A Hierarchical Non-Parametric Method for Capturing Non-Rigid Deformations

Ady Ecker
University of Toronto
adyecker@cs.toronto.edu

Shimon Ullman
Weizmann Institute of Science
shimon.ullman@weizmann.ac.il

Abstract

We present a novel approach for measuring deformations between image patches. Our algorithm is a variant of dynamic programming that is not inherently one-dimensional, and its scores are on a relative scale. The method is based on the combination of similarities between many overlapping sub-patches. The algorithm is designed to be robust to small deformations of parts at various positions and scales.

1. Introduction

Current approaches for measuring images similarity usually consist of two steps. The first step is carried out by matching features from both images. This matching can range from simple template matching to complex feature matching such as SIFT features [1] or shape context [2]. The second step is searching for a transformation that aligns the images. This is usually done by using affine transformations or more complicated parametric transformations [2]. In this paper we deal with the question of how to capture non-rigid deformations. Figure 1 illustrates the sort of transformations we would like to be able to explain. At the bottom image of figure 1, part ***A*** stays in the same place, part ***B*** moves rightwards and part ***C*** upwards. With more subdivisions of the image, this sort of transformations can be made arbitrarily complex and no parametric transformations can capture that sort of changes. Note that 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 a wide range of non-rigid deformations of image patches in theory.

Our work was inspired by dynamic programming methods for matching strings, such as edit distance [3]. The main difficulty in applying the dynamic programming paradigm directly is that two

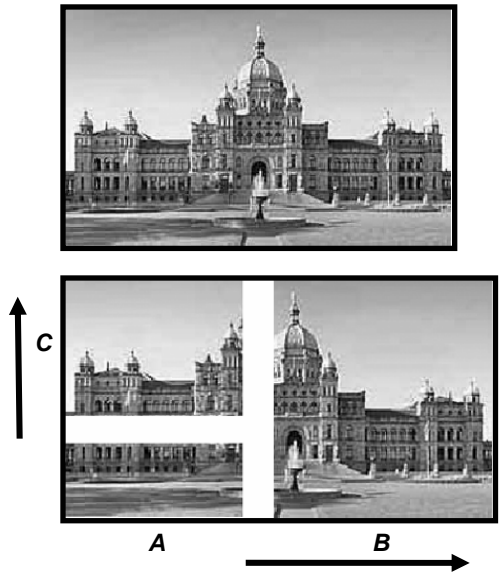


Figure 1. The bottom image was obtained by shifting patches *B* (rightwards) and *C* (upwards). While the two images are similar, no parametric transformation can explain this.

dimensional images have no natural scanning direction. For example, the rows of the two images in figure 1 don't match exactly, as well as their columns. There have been several previous attempts to use dynamic programming to evaluate similarity of images. In [4] the longest common subsequence algorithm has been used over the columns of a matched template. In [5] and [6] shape contours were aligned using dynamic programming. In [7] 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 [8] a polygon triangulation is matched by dynamic programming using an elimination order for the triangles.

Our method differs substantially from these works in that it works entirely in two dimensions. To do so we had to abandon a key property of the dynamic

programming paradigm, the definition of a cost function. Instead, we apply iterative steps that optimize some implicit 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 empiric 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 [9], the converse also holds: if two images have many similar overlapping shape fragments, then with high probability the images are similar. It is very 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. Similar ideas appear in [10]. Works in computer graphics (e.g. [11]) demonstrated patch-based method to be very effective for texture synthesis. However, the problem of systematically identifying similar sub-patches in images has not been studied previously.

2. The Relative Dynamic-Programming algorithm

In this section we present our algorithm, which we call RDP (Relative Dynamic Programming). 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 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 reuses optimal solutions to sub-problems.

In contrast to standard dynamic programming algorithms our algorithm is extendible to more than one dimension. We don't only mean that we use multidimensional tables. 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 implicit, i.e. cannot be explicitly written down in a short way. We don't start by stating the objective function and formulating the iterative steps to optimize this function. We start by formulating the iterative steps in a way that the resulting 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. Note that even for standard dynamic programming algorithms, like edit distance [3], 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.

