PREPRINT, published in
International Journal of Pattern Recognition and Artificial Intelligence,
Vol. 17, No. 7, pages 1279–1302 (2003)

EXTRACTION AND MATCHING OF SYMBOLIC CONTOUR GRAPHS

Tino Lourens

Honda Research Institute Japan, Co., Ltd.
8-1 Honcho, Wako-shi, Saitama, 351-0114, Japan
tino@jp.honda-ri.com

Rolf P. Würtz

Institute for Neurocomputing, Ruhr-University Bochum,
D-44780 Bochum, Germany
Rolf.Wuertz@neuroinformatik.ruhr-uni-bochum.de

We describe an object recognition system based on symbolic contour graphs. The image to be analyzed is transformed into a graph[a] with object corners as vertices and connecting contours as edges. Image corners are determined using a robust multiscale corner detector. Edges are constructed by line-following between corners based on evidence from the multiscale Gabor wavelet transform. Model matching is done by finding subgraph isomorphisms in the image graph. The complexity of the algorithm is reduced by labeling vertices and edges, whereby the choice of labels also makes the recognition system invariant under translation, rotation, and scaling. We provide experimental evidence and theoretical arguments that the matching complexity is below $O(\#V^3)$, and show that the system is competitive with other graph-based matching systems.

Keywords: Object recognition; contours; graph matching; corner detection; edge extraction; subgraph isomorphism; attributed graphs.

1. Introduction

A human carries out an object recognition task with such ease, that we hardly consider it as difficult. A closer view reveals the problems behind this act of perception. A distribution of light intensities on the retinas is processed by the brain. Although a distribution of light intensities can change considerably, e.g., when an object is placed in a different environment or scaled, the object is being recognized as the same. The brain transforms visual data into another representation where these changes are compensated for, and then performs recognition tasks. Similarly in computer vision, where the space in which tasks are performed is usually different from the space of visual measurements. It is usually so different from the space of visual measurements that using the first to guide the gathering of information in the second is an area that is said to be still in its infancy.[12]

[a]A graph in its basic form is defined as a set of vertices and a set of edges, where the latter are tuples of vertices.

PREPRINT, published in
International Journal of Pattern Recognition and Artificial Intelligence,
Vol. 17, No. 7, pages 1279–1302 (2003)

Unfortunately, the functionality and representation space of the brain are partly known only. The existence of the so called simple, complex, and endstopped cells found by Hubel and Wiesel[19] in the visual cortex proves that edges and corners play an important role in mammalian vision. Even though much effort has gone into revealing the functionality of different groups of cells in the primary visual cortex, still most parts are partly understood only. The complexity beyond the visual cortex increases dramatically and neural models about recognition become very hypothetical, even though it seems obvious that matching of some kind must take place.

Starting from the observation that *corners* and *contours* of objects are relatively robust under perspective transformations there is a variety of matching algorithms that rely on corners as vertex points and connect vertices if there is a visual contour connecting the corners. Throughout this paper we will use the term "contour" for visual edges in order to avoid confusion with graph edges. Our algorithm is of this type, and others include those by Eshera and Fu,[10] Messmer and Bunke,[29] Gold and Rangarajan,[15] and Rosin.[37]

More concretely, the term *graph matching* refers to the process of trying to find correspondences between graphs. Two graphs are *isomorphic* if there is a one-to-one correspondence function between the vertices of the two graphs such that the structure of the edges is preserved by the function (McHugh[28]). If one of the graphs involved in the matching process is larger than the other, i.e. the image graph $G(V, E)$ contains more vertices than the model graph $G_m(V_m, E_m)$, then we are looking for a *subgraph isomorphism* from G_m to G. In many applications, the encoding of objects as graphs will not be perfect due to, e.g, noise or limitations of the graph extraction algorithm. Hence, it is not realistic to require isomorphic matching. A natural idea is to introduce cost functions and incorporate the concept of errors into graph matching. Graphs are then compared to each other by means of the *error-tolerant subgraph isomorphism*.

Dependent on the application, different attributes are assigned to the vertices and edges in a graph. Heuristics that use these attributes can decrease the search space to a low order polynomial (Gold and Rangarajan[15]). The currently existing algorithms can globally be divided into two groups.

The *first group* involves the construction of a state-space which is then searched exhaustively, a procedure which always guarantees the *optimal* solution. One of the best known methods for graph and subgraph isomorphism detection is based on depth-first backtracking search (Corneil and Gotlieb[6]). The number of backtracking steps in a search tree can be greatly reduced by a combination of backtracking and forward checking (Ullman[45]). Another approach for subgraph isomorphism detection is based on building a so called association graph and searching for maximal cliques in such a graph (Falkenhainer et al.,[11] Horaud and Skorda,[18] Myaeng and Lopez-Lopez,[31] and Pelillo et al.[34])

The problem of *error-correcting* subgraph isomorphism consists of finding the sequence of edit operations with *minimal* cost such that a subgraph isomorphism

PREPRINT, published in
International Journal of Pattern Recognition and Artificial Intelligence,
Vol. 17, No. 7, pages 1279–1302 (2003)

exists (Messmer and Bunke[29]). Most of these algorithms are based on the A^*-algorithm (Nilsson[32]). By introducing a heuristic cost estimation function, the size of the search tree can be greatly reduced. Various cost functions have been proposed (e.g, Tsai and Fu,[44] Shapiro and Haralick,[39] Fu,[13] Sanfeliu and Fu,[38] Eshera and Fu,[9] and Wong.[49])

All algorithms in the first group are guaranteed to find the graph and subgraph isomorphisms. The *second group* is based on *probabilistic* approximation methods. They generally have a lower computational complexity, but do not guarantee to find the best solution. The algorithms are based on relaxation (e.g., Rosenfeld et al.,[36] Peleg,[33] Ton and Jain,[42] Christmas et al.,[5] and Gold and Rangarajan,[15]) neural networks (e.g., Kuner and Ueberreiter,[22] Yu and Tsai,[53] and Suganthan et al.,[41]) linear programming (Almohamad and Duffuaa[1]), genetic algorithms (Krcmar and Dhawan[21]), or Lagrangian optimization (Rangarajan and Mjolsness[35]).

Most systems for object recognition and model-based scene analysis rely on *point matching* or the estimation of corresponding point pairs in a stored *model* (or prototype) and an *image* to be analyzed.

In the first instance, matching algorithms rely on feature comparison. Visual features, which are robust under the changes that are expected between model and image, are extracted on both sides and their similarity yields a first hint at possible candidates for correspondences. It is well known that correspondence cannot be established on the grounds of feature similarity alone, and the *relative position* of features must also be taken into account.

As a consequence of this dual constraint, *attributed graphs* present an ideal data format for models. Points to be matched can be identified with graph vertices, and the connecting graph edges can be used to code the relative positions. Any useful further information about point features can be conveniently coded as vertex attributes.[43]

The corresponding graph in the image can be found in various ways. One possibility is to represent the image as a matrix of point features and choose vertices in the image by optimizing a cost function which combines some average feature similarity with the deviation from an undeformed model graph. This approach has been quite successful for face recognition.[23,47] It is well suited for problem classes whose feature values vary more or less smoothly over the image.

A different approach is to represent both model and image as graphs. This is attractive, because the problem is transformed into a purely combinatorial one, which makes it susceptible to complexity analysis. In the problem's purest form, the image graph must be searched for a subgraph isomorphic to the model graph. In full generality, this problem is known to be NP-complete in the number of image vertices,[14] which makes it doubtful that such algorithms will be efficient in real-world applications. However, the complexity is potentially greatly reduced by the requirement of feature similarity and by further properties of the graphs involved. It can be shown that subgraph isomorphism for *planar graphs* without attributes

PREPRINT, published in
International Journal of Pattern Recognition and Artificial Intelligence,
Vol. 17, No. 7, pages 1279–1302 (2003)

can be computed in a time proportional to the number of image vertices.[8]

For practical systems, the situation becomes more complicated, because robustness in the presence of feature variations and eventually missing edges and vertices become crucial. Additional requirements are necessary, which usually prohibit theoretical analysis of the algorithmic complexity. Furthermore, the constant factors governing the matching time can no longer be ignored. Both turn the efficiency of graph-matching algorithms for correspondence largely into an experimental matter.

