the similarity between the two members of the pair is evident and compelling for human observers. However, capturing and explaining these non-rigid transformations is not trivial, and most existing measures for comparing image regions will fail to reflect the similarity of the image pairs. For instance, in the second pair of test 1, the left part moves vertically while the right part moves horizontally. Even if we apply a global

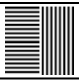
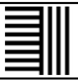
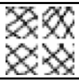
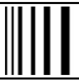
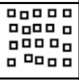
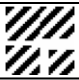
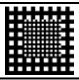
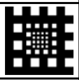
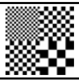
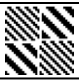
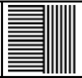
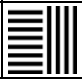
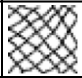
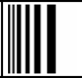
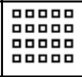
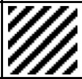
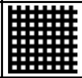
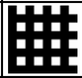
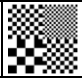
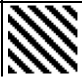
$D(A_i,B_j)$		B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9	B_{10}
											
A_1		**0.008**	0.008	0.914	0.844	0.656	0.844	0.211	0.547	0.508	0.695
A_2		0.008	**0.008**	0.336	0.055	0.102	0.094	0.203	0.219	0.172	0.227
A_3		0.906	0.336	**0.008**	0.234	0.141	0.016	0.898	0.781	0.133	0.016
A_4		0.477	0.117	0.445	**0.008**	0.156	0.141	0.164	0.117	0.438	0.148
A_5		0.445	0.063	0.07	0.055	**0.008**	0.164	0.938	0.891	0.586	0.32
A_6		0.805	0.664	0.047	0.281	0.344	**0.008**	0.641	0.25	0.023	0.117
A_7		0.633	0.297	0.992	0.602	0.984	0.727	**0.008**	0.016	0.57	0.859
A_8		0.961	0.125	0.523	0.219	0.789	0.133	0.055	**0.016**	0.25	0.172
A_9		0.289	0.289	0.133	0.375	0.492	0.086	0.297	0.305	**0.008**	0.008
A_{10}		0.805	0.438	0.055	0.281	0.359	0.344	0.586	0.281	0.023	**0.008**

Fig. 8. RDP scores on test 1. No errors on this set (our implementation normalizes the score using a table such that $0.008 \approx 1/128$ is the lowest possible score).

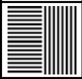
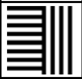
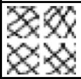
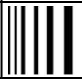
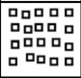
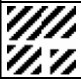
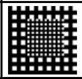
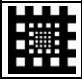
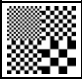
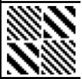
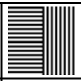
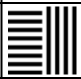
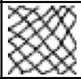
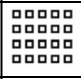
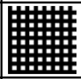
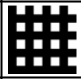
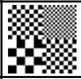
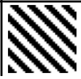
$D(A_i,B_j)$		B_1 	B_2 	B_3 	B_4 	B_5 	B_6 	B_7 	B_8 	B_9 	B_{10} 
A_1		**0.500**	0.500	0.502	**0.468**	0.495	0.498	0.490	0.500	0.500	0.515
A_2		0.500	**1.000**	0.473	**0.468**	0.509	0.488	0.494	0.500	0.500	0.485
A_3		0.504	0.524	**0.375**	0.496	0.480	0.494	0.481	0.471	0.555	0.477
A_4		0.500	0.500	0.473	**0.453**	0.494	0.493	**0.422**	0.472	0.500	0.491
A_5		**0.491**	**0.472**	0.513	0.560	**0.387**	0.523	0.482	0.543	0.495	0.502
A_6		0.500	0.500	0.470	0.500	0.518	**0.241**	0.499	0.446	0.531	0.501
A_7		0.500	0.500	0.505	0.518	0.503	0.497	**0.144**	0.406	0.644	0.574
A_8		0.500	0.500	0.478	0.518	0.606	0.449	0.511	**0.125**	0.572	0.511
A_9		0.500	0.500	0.481	0.500	0.507	0.557	0.633	0.586	**0.500**	**0.429**
A_{10}		0.500	0.500	0.539	0.500	0.475	0.518	0.501	0.554	**0.469**	**0.501**

Fig. 9. Scores of cross-correlation on test 1. Errors are underlined. Error rate = 0.45.