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. The algorithm works in phases. In phase h, where $h=1,2,\ldots,\log(N)$, the algorithm computes the similarity score between 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 at phase transitions. The final output is the similarity score of phase $\log(N)$, when the patch size equals N.

Intermediate scores are stored in a table for use in the subsequent phase. We assume 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). Some of the entries in the similarity table are empty due to boundary overflows. From here on we will ignore such problems which are handled in the implementation.

The first phase is done using a basic similarity function D on the intensity values of two 2-pixels patches:

$$T(i,\delta)=D([\mathrm{I}_1(i),\mathrm{I}_1(i+1)],[\mathrm{I}_2(i+\delta),\mathrm{I}_2(i+\delta+1)]).$$

The basic similarity function could range from simple template matching such as L_2 to more complicated such as comparing features over two bigger 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 the appendix.

To allow for small distortions between the images being compared, we maintain two tables M_1, M_2, of the same dimensions as T. These two tables are introduced for efficiency reasons. 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:

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

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:

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

The minima are weighted in such a way that a movement has a fixed cost c_1. This cost was introduced to give preference to coherent movements, 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.

To see why it is safer to perform the bidirectional comparison, consider a case where I_1=(0,1,0,1,0,1,…), I_2=(0,1,1,0,1,1,…). Each sub-patch of two pixels in I_1 can find an exact mate in I_2 by moving no more than one position, but the "11" sub-patches of I_2 have no corresponding mates in I_1.

Assume we know the empiric distribution of the values in M_1 and M_2. Every value in the tables M_1 and M_2 is normalized by this distribution. After normalization, the entries are expected to be distributed uniformly between 0 and 1.

In the iterative step, the algorithm computes the similarity of doubled 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:

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

$T(i,\delta)$ is a weighted geometric mean of these values normalized by its empiric 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 [12], for each point i in the image, we compute

$$W(i)=\sum_{j=-2}^{2}\left(\mathrm{I}(i+j)-\mathrm{I}(i)\right)^2.$$

We assign a weight w for each patch by averaging $W(i)$ over the points i in that patch (we used some minimal weight value to insure that uniform patches get a small positive weight). Let

$$L=\frac{\sum_{i=1}^{6} w_i \log(v_i)}{\sum_{i=1}^{6} w_i}.$$

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

This process is iterated over. Each phase involves computing the minima tables M_1 and M_2, normalizing their values by an empiric 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.

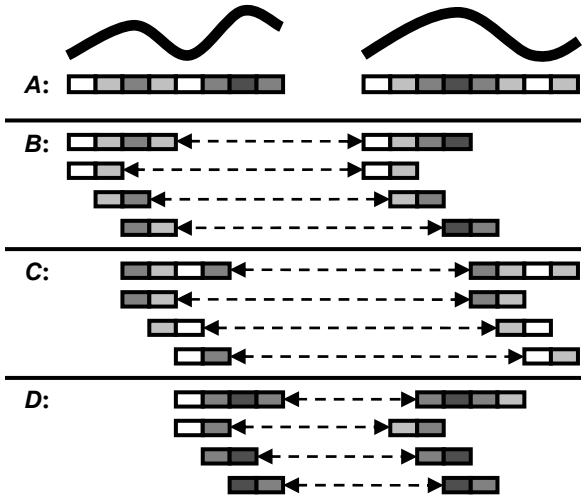


Figure 2. Hierarchical alignment of 1D patches. *A*: The two 8-pixels patches being compared. *B*, *C*, and *D*: The best matches for the 4-pixels sub-patches based on the movements of the 2-pixels sub-patches.

The resulting hierarchical alignment is illustrated in figure 2. Patches of two pixels can match by moving up to one position with respect to the four-pixel patch that contains them. Patches of four pixels can move two positions, and are matched by their overlapping sub-patches of two pixels.