In this paper, we test subgraph matching for the analysis of scenes, which may contain several instances of the objects in the database. We focus on corner and contour matching, no surface features are taken into account. Since we want to recognize all objects in the database that differ at most an a-priori known error-value from the exact object in the image, we have the problem of error-tolerant subgraph isomorphism. The algorithm we use belongs to the first group, since we want all possible objects in the image graph to be found. Images as well as stored models are represented as graphs whose vertices correspond to object corners and whose edges code for edges connecting corners in the image. The method is invariant under translation, rotation, and scaling and robust under changes in background, limited changes in perspective, and small distortions, but cannot handle occluded corners.

For model matching it is assumed that corners can only be matched onto corners and connecting edges must match edges in the image. This makes the process equivalent to finding subgraph isomorphisms. In our approach, the complexity is reduced by demanding that the labels of matched vertices and edges, respectively, be roughly equal. We will show experimentally that the problem is so reduced to a tractable size.

Because of the representational power of graphs, much effort has gone into the development of efficient matching algorithms. A problem that has been tackled party is the *extraction* of edges and corners from a two dimensional grid of picture elements. However, a fully symbolic representation is necessary for recognition by graph matching. Widely applied techniques for edge detection in image processing are based upon enhancement of edges followed by thresholding. After that an extraction algorithm should be applied to obtain a symbolic representation. Methods that are well known and often used in computer vision that are based on this technique are from Canny[4] and more recently from Smith and Brady.[40] The problem of such an approach is setting a threshold, which is different for every image and that is called a *plague* by Faugeras.[12] To his knowledge thresholding is unavoidable, and should be tackled with courage. Very recently, Jarvis[20] stated that even when sophisticated edge extraction and linkage algorithms are used, *gaps* in contours can severely disrupt the segmentation result. Domain-specific knowledge can help bridge the gaps, but restricts the scope of applicability. In this paper a greedy edge contour following algorithm will be proposed that avoids manual thresholding and length of contour settings for every different image. The algorithm, that uses

PREPRINT, published in
International Journal of Pattern Recognition and Artificial Intelligence,
Vol. 17, No. 7, pages 1279–1302 (2003)

detected corners and enhanced contour images, accurately finds the contours along lines and edges and guarantees closed contours. Algorithms for edge extraction based on the following principle are the border-tracking algorithm[16] and following as graph searching,[2] but need further complicated processing to bridge gaps.

The corner extraction method, which is described here only briefly, is based on a model for end-stopped cells in the visual cortex and thus models some aspect of human vision. Recently, models start to evolve that present neuronal algorithms for contour following.[46] A neuronal model for graph matching was proposed.[23,48] Although the evidence is still sketchy, it can be expected that the whole algorithm allows a neural implementation and thus potentially models aspects of human object recognition, for which corners and edges play an important role. For example, Biedermann[3] showed that partial contour deletion only impedes object recognition if it is accompanied by altering corner attributes.

2. Symbolic contour graphs

The transformation from the input image, which is a two-dimensional grid of pixels into completely different graph representation is a complicated task that requires several steps.

Initially, Gabor filters that are closely related to the function of simple cells in the visual cortex of primates are used.[7,27] Sequentially the energy of the amplitude of the Gabor filters is taken that is related to a model of complex cells which combines the responses of a phase pair of simple cell responses. In turn the end-stopped cells, that enhance line-ends and corners, receive input from the complex cells. A local maximum operator is used to extract the corners. The set of extracted corners represents the set of vertices in the graph.

The set of corners together with the modeled complex cell responses form the input for the contour extraction algorithm. A greedy algorithm where a contour along the strongest responses of the complex cell operator is followed from one corner to another. This corner pair forms an edge, but since contours are not necessarily straight we add attributes to the graph. We keep the complete contour represented by a Freeman chain as additional edge attribute. This attribute is of importance if we want to compare the similarity of two curves. We used normalized surface matching as additional matching criterion, that will not be discussed in this paper.

When the complete graph is extracted additional attributes such as relative length of edges and angles between edges connecting the same corners are added to the graph. These attributes (ignoring, scale, translation, and rotation) guarantee that we can make a frame of the original input image and by filling the surfaces in between even a reconstruction of the original input image. Another advantage of these attributes is that they strongly reduce the search space. When the first two corners of an object, stored in a database, are matched with two different corners in the image graph, all the other corners of the object have a fixed position in the

PREPRINT, published in
International Journal of Pattern Recognition and Artificial Intelligence,
Vol. 17, No. 7, pages 1279–1302 (2003)

image graph. Hence, if one would be searching exact copies of a known object, the algorithm would be of order $O(V^2)$, where V is the number of corners. In practice one will tolerate small differences in length and angle, which means that all corners in the image graph within a certain radius from the exact point will be accepted as a partial match. The differences in angle and length should be rather small otherwise accepted matches can be so strongly deformed that they are not being perceived (by a human observer) as the same object.

2.1. *Complex cells*

We will describe objects and scenes as graphs with corners as vertices and contours as edges. Corner detection and edge following are based on a Gabor wavelet transform of the image. Complex-valued Gabor functions at scale σ and orientation θ are defined as

$$\hat{G}_{\sigma,\theta}(x,y) = \exp\left(i\frac{\pi}{\sqrt{2}\sigma}(x\cos\theta + y\sin\theta)\right)\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \quad , \tag{1}$$

These Gabor functions have been modified such that their integral vanishes and their one-norm (the integral over the absolute value) becomes independent of σ, resulting in $G_{\sigma,\theta}(x,y)$. They provide an transform of the image $I(x,y)$ via spatial convolution. Afterwards, only the amplitudes of the complex values are retained for further processing:

$$\mathcal{C}_{\sigma,\theta}(x,y) = |I * G_{\sigma,\theta}| \quad . \tag{2}$$

This "local energy representation" is the basis of all subsequent processing (for analytical properties see Wundrich et al.[50]). A high value at a certain combination of x, y and θ represents evidence for a contour element in the direction orthogonal to θ. Orientations and scales are sampled linearly: $\theta_i = \frac{i\cdot 180}{N}, i = 0\ldots N-1$, $\sigma_j = \sigma_0 + j\Delta\sigma, j = 0\ldots S-1$.

2.2. *Robust corner detection*

Starting from that representation, we have developed a biologically motivated method for corner detection,[52] which is described here only briefly. Our method for detecting corners yields position, sharpness, size and color and contrast. The subtended angle can be determined a posteriori by following the line segments that constitute the corner (see section 2.3). It is based on a model of cortical end-stopped cells.[17]

The first step towards an end-stopped operator is an approximation of the first derivative of the $\mathcal{C}$-operator in the direction orthogonal to that of the line segment in question:

$$\begin{aligned}\hat{\mathcal{E}}^s_{\sigma,\theta}(x,y) = \mathcal{C}_{\sigma,\theta}(x + d\sigma\sin\theta, y - d\sigma\cos\theta) - \\ \mathcal{C}_{\sigma,\theta}(x - d\sigma\sin\theta, y + d\sigma\cos\theta) \quad ,\end{aligned} \tag{3}$$

PREPRINT, published in
International Journal of Pattern Recognition and Artificial Intelligence,
Vol. 17, No. 7, pages 1279–1302 (2003)

and second derivative:

$$\begin{aligned}\hat{\mathcal{E}}^d_{\sigma,\theta}(x,y) = \mathcal{C}_{\sigma,\theta}(x,y) - \\ 0.5\mathcal{C}_{\sigma,\theta}(x + 2d\sigma\sin\theta, y - 2d\sigma\cos\theta) - \\ 0.5\mathcal{C}_{\sigma,\theta}(x - 2d\sigma\sin\theta, y + 2d\sigma\cos\theta)\ .\end{aligned} \tag{4}$$