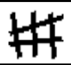
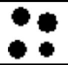
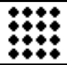
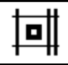
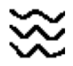
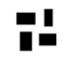
$D(A_i,B_j)$		B_1 	B_2	B_3	B_4 	B_5 	B_6	B_7 	B_8	B_9	B_{10}
A_1		**0.023**	0.063	0.281	0.523	0.852	0.586	0.266	0.75	0.18	0.477
A_2		0.156	**0.055**	0.18	0.32	0.406	0.477	0.141	0.727	0.391	0.656
A_3		0.117	0.25	**0.016**	0.477	0.609	0.438	0.117	0.531	0.25	0.477
A_4		0.492	0.641	0.352	**0.008**	**0.055**	0.422	0.492	0.102	0.508	**0.078**
A_5		0.609	0.516	0.797	**0.141**	**0.375**	0.813	0.211	0.391	0.281	0.148
A_6		0.391	0.445	0.617	0.336	0.102	**0.016**	0.453	0.945	0.234	0.57
A_7		0.32	0.211	0.352	0.156	0.063	0.102	**0.031**	0.391	0.492	0.102
A_8		0.914	0.742	0.922	0.359	0.969	0.977	0.313	**0.047**	0.961	0.703
A_9		0.211	0.266	0.555	0.297	0.75	0.266	0.211	0.156	**0.039**	0.227
A_{10}		0.391	0.266	0.203	**0.055**	0.055	0.234	0.203	0.336	0.313	**0.086**

Fig. 10. Test 2: synthetic pairs of patches with similar form. The similarity scores of RDP are shown, and errors underlined. Error rate = 0.2. The error rates of L_2 and correlation are 0.35 and 0.3.

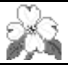
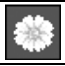
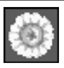
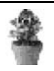
$D(A_i,B_j)$		B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9	B_{10}
											
A_1		**0.07**	0.516	0.742	0.367	0.914	0.508	0.664	0.398	0.633	0.484
A_2		0.383	**0.055**	0.063	0.141	0.211	0.141	0.141	0.773	0.711	0.102
A_3		0.508	0.18	**0.055**	0.094	0.148	0.055	0.094	0.773	0.578	0.07
A_4		0.57	0.352	0.141	**0.055**	0.141	0.063	0.063	0.633	0.82	0.055
A_5		0.891	0.477	0.094	0.094	**0.031**	0.055	0.172	0.961	0.828	0.102
A_6		0.445	0.148	0.141	0.172	0.141	**0.047**	**0.055**	0.75	0.891	0.141
A_7		0.375	0.367	0.094	**0.055**	0.25	0.102	**0.07**	0.094	0.07	0.07
A_8		0.844	0.703	0.414	0.281	0.656	0.539	0.188	**0.023**	0.688	0.422
A_9		0.578	0.555	0.477	0.25	0.57	0.477	0.422	0.656	**0.055**	0.516
A_{10}		0.68	0.25	0.141	0.07	0.102	0.141	0.102	0.695	0.82	**0.055**

Fig. 11. Test 3: icons of flowers. The similarity scores of RDP are shown, and errors underlined. Error rate = 0.1. The error rate of both L_2 and correlation is 0.4.

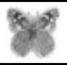
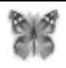
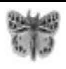
$D(A_i,B_j)$		B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9	B_{10}
											
A_1		**0.031**	0.055	0.797	0.898	0.047	0.055	0.188	0.18	0.055	0.055
A_2		0.055	**0.031**	0.773	0.922	0.055	0.055	0.141	0.086	0.055	0.055
A_3		0.609	0.805	**0.031**	**0.039**	0.656	0.516	0.492	0.453	0.594	0.82
A_4		0.719	0.898	0.055	**0.047**	0.523	0.523	0.383	0.125	0.75	0.555
A_5		0.063	0.078	0.484	0.578	**0.047**	0.063	0.055	0.117	0.055	0.078
A_6		0.055	0.055	0.758	0.781	0.055	**0.039**	0.188	0.125	0.055	0.094
A_7		0.141	0.078	0.578	0.688	0.063	0.078	**0.055**	0.055	0.055	0.094
A_8		0.164	0.258	0.242	0.484	0.359	0.281	0.234	**0.039**	0.203	0.359
A_9		0.055	0.055	0.609	0.875	0.055	0.055	0.063	0.234	**0.039**	0.055
A_{10}		0.055	0.055	0.852	0.93	0.055	0.078	0.141	0.234	0.055	**0.039**

Fig. 12. Test 4: real images of butterflies. The similarity scores of RDP are shown, and errors underlined. Error rate = 0.05. The error rates of L_2 and correlation are 0.35 and 0.25.