We end the description of the algorithm with a cross-resolution enhancement. Scale changes need not be homogenous over the images: some parts of the image may grow while other parts shrink. It might be that within a growing part some subparts are shrinking, and this property is recursive.

To deal with scale variations, we add an additional dimension to all of our data structures. First we build 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:

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

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:

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

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

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

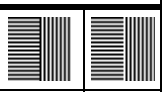
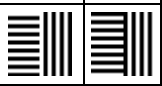
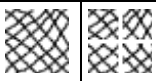
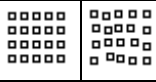
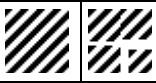
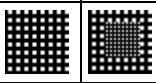
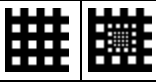
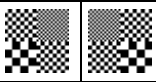
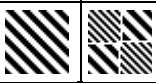
The scheme carries on naturally to two dimensions. The sub-patches compared at phase h are squares of size $2^h \times 2^h$. The data structures are extended by additional two indices. $T(r,z,i_1,i_2,\delta_1,\delta_2)$ represents the similarity value between two sub-patches of $I_1(r)$ and $I_2(r+z)$. 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 complexity of the algorithm, as well as the storage requirement, is $O(N^2 \cdot \Delta^2)$.

The algorithm could 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).

3. Results

We examined the performance of our implementation on several test sets of 32×32 images. Three of the sets are shown in table 1. The images in test 1 were created to demonstrate the properties of the similarity measure. 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 rigid transformation before computing L_2, half of the pixels will be misaligned. Notice that this simple example poses conceptual difficulties to any method based on local features, since any local feature computed on the left image will match too many 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 within the affine model. In contrast, our method allows sub-patches to move independently and capture the transformation locally. While feature based method 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.

Table 1.

	Test 1		Test 2		Test 3	
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						
RDP		0		0.1		0.05
L_2		0.5		0.4		0.35
Correlation		0.45		0.4		0.25

In each of the three tests every image was compared to the 10 images on the other column. A comparison of an image to its similar pair was considered as a mistake when the score was not the minimum out of the comparisons to the 10 possible pairs in that test. The mistake rate of a test is the number of error out of the 20 possible. Mistakes rates are summarized at the bottom of table 1. 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 and on several implementation details. The score depends on the distributions used (we sampled the distributions for each test separately), the basic similarity function, the choice of cost constants and the weight of the background in the images. In all of these tests we used the same constants $c_1 = c_2 = 0.05,\ c_3 = \frac{0.2}{h-1}$, so that the cost for comparison in low resolution decreases with the phase. One can see that the results

Table 2.

Test 4	Test 5	Test 6
		
		
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

obtained by RDP are superior to standard distance functions used to compare images, such as L_2 and correlation. Parts of the images can undergo different deformations simultaneously and still be matched.

We investigated how the similarity score of our implementation varies under controlled image deformations. The results are shown in table 2. In test 4, the images are scaled by factors of 1,0.95, 0.9,…,0.55. The score below each image is the similarity to the top image (the scores don't vary smoothly because we used tabulated distributions). In test 5, the images are rotated by 0,5,10,15,…,45 degrees. Test 6 evaluates the behavior of the score

under different levels of noise (the score is written 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 empiric distributions and the bi-directional computation [13]. We concluded that in practice normalizations by the empiric distributions are not always necessary, because the evidences were highly correlated so the distribution of the minimum of several normalized variables was almost uniform. On the other hand, there are situations where not using the correct distribution can lead to convergence of the similarity score to 0. Performing a bi-directional computation slightly improves the accuracy on the expense of running time.

4. Discussion

We presented a general framework for similarity assessment based on the composition of parts. Our implementation demonstrated the qualitative properties of the scheme. No domain-specific features have to be detected and there is no need to specify the 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 many local transformations of sub-patches at various sizes, performed one on top of the other. The aligning transformation has many degrees of freedom that allow for very flexible matching. The method involves an unsupervised learning phase, in which the program collects statistics on the distribution of similarity scores in the domain of the images. The scheme is plausible because it involves only simple operations. It can be implemented in parallel hardware and with neural networks.