These two operators are both inhibited by a tangential and a radial inhibiting operator:

$$\mathcal{I}^t_\sigma(x,y) = \sum_{i=0}^{2N-1} \left[-\mathcal{C}_{\sigma,\theta_{i \bmod N}}(x,y) + \mathcal{C}_{\sigma,\theta_{i \bmod N}}(x_1,y_1)\right]^{\geq 0} \tag{5}$$

and

$$\mathcal{I}^r_\sigma(x,y) = \sum_{i=0}^{2N-1} \left[\mathcal{C}_{\sigma,\theta_{i \bmod N}}(x,y) - w_r \mathcal{C}_{\sigma,\theta_{\left(i+\frac{N}{2}\right) \bmod N}}(x,y)\right]^{\geq 0}\ , \tag{6}$$

where $x_1 = x + d\sigma\cos\theta_i$, $y_1 = y + d\sigma\sin\theta_i$, $[z]^{\geq 0}$ is equal to 0 for negative z and equal to z elsewhere (half-wave rectification), and constant $w_r = 4$. The corner operators on a single scale in a single direction then are:

$$\mathcal{E}_{\sigma,\theta_i} = \left[\left[\hat{\mathcal{E}}_{\sigma,\theta_i}\right]^{\geq 0} - g(\mathcal{I}^t_\sigma + \mathcal{I}^r_\sigma)\right]^{\geq 0}\ . \tag{7}$$

Constant $g = 2$ is a gain factor and for $\mathcal{E}$ one can substitute $\mathcal{E}^s$ or $\mathcal{E}^d$. For details and motivation of constants, see Lourens and Würtz.[24,52]

At each point we consider only the maximum over all orientations, and also the maximum of single and double end-stopped operators:

$$\mathcal{E}_\sigma = \max_{i=0}^{2N-1} \max(\mathcal{E}^s_{\sigma,\theta_i}, \mathcal{E}^d_{\sigma,\theta_i})\ . \tag{8}$$

Sharp corners are characterized by strong responses over a wide frequency range. If only high frequency cells respond, the feature is likely to be noise or texture rather than a corner. We found that averaging the responses over a range of frequencies yields a much more robust corner detection:[51,52]

$$\mathcal{E}_{\text{avg}}(x,y) = \frac{1}{S}\sum_{j=0}^{S-1} \mathcal{E}_{\sigma_j}(x,y)\ . \tag{9}$$

With a slight and biologically justified extension of the concept of complex cells, corner detection can be extended to color channels (red-green and blue-yellow opponent), which is described in detail in Würtz and Lourens.[51,52] The amplitude of $\mathcal{E}_{\text{avg}}$ for each of the red-green, blue-yellow, and grey value channels yields the final corner detector $\mathcal{E}_{all}$. The $\mathcal{I}^r$-operator (6) has been improved compared to former work;[51,52] it now yields better results at corners with large angles. Currently angles up to 140° are recognized compared to a maximum of 120° before, while the remaining properties are preserved.

PREPRINT, published in
International Journal of Pattern Recognition and Artificial Intelligence,
Vol. 17, No. 7, pages 1279–1302 (2003)

2.3. *Contour extraction*

The idea of contour following is, to select corners and local edge maxima as starting points, and then to follow a contour to another corner or local maximum by selecting the strongest edge responses during the following process. Such a mechanism might exist in the brain as well. A group of cells, which are coined linking cells,[25] receive their inputs from the complex and endstopped cells. Initially, only the strongest local responses of the complex and endstopped cells will trigger the linking cells at the same spatial positions. These activated neurons will activate their neighbor with the strongest complex input response; both will start firing in synchrony. In turn, these newly activated neurons will activate an inactive neighbor, and a cascade of synchronously firing neurons along the edge contours will be the final result.

In order to follow a contour, we combine the various orientations, scales and colors in the same way as for the end-stopped operators:

$$\mathcal{C}_\sigma = \max_{i=0}^{N-1} \mathcal{C}_{\sigma,\theta_i} \quad , \tag{10}$$

$$\mathcal{C}_\sigma^{\text{all}} = \sqrt{\mathcal{C}_\sigma^2 + \left(\frac{1}{2}\mathcal{C}_\sigma^{r,g}\right)^2 + \left(\frac{1}{2}\mathcal{C}_\sigma^{b,y}\right)^2} \quad , \tag{11}$$

where $\mathcal{C}_\sigma^{\text{all}}$ is the amplitude over the three, two chromatic and one achromatic, channels.

Globally the algorithm, illustrated in Figure 1, contains 3 stages:

(1) detecting corners and local edge maxima by thresholding (lines 2-3); the initial stage of a two dimensional layer of linking cells
(2) extracting contours starting at local maxima (lines 4-7) and corners (lines 8-12); activating and linking neighbors along an edge contour
(3) connecting corners with edge contours and each other (lines 13-16); also activating and linking neighboring neurons along an edge contour

This contour following method, which is described in more detail in Lourens et al.,[26] is more sophisticated than extracting binarized responses of appropriate edge detectors, because the latter are very sensitive to changes in local contrast and size of thresholds. Furthermore, it can be tailored to special situations by taking the concrete distribution of Gabor energy over σ and θ into account locally. Our method starts at corners and collects evidence for a contour to connect this corner with another one. Thus, the resulting contour graphs can be called "symbolic".

2.4. *Evaluation of edge extraction*

The proposed method is compared with two other methods: the well known Canny edge detector[4] combined with the curvature scale space detector,[30] and with the SUSAN edge and corner detector.[40] Although edge detection and extraction are not the same, the results can be compared by displaying the extracted data in a

PREPRINT, published in
International Journal of Pattern Recognition and Artificial Intelligence,
Vol. 17, No. 7, pages 1279–1302 (2003)

two-dimensional image. For both Canny-CSS and SUSAN the thresholds were set manually for every image to obtain good results.

Extraction of contours in both Canny-CSS and SUSAN, in practice, is very difficult. This is caused by gaps, overlapping contours, or multiple detection of a single contour, as illustrated in Figure 2. Even in the synthetic P image the contours near the corners are not closed. In case of noise or speckle (or textures that have a similar effect) these methods underperform due to small filter kernels and high contrast sensitivity.

The advantage of the proposed model over other two models is that the same low constant thresholds for corner and local contour maxima detection are used for different images.[26] Closed contours are guaranteed by the contour following algorithm. The results of the extracted contours (when displayed as being detected) show similar or better results compared to the SUSAN and Canny-CSS detectors.

2.5. *Graph attributes*

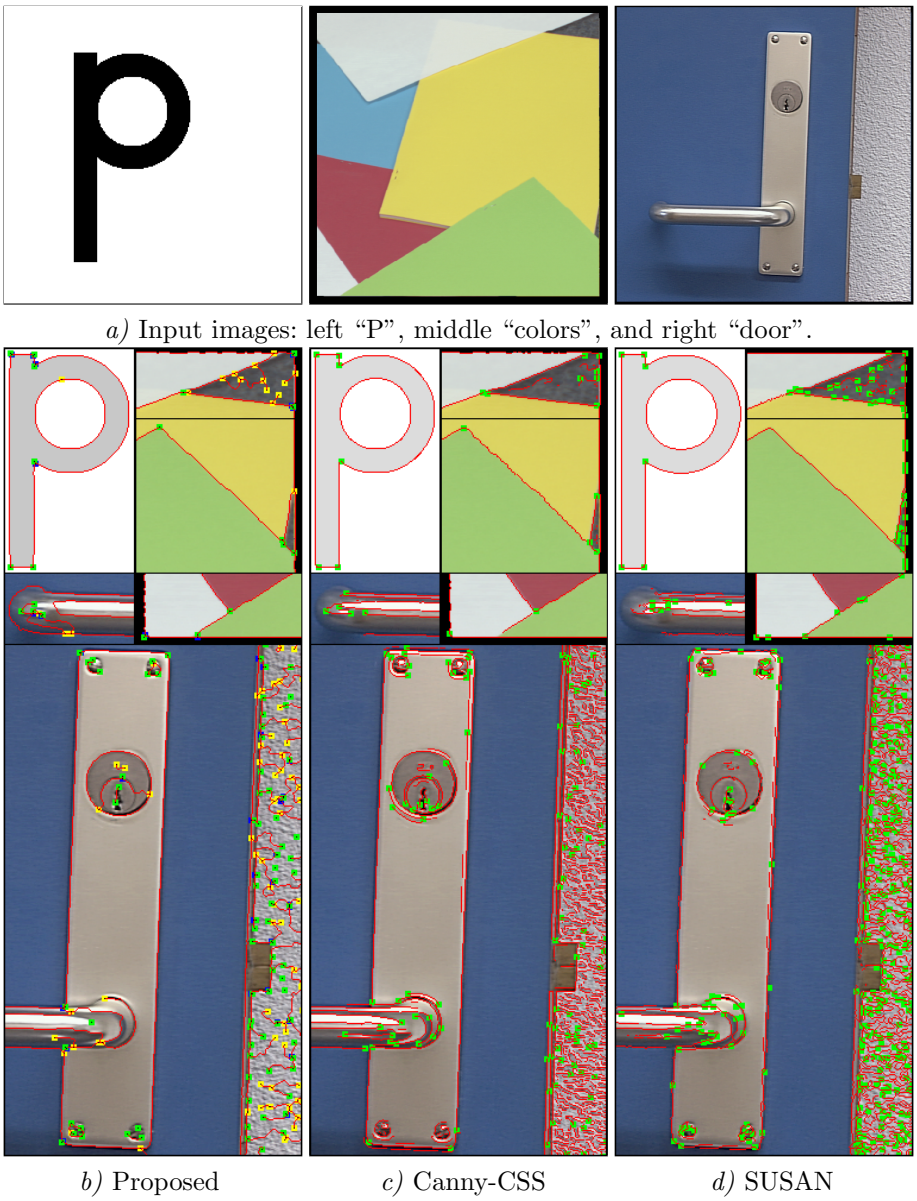
After contour following, we end up with a graph that has corners as vertices and contours described by a series of points as its edges. After graph extraction, one optimization step is performed: removal of multiple extracted contours. Additional optimizations can be added, like deleting corners and partly double detected contours, to further improve the graph.