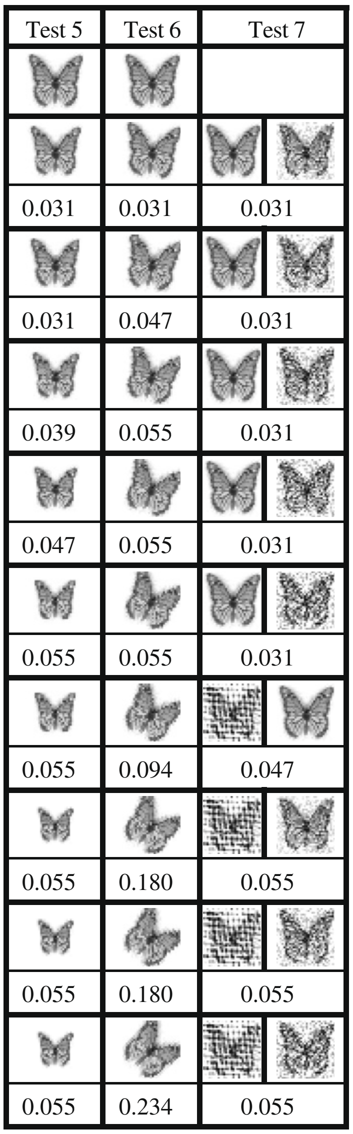
rigid transformation before computing a point-wise metric such as L_2, half of the pixels will be misaligned. The simple patterns in this example already pose conceptual difficulties to many methods based on local features, since any local feature computed on the left image will match multiple features on the right image (except for the centers of those images). Thus, evaluating which of these matches is "correct" becomes an optimization problem that has no exact solution using only affine transformations. In contrast, our method allows sub-patches to move independently and captures the transformation locally. While some feature-based methods will penalize for the small movement of each feature a bit, our method will penalize only for the movement of the two parts as wholes.

The evaluation is summarized in the tables shown in Figs. 8–12. Each test set consists of 10 pairs of similar images. A comparison of an image to its similar pair was considered as an error when the similarity score of the correct match was not the minimum of the comparisons to the other 10 possible images along the row or the column of the image. Ideally, the elements on the diagonal of the tables should be the minimal score along their rows and columns. The mistake rate of a test is the number of error out of the 20 possible. We compared the scores of RDP with L_2 and correlation (transformed linearly to $[0,1]$). The performance of RDP on test sets 1–4 is shown in Figs. 8, 10, 11 and 12. The error rates of RDP on these tests are 0, 0.2, 0.1 and 0.05. In comparison, the error rates of L_2 are 0.5, 0.35, 0.4 and 0.35. The error rates of correlation are 0.45 (Fig. 9), 0.3, 0.4 and 0.25. One can see that the results obtained by RDP are superior to standard distance functions used to compare images, such as L_2 and correlation. The basic reason is that in the RDP comparison, parts of the images can undergo different deformations simultaneously and still be matched.

The performance of the method should not be understood just from the statistical rate of success, because the numeric results depend on how confusing the images are, as well as on several implementation details. The score depends on the distributions used (we sampled the distributions for each test separately), the basic similarity function (we applied the basic similarity measure described in the appendix over patches of 4×4 pixels centered around the 2×2 patches), the choice of cost constants and the weight of the background in the images. In all of these tests we used the same constants: $c_1 = c_2 = 0.05$, $c_3 = 0.2/(h-1)$, so that the cost for comparison in low resolution decreases with the phase h of the algorithm.

We investigated how the similarity score of our implementation varies under controlled image deformations. The results are shown in Fig. 13. In test 5, the images are scaled by factors of $1, 0.95, 0.9, \ldots, 0.55$. The score below each image is the similarity to the top image (the scores do not vary smoothly because we used tabulated distributions). In test 6, the images are rotated by $0°, 5°, 10°, 15°, \ldots, 45°$. Test 7 evaluates the behavior of the score under different levels of noise (the score is shown below each pair). For comparison, the averaged L_2 distance between the last pair is 0.327. These examples suggest that the scores are robust to small changes in scale, small rotations and noise.