Our experiments show that the implementation of RDP 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 point-wise comparison does not detect such similarity. This supports the assumption that perceptual similarity is based in part on the common substructures shared by the objects.

However, the implementation is still inferior to humans' similarity assessment. Two main causes for inaccuracy have been identified. 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 pays no attention to whether the shapes are connected, closed or contain holes. This is in contrast to humans' perception [14]. 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.

The principles of the RDP algorithm are fairly general and could 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 remain for future research.

5. Appendix

In this appendix we describe a new variant of cross correlation that we used as our basic similarity function. We found this variant to be more effective than correlation since it takes into account the difference between the means of the two vectors.
We define the inter-correlation between two vectors x and y with means $\bar{x}$ and $\bar{y}$:

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

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

1. $IC(x,y) =$
$$\frac{E\big((x-\bar{x})(y-\bar{y})\big)-(\bar{x}-\bar{y})^2}{\sqrt{\Big(E(x-\bar{x})^2+(\bar{x}-\bar{y})^2\Big)\Big(E(y-\bar{y})^2+(\bar{x}-\bar{y})^2\Big)}}$$

2. $|IC(x,y)| \le 1$. This follows from the Cauchy-Schwarz inequality.

3. $|IC(x,y)| = 1 \Leftrightarrow y = \bar{x} + a(x-\bar{y})$ for some constant $a \neq 0$.

The proof is by equating to 0 the discriminant of the non-negative quadratic expression:

$$E\left(a(x-\bar{y})-(y-\bar{x})\right)^2 .$$

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:

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

$IC(x,y)=1$ 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 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] D.G. Lowe, "Distinctive image features from scale-invariant keypoints", *IJCV* 60(2), 91-110, 2004.

[2] S. Belongie, J. Malik and J. Puzicha, "Shape Matching and Object Recognition Using Shape Contexts", *PAMI* 24(4), 509-522, 2002.

[3] A. Apostolico and Z. Galil, *Pattern matching algorithms.* Oxford university press, 1997.

[4] A.L. Ratan, W.E.L. Grimson, and W.M. Wells III. "Object detection and localization by dynamic template warping", *CVPR*, 634-640, 1998.

[5] R. Basri, L. Costa, D. Geiger, and D. Jacobs, "Determining the similarity of deformable shapes", *Vision Research* 38, 2365-2385, 1998.

[6] E. Milios and E.G.M. Petrakis, "Shape Retrieval Based on Dynamic Programming", *IEEE Trans. on Image Processing* 9(1), 141-147, 2000.

[7] P. Klein, S. Tirthapura, D. Sharvit, and B. Kimia, "A tree-edit distance algorithm for comparing simple, closed shapes", *SODA*, 696-704, 2000.

[8] P.F. Felzenszwalb, "Representation and Detection of Deformable Shapes", *PAMI* 27(2), 208-220, 2005.

[9] E. Sali and S. Ullman, "Combining class-specific fragments for object classification", *BMVC*, 203-213, 1999.

[10] R.C. Nelson and A. Selinger, "A Cubist Approach to Object Recognition", *ICCV*, 614-621, 1998.

[11] A.A. Efros and T.K. Leung, "Texture Synthesis by Non-Parametric Sampling", *ICCV*, 1033-1038, 1999.

[12] H.P. Moravec, "Towards automatic visual obstacle avoidance", *Proc. of the International Joint Conference on Artificial Intelligence*, 584, 1977.

[13] A. Ecker, *Images Similarity Based on the Distributions of Similar Substructures*, M.Sc. thesis, Weizmann Institute of Science, 2002.

[14] S.E. Palmer, "Structural aspects of visual similarity", *Memory and cognition* 6(2), 91-97, 1978.