Once stored models and the image to be analyzed are represented in this way, model matching can be done by finding a copy of the model graph in the image graph.

```
1  Function ExtractAttributedGraph (E_ω^all, C_σ^all)
2      C := Set of corners obtained by taking local maxima from E_ω^all
3      M := Set of local edge maxima obtained from C_σ^all
4      forall m ∈ M
5          forall n ∈ BestNeighborSelectedByResponseAndOppositeCoord (C_σ^all, m)
6              l := ExtractContour (C_σ^all, C, M, m, n)
7              Add (L, l)
8      forall c ∈ C
9          forall n ∈ EightNeighbors (c)
10             l := ExtractContour (C_σ^all, C, M, c, n)
11             Add (L, l)
12         RemoveDoubleDetectedContours (L)
13     forall l ∈ L
14         forall c ∈ C
15             l_c := ConnectCornerToContour (l, c, dσ)
16             Add (L_c, l_c)
17     G := CreateAttributedGraph (L ∪ L_c)
18     return G
```

Fig. 1. Algorithm for graph extraction. Variable ω is σ or avg.

PREPRINT, published in
International Journal of Pattern Recognition and Artificial Intelligence,
Vol. 17, No. 7, pages 1279–1302 (2003)



a) Input images: left “P”, middle “colors”, and right “door”.

b) Proposed *c)* Canny-CSS *d)* SUSAN

Fig. 2. *a)* Top row input images used as benchmarks and bottom row enlarged results of parts of the input images. *b)* Results obtained with proposed method using $\sigma = 2.36$ and default parameter settings. Corners are represented by green squares and detected contours are marked in red. Blue squares are corner edge connections. Yellow squares denote a junction (due to extraction) or the end of a contour. *c)* Results of the Canny edge detector combined with the CSS corner detector. *d)* Results of the SUSAN edge and corner detector.

PREPRINT, published in
International Journal of Pattern Recognition and Artificial Intelligence,
Vol. 17, No. 7, pages 1279–1302 (2003)

Each corner is labeled with the angles between all pairs of adjacent line segments starting from it. The edges are labeled with the *relative length* of the line segments, i.e., the ratio of the length to the length of the longest line segment in the whole graph.

An example model graph is shown in Figure 3a. The extracted corners are known by their (x, y)-coordinates. To keep a graph *translation invariant* these are not directly suitable for attributes, but only the *relative* positions may be used. This is obtained by taking the *lengths of the contours* and the starting angles between contours at corners. This makes the representation *rotation invariant* as well. Finally, to make the graph be scale independent we choose the lengths relative to each other. This is illustrated in Figure 3b, where the edge with the longest contour is assigned 100.00 and the others their contour's length as percentage of the longest one. Angles are measured counterclockwise: as a consequence, we have, e.g., in Figure 3b that the angle at vertex 0 is different from the angle at vertex 1, although the graph is symmetrical in the vertical axis. The choice of these attributes keeps the graph invariant under translation, rotation, and scale, but not under mirror reflection. The attributes of the graph are represented in a table, an example is shown in the lower table of Figure 3c.

3. Graph matching

A depth first tree search algorithm is used, and partial solutions are pushed on a stack. At every vertex at least one evaluation takes place, but multiple evaluations $(N-1)$ at a single vertex can take place depending on the number of edges N that start from a vertex in the model graph. For example two evaluations take place at vertex 2 and 3 in the model graph of Figure 3c. In our experiments we measured the complexity in pushes. The time between a push and a pop from the stack is small because only a few criteria are evaluated (see Figure 4), hence it gives the most objective measurement. Often the complexity is measured in the number of states in the search tree, which is equal or smaller than the number of pushes, depending on the different number N of edges starting at the vertex.

In common, to cut down evaluation expenses attributes as described above are assigned to vertices and edges. To cut down evaluation expenses in graph matching often only the best matching copy is searched. This is not acceptable here, because the same object may appear several times in the image graph. Consequently, we require to find *all* copies of a model graph G_m in the given image graph G.

3.1. *Cost function*

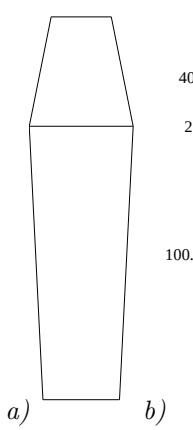
The standard representation of a graph is the Boolean adjacency matrix B. Its elements $B(i, j)$, $i < j$ and $i, j \in V$, of this matrix are chosen to be true if and only if $(i, j) \in E$, and false otherwise. We generalize it by replacing the Boolean with a real-valued *cost function* $EC(v_1, v_2)$, $v_1, v_2 \in V$, which defines the cost of an edge missing in the image graph. During the process of matching we sum the costs

PREPRINT, published in
International Journal of Pattern Recognition and Artificial Intelligence,
Vol. 17, No. 7, pages 1279–1302 (2003)

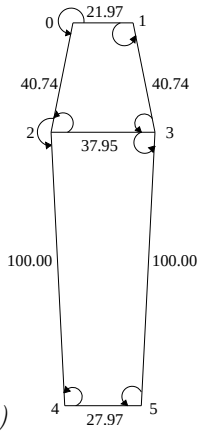
for these tolerated missing edges. This accommodates cases where missing edges in certain places are considered less severe than in others. Since we use undirected graphs we only need the lower- or upper-diagonal of the adjacency matrix with $\frac{1}{2}\#V(\#V-1)$ elements, where $\#V$ denotes the number of vertices in V, throughout this paper.

To cut down evaluation costs during matching we try to evaluate as little as possible, while allowing inexact matches and without losing any solutions. During the matching process we tolerate one or more missing edges in the image graph. Only inexact matches with a total cost less than an a-priori known cost are considered.

The flexibility of the edge matching reflects the philosophy that contours are often weak in real images, so contour following may fail in the image. It is, however, assumed that the model is known well enough, i.e., images of sufficient quality are available, for all relevant contours to be present in the image graph. Beside the added robustness this flexibility yields a method of processing unconnected model graphs by adding extra edges. See Figure 6 for examples. Unconnected components of model graphs might attain arbitrary relative positions in our coding, and the



a)



b)

c)

0						
1	0					
2	0	∞				
3	∞	0	0			
4	∞	∞	0	∞		
5	∞	∞	∞	0	0	
	0	1	2	3	4	5

angle in degrees	at vertex	from edge	to edge	edge ratio
258.69	0	(0, 1)	(0, 2)	0.54
101.31	1	(1, 0)	(1, 3)	0.54
78.69	2	(2, 3)	(2, 0)	0.93
194.17	2	(2, 0)	(2, 4)	0.41
78.69	3	(3, 1)	(3, 2)	1.07
87.14	3	(3, 2)	(3, 5)	0.38
92.86	4	(4, 5)	(4, 2)	0.28
92.86	5	(5, 3)	(5, 4)	3.58

Fig. 3. *a)* An example of a model graph. *b)* The same graph augmented with edge attributes, which are the (relative) length of a contour and vertex attributes, which are the angle between two contours. *c)* Two tables are used to represent the attributes of the graph. At the top the adjacency matrix is displayed, zero denotes that there is an edge and ∞ denotes that there is no edge between the two vertices connected by the edge. The table at the bottom gives the angular attribute plus the ratio between the lengths of two contours as a percentage of the length of the "from contour" by the length of the "to contour".