We also examined the dependency of the algorithm on specific factors such as the empirical distributions and the use of symmetric, bi-directional computation. We concluded that performing a bi-directional computation slightly improves the accuracy on the expense of running time. In practice, normalizations by the empirical distributions are not always necessary. On the other hand, there are situations where not using the correct distribution can lead to convergence of the similarity score to zero. This may happen because at each phase we take the minimum of several scores, and in general we would expect the similarity score to decrease as these operations are iterated. We tested the algorithm with five levels in the hierarchy and expect that



Test 5	Test 6	Test 7
0.031	0.031	0.031
0.031	0.047	0.031
0.039	0.055	0.031
0.047	0.055	0.031
0.055	0.055	0.031
0.055	0.094	0.047
0.055	0.180	0.055
0.055	0.180	0.055
0.055	0.234	0.055

Fig. 13. Evaluation of RDP's robustness. Test 5: change of scale. Test 6: rotation. Test 7: noise.

fewer levels would diminish this problem. Additional details are provided in [1].

4. Discussion

We presented a general framework for assessing the similarity of image regions based on the composition of parts. The measure uses a basic similarity measure, such as correlation, only for simple local sub-parts, and then evaluates the similarity between the regions by identifying common sub-parts. The scheme allows complex, non-rigid deformation of the images, and penalizes irregular deformations (small sub-parts that move independently) more than coherent shifts of larger sub-parts. The operations are performed efficiently by a dynamic programming like implementation.

Our implementation demonstrated the qualitative properties of the scheme. No domain-specific features have been detected, and there was no need to specify the spatial configuration of the parts. The parts within each image are not identified explicitly. The algorithm matches systematically a large number of overlapping sub-patches. Similar overlapping patches impose similar configuration. The dynamic programming procedure looks for an optimal transformation that aligns the patches of both images. This transformation is not a global transformation, but a composition of multiple local transformations of sub-patches at various sizes. The aligning transformation has many degrees of freedom that allow for highly flexible matching. The method involves a phase of collecting statistics regarding the distribution of similarity scores in the domain of the images. The scheme is biologically plausible because it is based on multiple simple operations, and it can be implemented in neural networks and parallel hardware.

Our experiments show that the RDP implementation is superior to L_2 and correlation. The algorithm can deal with various types of non-rigid transformations, where parts of the images can undergo different types of deformations at the same time. Objects that are perceptually similar are often judged by the algorithm to be similar, even when a direct comparison does not detect such similarity. This supports the assumption that perceptual similarity is based in part on common substructures shared by the objects under comparison.

However, the current implementation of the RDP method is still inferior to humans' similarity assessment. Two main causes for inaccuracy have been identified in testings. The first weakness is the saliency measure for regions. Small but important regions in the images may have large impact on the perceived similarity. The second issue neglected by our scheme is invariant topological properties of shapes. In particular, the method does not identify and match the contours of the shapes. It does not examine whether the shapes are connected, closed or contain holes. This is in contrast to humans' perception, which can use at least some aspects of these properties [19]. It was encouraging to observe that low error rates were achieved even though the ability to match contours was not considered in the design of our implementation. Incorporating some pre-processing for contour detection or enhancement is likely to further improve the performance of the similarity measure and its agreement with human perception.

The principles of the RDP algorithm are general and can be extended to higher dimensions. For instance, it is possible to deal with independent rotations of sub-patches by adding an additional dimension to the tables and comparing each patch to several rotated versions of the patch from the other image. The same method could be used for the alignment of 3D volumes represented as sets of voxels, or the alignment of cube-patches in video sequences. These possibilities and extensions remain for future research.

Acknowledgement

S.U. was supported by EU IST Grant FP6-2005-015803 and ISF Grant 7-0369.

Appendix A

In this appendix we describe a variant of cross correlation that we used as our basic similarity function to compare sub-patches. We found this variant to be more effective than correlation since it takes into account the difference in the mean brightness of the patches. In a real world application global brightness normalization could be performed before applying this operation to local sub-patches.

We define the inter-correlation between two vectorized patches x and y with expectations $E(x) = \bar{x}$ and $E(y) = \bar{y}$:

$$IC(x,y) = \frac{E((x-\bar{y})(y-\bar{x}))}{\sqrt{E(x-\bar{y})^2 E(y-\bar{x})^2}} \tag{10}$$

By definition $IC(x,y) = 0$ when the denominator is 0. Properties of this expression are listed below:

1. $IC(x,y) = \frac{E((x-\bar{x})(y-\bar{y}))-(\bar{x}-\bar{y})^2}{\sqrt{(E(x-\bar{x})^2+(\bar{x}-\bar{y})^2)(E(y-\bar{y})^2+(\bar{x}-\bar{y})^2)}}$
2. $|IC(x,y)| \leqslant 1$. This follows from the Cauchy-Schwarz inequality. Having a bounded range is more convenient to work with compared to metrics such as Sum of Squared Differences (SSD).
3. $|IC(x,y)| = 1 \iff y = \bar{x} + a(x-\bar{y})$ for some constant $a \neq 0$. The proof is by equating to 0 the discriminant of the non-negative quadratic expression: $E(a(x-\bar{y}) - (y-\bar{x}))^2$.

By taking expectations it follows that $\bar{y} = \bar{x} + a(\bar{x} - \bar{y})$. At the extremes of $IC(x,y)$, either $a = -1$ or $\bar{x} = \bar{y}$. Substituting y by $\bar{x} + a(x-\bar{y})$ in the IC formula, we get:

$IC(x,y) = +1 \iff y = \bar{x} + a(x-\bar{x})$ and $a > 0$

$$IC(x,y) = -1 \Longleftrightarrow (y = \bar{x} + a(x-\bar{x}) \text{ and } a < 0) \text{ or } (y = \bar{y} + \bar{x} - x)$$

$IC(x,y) = 1$ if and only if x and y have the same mean, and their deviations from this mean are up to a positive constant factor. $IC(x,y) = -1$ in two cases. The first possibility is that both vectors have the same mean, and their deviation from this mean is a linear reflection. The second possibility is that they have different means, but the distribution around their means is an exact reflection.

References

[1] A. Ecker, Images Similarity Based on the Distributions of Similar Substructures, M.Sc. thesis (2002), Weizmann Institute of Science.
[2] A. Ecker, S. Ullman, A hierarchical non-parametric method for capturing non-rigid deformations, in: Proceedings of the Second Canadian Conference on Computer and Robot Vision 2005, pp. 50–56.
[3] D.G. Lowe, Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision 60 (2) (2004) 91–110.
[4] S. Belongie, J. Malik, J. Puzicha, Shape matching and object recognition using shape contexts, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (4) (2002) 509–522.
[5] A. Apostolico, Z. Galil, Pattern matching algorithms, Oxford University Press, 1997.
[6] A.L. Ratan, W.E.L. Grimson, and W.M. Wells III. Object detection and localization by dynamic template warping, In Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition (1998), 634-640.
[7] R. Basri, L. Costa, D. Geiger, D. Jacobs, Determining the similarity of deformable shapes, Vision Research 38 (1998) 2365–2385.
[8] E. Milios, E.G.M. Petrakis, Shape retrieval based on dynamic programming, IEEE Transactions on Image Processing 9 (1) (2000) 141–147.
[9] P. Klein, S. Tirthapura, D. Sharvit, and B. Kimia, A tree-edit distance algorithm for comparing simple, closed shapes, In Proceedings of ACM-SIAM Symposium on Discrete Algorithms (2000), 696-704.
[10] P.F. Felzenszwalb, Representation and detection of deformable shapes, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (2) (2005) 208–220.
[11] M.A. Fischler, R.A. Elschlager, The representation and matching of pictorial structures, IEEE Transactions on Computers 22 (1) (1973) 67–92.
[12] P.F. Felzenszwalb, D.P. Huttenlocher, Pictorial structures for object recognition, International Journal of Computer Vision 61 (1) (2005) 55–79.
[13] E. Sali, S. Ullman, Combining class-specific fragments for object classification, in: Proceedings of the British Machine Vision Conference 1999, pp. 203–213.
[14] R.C. Nelson, A. Selinger, A cubist approach to object recognition, in: Proceedings of IEEE International Conference on Computer Vision 1998, pp. 614–621.
[15] A.A. Efros, T.K. Leung, Texture synthesis by Non-parametric Sampling, in: proceedings of IEEE international conference on computer vision 1999, pp.1033–1038.
[16] B. Leibe, A. Leonardis and B. Schiele, Combined object categorization and segmentation with an implicit Shape Model, in: ECCV'04 Workshop on Statistical Learning in Computer Vision (2004).
[17] J. Winn, J. Shotton, The Layout Consistent Random Field for Recognizing and Segmenting Partially Occluded Objects, in: Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition 2006, pp. 37–44.
[18] H.P. Moravec, Towards Automatic Visual Obstacle Avoidance, in: Proceedings of the International Joint Conference on Artificial Intelligence 1977, 584.
[19] S.E. Palmer, Structural aspects of visual similarity, Memory and Cognition 6 (2) (1978) 91–97.