PREPRINT, published in
International Journal of Pattern Recognition and Artificial Intelligence,
Vol. 17, No. 7, pages 1279–1302 (2003)

matching algorithm also assumes connectedness.

3.2. *Attribute similarity thresholds*

In the lower table of Figure 3c, the vertices are augmented with a set of angle and ratio attributes. In the matching process these attributes are used to reduce the search space. When a vertex is matched we tolerate an angle and a ratio at the vertex in the image graph to differ by at most a known constant angle and a known constant ratio from the corresponding angle and ratio in the model graph. Additionally we allow the average angle difference of all angles to differ at most a known constant angle. Similarly the average ratio difference is also bounded by a constant. We introduce four bounding constants:

(1) the angle tolerance,
(2) the average angle tolerance,
(3) the ratio tolerance, and
(4) the average ratio tolerance.

We assume that the model graphs are either constructed by hand or extracted from "clean" images. Thus, they are supposed to contain both all corners and *all* edges.

3.3. *Graph matching algorithm*

The algorithm for graph matching is illustrated in Figure 4. It can find all copies of a model graph G_m in an image graph G. Lines 2–11 are the initial stage of the algorithm, we start with an empty stack and push all vertices of the image graph onto the stack one after another, since each of them can, in principle, be matched with the first vertex of the model graph. Lines 12–30 constitute the matching proper. In line 13 we take a possible partial solution from the top of the stack and check if the match is already complete (line 14). If so, the maximal and average relative length differences to the model are calculated; if both are within the bounding constants the match is accepted and displayed. If the match is incomplete the process continues at line 17. Here we check if the current angle and ratio can already be evaluated. If we can not evaluate because one or both vertices to form the angle are still missing, then the missing model vertex is parsed by adding it to the list (line 26) and all possible vertices in the image graph (lines 27–30) are searched. A vertex v is added if it is not yet matched and if the cost EC of adding edge (L_{cv}, v) is smaller than the allowed cost.

The speed of the algorithm mainly depends on the condition in line 18. If angle and ratio differences are chosen properly most of the partial matches will be rejected here, and the path is rejected by not pushing it back on top of the stack. During matching, the difference in ratio $\delta r = \max\left(\frac{r_m}{r}, \frac{r}{r_m}\right)$ (where r_m denotes the ratio of an edge pair in the model graph and r denotes the corresponding matched ratio in the image graph) between two pairs of edges is obtained by scaling the edge pairs

PREPRINT, published in
International Journal of Pattern Recognition and Artificial Intelligence,
Vol. 17, No. 7, pages 1279–1302 (2003)

in such a way that the first edge of both pairs is one, then the ratio is the size of the rescaled second edge of the first pair: the size of the rescaled second edge of the second pair. The average length difference is evaluated by using the relative lengths of both model and found match in the image graph, as described earlier in this paper. The absolute difference of the model edge with its corresponding image edge is taken, when there was no edge between v_a and v_b in the image graph we used the relative length of $\mathrm{dist}(v_a, v_b)$. The average of all edges is taken to represent the average relative length difference.

4. Results

Figure 5 shows a color image and the extracted image graph. We have used the model graph illustrated in Figure 3b to find all markers in the image.

The result is illustrated in Figure 5c. We allowed at most three (by setting the cost function EC to one for every missing edge) out of the seven edges in the model graph to be added and a maximal δr of 5. We tolerated an angle difference and an average angle difference of at most 36° (which in practice is far too large, hence results might be strongly deformed that they are not being recognized by a human anymore). When a match is found we tolerate a maximum relative length difference of 50% and an average relative length difference of 10%. The matching time for the image graph is less than 0.1 second on a standard PC.

```
1  Procedure MatchGraph (G, MG)
2      stack := ∅
3      forall v ∈ V(G)
4          L_0  := v                          /* list of parsed vertices of image graph */
5          ML_0 := first model vertex         /* list of parsed vertices of model graph */
6          mv   := 1                          /* number of matched vertices             */
7          cv   := first model vertex         /* vertex being evaluated                 */
8          cva  := first angle of cv          /* angle of cv to be evaluated            */
9          ED   := 0.0                        /* accumulated edge difference            */
10         AD   := 0.0                        /* accumulated angle difference           */
11         Push (mv, cv, cva, L, ML, ED, AD)
12     while stack ≠ ∅
13         Pop (mv, cv, cva, L, ML, ED, AD)
14         if mv = cv = #V(MG)                /* Match found                            */
15             Evaluate maximum and average relative length differences
16         else
17             if Angle cva of vertex cv can be evaluated
18                 if Evaluation accepts angle and ratio
19                     if cva = last angle of cv
20                         cv := next (cv)
21                         cva := first angle of cv
22                     else
23                         cva := next (cva)
24                     Push (mv, cv, cva, L, ML, ED, AD)
25             else                           /* Add a missing vertex                   */
26                 ML_mv := missing-vertex
27                 forall v ∈ V(G) − L        /* Unused vertices only                   */
28                     L_mv := v
29                     if ProperVertex (EC, max ED − ED)
30                         Push (mv + 1, cv, cva, L, ML, ED + EC, AD)
```

Fig. 4. Algorithm for graph matching. See section 3.3 for explanation.

PREPRINT, published in
International Journal of Pattern Recognition and Artificial Intelligence,
Vol. 17, No. 7, pages 1279–1302 (2003)

5. Application

The crucial question is how the execution time scales with the number of vertices in the image graph. We measure the speed in the number of pushes occurring in lines 11, 24, and 30 of Figure 4, because this number is independent of machine and implementation details. For translating the values into real time, over one million pushes can be evaluated per second on a PC with a 2 GHz Pentium 4 processor.

In this section we present a case study to get insights in the actual matching time needed for graphs up to 250 vertices. Our choice is to recognize traffic signs in a normal environment. For our experiments, we used 39 different image graphs and three model graphs (Figure 6). Among the image graphs 26 contained one or more traffic signs, the other 13 were without traffic signs. Extra edges in the model graphs of Figure 6a-b are needed to retain the relative position and size of the different parts of the graph.

In the simple case of the marker recognition, we have used a simple cost function

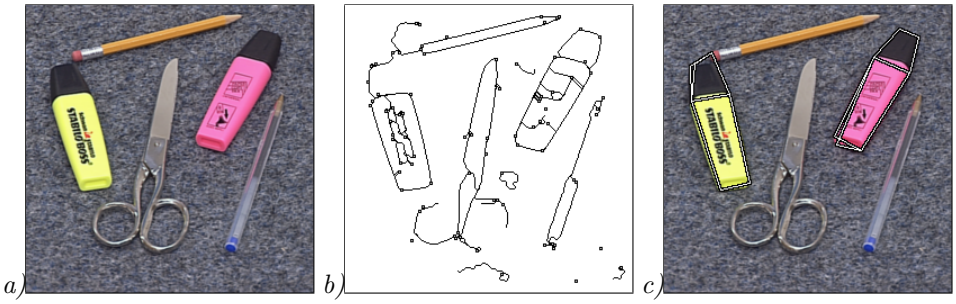


Fig. 5. *a)* Input image. *b)* Image graph extracted from *a)*. The small squares denote detected corners. In the graph an additional chain attribute is used to describe the exact curve of the contour. *c)* Found matches of the markers. Parameters values are: maximally three edges to be added, ratio between two edge pairs $\delta r = 5$, average angle tolerance 10%, maximal angle tolerance 10%, maximal length tolerance 50%, and an average length tolerance 10%.

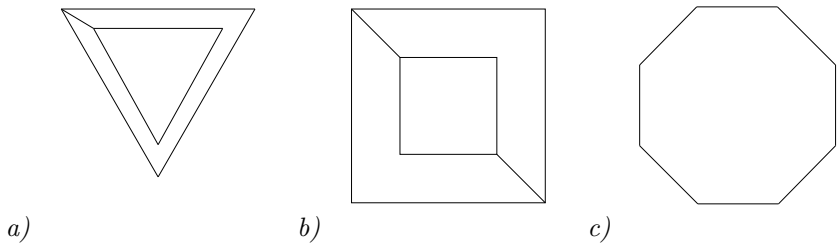


Fig. 6. The three model graphs used. *a)* Danger sign, *b)* Precedence sign, *c)* Stop sign. As the method requires connected model graphs, extra edges have been introduced in *a)* and *b)*.

PREPRINT, published in
International Journal of Pattern Recognition and Artificial Intelligence,
Vol. 17, No. 7, pages 1279–1302 (2003)

(one per missing edge) for the image graph. For the traffic signs this has been extended as follows. The cost function for an existing edge is still zero, but the cost for a non-existing edge between a and b is one if there exist a chain between a and b, i.e. there are one or more vertices v_i on or near the imaginary line-segment (a, b) for which there exists edges between (a, v_0), (v_0, v_1), ..., (v_n, b). If there is no edge and no chain between vertices a and b the cost for tolerating the edge is two plus the number of intersections (a, b) makes with existing edges. In common, this implies that adding short edges between a and b have a lower cost than longer edges. This choice of cost function decreases the matching time.

We have evaluated a total of 42 image graphs. Three of them are the model graphs transformed to image graphs. All three are (of course) recognized correctly. In the 13 graphs which do not contain traffic signs nothing is detected. In the other images all signs except one were detected. The undetected sign is due to a K-junction, which is not identified as a corner. This implies that a corner is missing and the match is not found. There are no falsely detected signs in any of the image graphs.

Four images of the experiments are illustrated in Figure 7. These images illustrate that recognition is invariant under translation, rotation, and scale. The evidence for rotation invariance can be obtained easily from the precedence model where two connecting edges are used, while the recognized image there are four connecting edges which is due to multiple recognition of the same object. Matching by corners and edges has the advantage that the method is robust under different lighting conditions (Figure 8), but require sufficient contrast to extract the corners and edges. The enlarged images illustrate clearly that the method is robust to noisy data.

5.1. *Resulting complexity*

Figure 9 shows the number of pushes required for finding all solutions for all 42 image graphs. The matching time clearly has an upper bound of $O(\#V^3)$. This is not what one would immediately expect since three vertices are needed for the evaluation of the first angle and ratio. This would suggest already $\#V^3$ but we should take into account that not every triple is possible. For example if one or both edges are missing the cost can be higher than we tolerate. For all three models we tolerate an edge cost of 15 which can be exceeded directly by the cost of tolerating the first edge, therefore a partial match containing a single vertex can be rejected.

On average 89 matches are obtained (but not all accepted) for the danger sign, which has six corners, seven edges, and eight angles (6-7-8). On average 39 matches are found for the stop sign (8-8-8). This model graph is one of the most time consuming examples that can be constructed with eight vertices, only a chain would be worse (8-7-6). The model graph of the precedence sign (8-10-12) on average has 8 matches. Extending model graphs with more vertices increases the matching time while the number of vertices is small but saturates for a larger number of vertices

PREPRINT, published in
International Journal of Pattern Recognition and Artificial Intelligence,
Vol. 17, No. 7, pages 1279–1302 (2003)

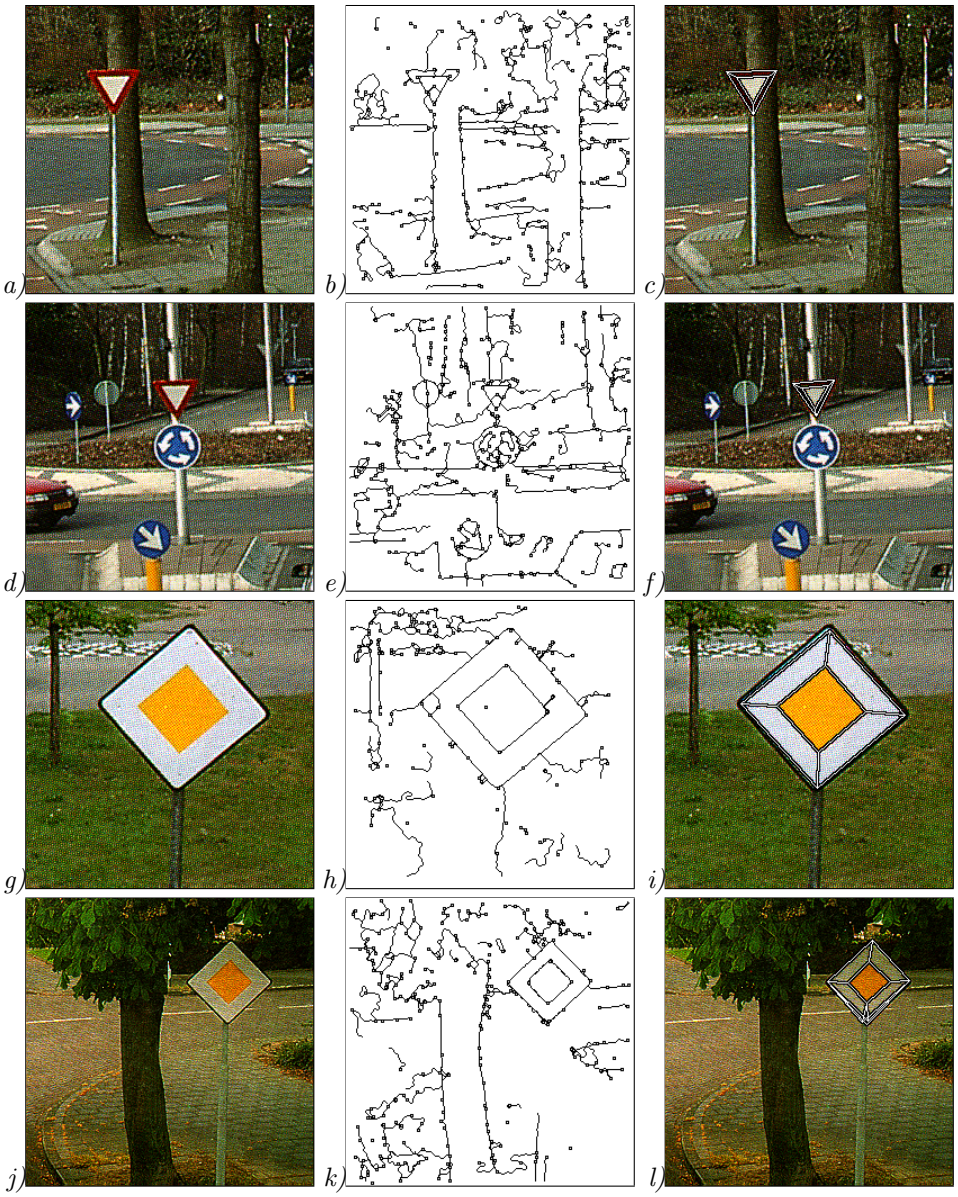


Fig. 7. Input images, extracted graphs, and found matches are shown in the left, middle, and right column, respectively. The following parameters have been used: the average angle error is 3, the average length error is 5, the maximal angle error is 5, the maximal length error is 14, the ratio tolerance is 1, and the maximal cost of missing edges is 15. The size ($\#V$, $\#E$) of the image graphs from top to bottom is (240,294), (247,351), (117,170), and (216,281).

PREPRINT, published in
International Journal of Pattern Recognition and Artificial Intelligence,
Vol. 17, No. 7, pages 1279–1302 (2003)

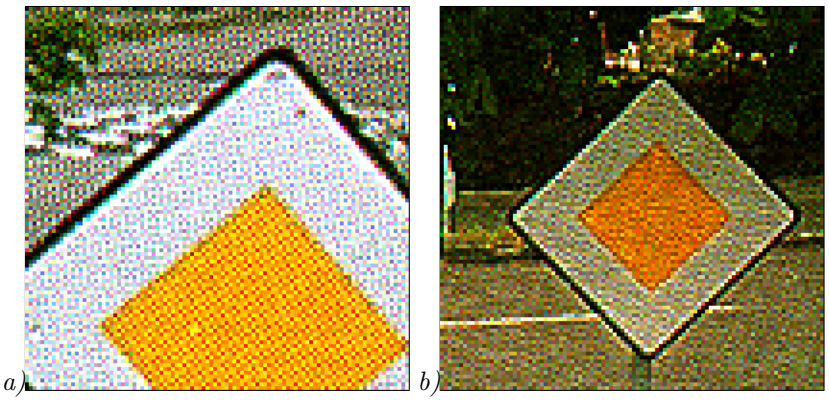


Fig. 8. Enlargements of Figure 7i and j.

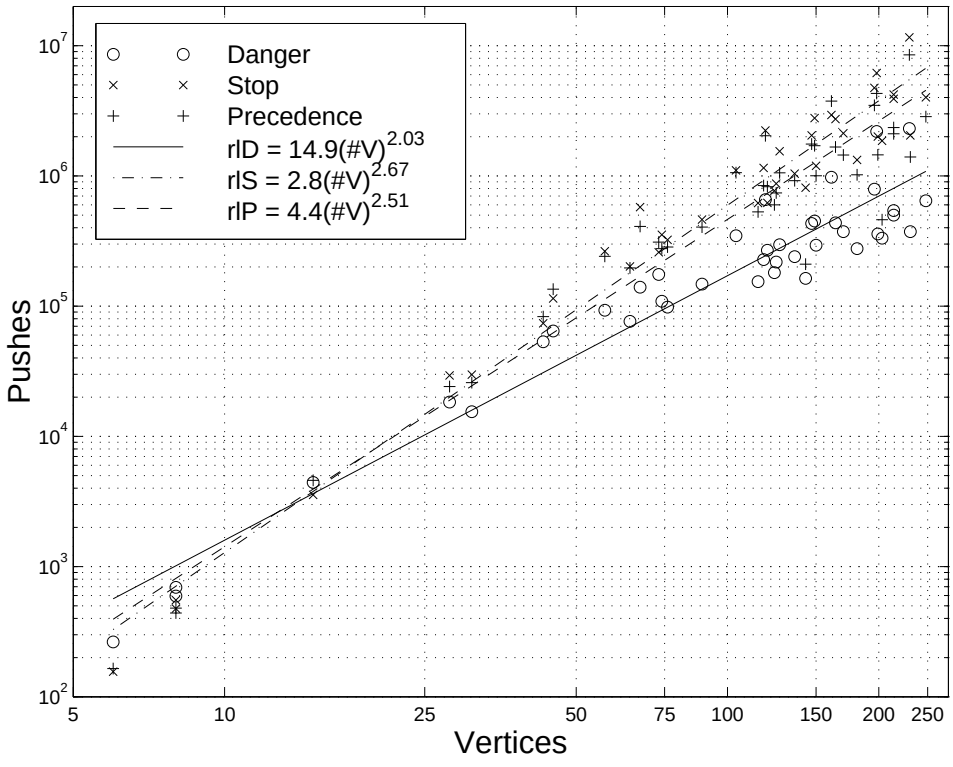


Fig. 9. Matching time measured in pushes and the regression lines rlD, rlS, and rlP of the three used model graphs. The correlation coefficients are 0.96, 0.98, and 0.97, respectively.

PREPRINT, published in
International Journal of Pattern Recognition and Artificial Intelligence,
Vol. 17, No. 7, pages 1279–1302 (2003)

since almost all partial matches are already rejected before the last vertices are evaluated. Extending the number of edges in the model graph will decrease the matching time. This implies that the matching time is not expected to increase dramatically with large model graphs.

In Figure 9 the regression lines indicate the average amount of pushes needed to match a model. The number of pushes needed is approximately $15(\#V)^{2.03}$, $3(\#V)^{2.67}$, and $4(\#V)^{2.51}$ for the danger, stop, and precedence model, respectively. All correlation coefficients were above 0.95, which implies that these regression lines are reliable and that increasing the set of images will hardly influence the results.

6. Related work

Most of inexact matching algorithms deal with error-correcting subgraph isomorphisms and very few with error-tolerating subgraph isomorphisms. The difference between the two is that the first always finds one subgraph isomorphism with a distance or cost and the second delivers all subgraph isomorphisms within a known distance or cost. Hence, it is difficult to compare the time complexity of these problems with each other.

Messmer and Bunke[29] used an error-tolerant subgraph isomorphism for detection and recognition of graphical symbols. Their algorithm tolerates missing vertices and edges, and can even merge vertices during matching. The angle attribute is used to increase the speed of matching, these angles are assigned to the edges and therefore a directed graph is necessary. As their focus is on organizing the model base, their results are not directly comparable but computing times seem to be on the same order of magnitude as for our algorithm.

Eshera and Fu[9,10] proposed an algorithm for the error-correcting subgraph isomorphism. They assigned attributes to vertices and edges to impose restrictions to the graphs. They called these graphs attributed relational graphs (ARGs), which use three different features types (straight line, arc, and curve) and have relations (joint, intersection, and facing) between these features. For these relations the distance and angle attributes are used. This algorithm is of polynomial time, approximately $O(l^3m^2)$, where l and m are the number of edges in the two graphs. The main problem of this algorithm is the preprocessing. It is difficult to match the segments with one of the three feature types.

More recently Gold and Rangarajan[15] used ARGs for image representation and analysis. Their probabilistic algorithm has a time complexity of $O(lm)$. They applied the algorithm to the error-correcting subgraph isomorphism problem to match an image containing a cup (model) with an image that contains a coffee pot and a cup. They manually marked 10 corners and 8 additional points in the model and approximately 40 corners and approximately 40 additional points in the image. They needed about 30 seconds to find the match. If we assume that we need $\#V^3$ pushes and that we can evaluate 10,000 pushes/sec. on a comparable machine then we need about 6 seconds to find the match. Thus, our algorithm is faster and

PREPRINT, published in
International Journal of Pattern Recognition and Artificial Intelligence,
Vol. 17, No. 7, pages 1279–1302 (2003)

guarantees to find an (inexact) subgraph isomorphism if there is one.

A complementary method is labeled graph matching which is applied successfully for, e.g. invariant face recognition.[23,47] In this method, vertices are assigned local texture elements and edges carry the geometrical information about relative locations. This method is limited to richly structured or textured objects like faces. Objects with homogeneous surfaces do not provide the sort of vertex labels required there and can only be matched using their contours.

7. Discussion

We have presented a graph matching scheme for object recognition based on corners and contours of objects. We have used a robust and biologically motivated operator to detect the corners. A relatively sophisticated algorithm has been used to follow lines between corners, which may allow to call these graphs symbolic information.

The problem of subgraph matching has been greatly simplified by assigning contour angles as vertex attributes and relative contour lengths as edge attributes. Although we currently can not make formal statements about the resulting complexity we have shown that the time requirements can be cut down to reasonable amounts for realistic problem sizes. The choice of labels makes the matching invariant under translations, rotations and scaling. When two isomorphic but different models are found, the similarity of the labels is used as as selection criterion. We conjecture that the worst-case complexity of the algorithm is $O(\#V^4)$, because under ideal conditions three matching point pairs determine the translation, rotation, and scaling involved, while the fourth is needed to check for further copies of the model.

We are currently testing the robustness of the algorithm on more images and are planning to introduce some extensions. The most serious limitations are that occluded corners or corners degraded enough for the corner detector to miss them impede the whole model matching. Also, the line following algorithm is restricted to the simple case of relatively straight lines starting and ending at corners (This causes the poor representation of the scissors in Figure 4b).

We did not use curved lines for matching but introduced intermediate points to achieve a proper description of the curve. Edge attributes can be extended to yield a more accurate description of the model and arrive at a more realistic cost of a match in the image graph. When additional coordinates are used we can apply a curve matching by surface difference algorithm.

These shortcomings show that the algorithm presented here does not constitute a general solution to the correspondence problem. On the contrary, it works only for objects with sufficiently many corners, no occlusion and is independent of surface texture. This makes it somehow dual to graph matching based face-recognition systems,[23,47] which concentrate on surface texture and do not attempt to match corners. The matching algorithms suited for these problems are very different in the sense that the latter relies on local optimization for point positions, while the

PREPRINT, published in
International Journal of Pattern Recognition and Artificial Intelligence,
Vol. 17, No. 7, pages 1279–1302 (2003)

one presented in this paper implements a combinatorial graph search. Nevertheless, they are related by building on the data format given by the magnitude of a Gabor transform.

A matching algorithm expected to bring computer vision anywhere close to human performance in object recognition and scene analysis will presumably need to integrate a large number of simple algorithms to arrive at robust solutions over a wide range of situations. The integration of the two matching methods is subject of current research, and we consider it as an important test case for integrating the very different procedures to yield a single result, which is much more robust than any of the individual ones.

References

1. H. A. Almohamad and S. O. Duffuaa. A linear programming approach for the weighted graph matching problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:522–525, May 1993.
2. D. H. Ballard and C. M. Brown. *Computer Vision*. Prentence-Hall Inc., 1982.
3. I. Biedermann. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94(2):115–147, 1987.
4. J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(9):679–698, November 1986.
5. W. J. Christmas, J. Kittler, and M. Petrou. Structural matching in computer vision using probabilistic relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):749–764, August 1995.
6. D. G. Corneil and C. C. Gotlieb. An efficient algorithm for graph isomorphism. *Journal of the ACM*, 17:51–64, 1970.
7. J. Daugman. Two-dimensional spectral analysis of cortical receptive field profiles. *Vis. Res.*, 20:847–856, 1980.
8. David Eppstein. Subgraph isomorphism in planar graphs and related problems. *Journal of Graph Algorithms and Applications*, 3(3):1–27, 1999.
9. M. A. Eshera and King-Sun Fu. A graph distance measure for image analysis. *IEEE Transactions on Systems, Man, and Cybernetics*, 14(3):398–408, May/June 1984.
10. M. A. Eshera and King-Sun Fu. An image understanding system using attributed symbolic representation and inexact graph-matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(5):604–618, September 1986.
11. B. Falkenhainer, K. D. Forbus, and D. Gentner. The structure-mapping engine: Algorithms and examples. *Artificial Intelligence*, 41:1–63, 1990.
12. O. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. The MIT Press, Cambridge, Massachusetts; London, England, 1993.
13. K. S. Fu. A step towards unification of syntactic and statistical pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5:200–205, March 1983.
14. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. Freeman, New York, 1979.
15. S. Gold and A. Rangarajan. A graduated assignment algorithm for graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(4):377–388, April 1996.
16. R. M. Haralick and L. G. Shapiro. *Computer and Robot Vision*. Addison-Wesley, 1992.
17. F. Heitger, L. Rosenthaler, R. von der Heydt, E. Peterhans, and O. Kübler. Simulation

PREPRINT, published in
International Journal of Pattern Recognition and Artificial Intelligence,
Vol. 17, No. 7, pages 1279–1302 (2003)

of neural contour mechanisms: from simple to end-stopped cells. *Vision Research*, 32(5):963–981, 1992.
18. R. Horaud and T. Skordas. Stereo correspondence through feature grouping and maximal cliques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(11):1168–1180, 1989.
19. D. H. Hubel. *Eye, Brain, and Vision*. Scientific American Library, New York, 1988.
20. R.A. Jarvis. *Data Segmentation and Model Selection for Computer Vision –A Statistical Approach*, chapter 1. 2D and 3D Scene Segmentation for Robotic Vision. Springer-Verlag, 2000.
21. M. Krcmar and A. Dhawan. Application of genetic algorithms in graph matching. In *Proceedings of the International Conference on Neural Networks*, volume 6, pages 3872–3876, 1994.
22. P. Kuner and B. Ueberreiter. Pattern recognition by graph matching combinatorial versus continuous optimization. *International Journal on Pattern Recognition and Artificial Intelligence*, 2:527–542, 1988.
23. M. Lades, J. C. Vorbrüggen, J. Buhmann, J. Lange, C. von der Malsburg, R. P. Würtz, and W. Konen. Distortion invariant object recognition in the dynamic link architecture. *IEEE Transactions on Computers*, 42(3):300–311, March 1993.
24. T. Lourens. *A Biologically Plausible Model for Corner-based Object Recognition from Color Images*. Shaker Publishing B.V., Maastricht, The Netherlands, March 1998.
25. T. Lourens, K. Nakadai, H. G. Okuno, and H. Kitano. Graph extraction from color images. In *Proceedings of the 9th European Symposium on Artificial Neural Networks, ESANN 2001*, pages 329–334, Brugge, Belgium, April 2001.
26. T. Lourens, H. G. Okuno, and H. Kitano. Automatic graph extraction from color images. In E. Ardizzone and V. Di Gisù, editors, *Proceedings of the 11th International Conference on Image Analysis and Processing, ICIAP 2001*, pages 302–308, Palermo, Italy, September 2001.
27. S. Marčelja. Mathematical description of the responses of simple cortical cells. *J. Opt. Soc. Amer.*, 70:1297–1300, 1980.
28. J. A. McHugh. *Algorithmic Graph Theory*. Prentence-Hall International, Inc., 1990.
29. Bruno T. Messmer and Horst Bunke. A new algorithm for error-tolerant subgraph isomorphism detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):493–504, 1998.
30. Farzin Mokhtarian and Riku Suomela. Robust image corner detection through curvature scale space. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 20(12):1376–1381, December 1998.
31. S. H. Myaeng and A. Lopez-Lopez. Conceptual graph matching: a flexible algorithm and experiments. *Journal of Experimental and Theoretical Artificial Intelligence*, 4:107–126, April 1992.
32. N. J. Nilsson. *Principles of Artificial Intelligence*. Tioga, Palo Alto, 1980.
33. S. Peleg. A new probabilistic relaxation scheme. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2:362–369, July 1980.
34. Marcello Pelillo, Kaleem Siddiqi, and Steven W. Zucker. Matching hierarchical structures using association graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(11):1105–1120, 1999.
35. A. Rangarajan and E. Mjolsness. A lagrangian relaxation network for graph matching. In *Proceedings of the International Conference on Neural Networks*, volume 7, pages 4629–4634, 1994.
36. A. Rosenfeld, R. Hummel, and S. Zucker. Scene labeling by relaxation operations. *IEEE Transactions on Systems, Man, and Cybernetics*, 6:420–433, June 1976.

PREPRINT, published in
International Journal of Pattern Recognition and Artificial Intelligence,
Vol. 17, No. 7, pages 1279–1302 (2003)

37. P. L. Rosin. Augmenting corner descriptors. *Graphical Models and Image Processing*, 58(3):286–294, May 1996.
38. A. Sanfeliu and K. S. Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:353–363, 1983.
39. L. G. Shapiro and R. M. Haralick. Structural descriptions and inexact matching. *IEEE Transactions on pattern analysis and machine intelligence*, 3:504–519, 1981.
40. S. M. Smith and J. M. Brady. SUSAN - a new approach to low level image processing. *Int. Journal of Computer Vision*, 23(1):45–78, May 1997.
41. P. Suganthan, E Teoh, and D. Mital. Pattern recognition by graph matching using the potts mft neural networks. *Pattern Recognition*, 28:997–1009, 1995.
42. J. Ton and A. Jain. Registering landsat images by point matching. *IEEE Transactions on Geoscience and Remote Sensing*, 27:642–651, September 1989.
43. J. Triesch and C. Eckes. Object recognition with multiple feature types. In L. Niklasson, M. Bodén, and T. Ziemke, editors, *Proceedings of the ICANN'98, International Conference on Artificial Neural Networks*, pages 233–238. Springer-Verlag, 1998.
44. W. H. Tsai and K. S. Fu. Error-correcting isomorphisms of attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 9:757–768, 1979.
45. J. R. Ullman. An algorithm for subgraph isomorphism. *Journal of the Association for Computing Machinery*, 23(1):31–42, 1976.
46. L. Weitzel, K. Kopecz, C. Spengler, R. Eckhorn, and H. J. Reitboeck. Contour segmentation with recurrent neural networks of pulse-coding neurons. In G. Sommer and J. J. Koenderink, editors, *Proceedings of the 7$^{\text{th}}$ International Conference on Computer Analysis of Images and Patterns*, Kiel, Germany, September 1997.
47. Laurenz Wiskott, Jean-Marc Fellous, Norbert Krüger, and Christoph von der Malsburg. Face recognition by elastic bunch graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):775–779, 1997.
48. Laurenz Wiskott and Christoph von der Malsburg. Face recognition by dynamic link matching. In Joseph Sirosh, Risto Miikkulainen, and Yoonsuck Choe, editors, *Lateral Interactions in the Cortex: Structure and Function*. The UTCS Neural Networks Research Group, Austin, TX, Electronic book, ISBN 0-9647060-0-8, http://www.cs.utexas.edu/users/nn/web-pubs/htmlbook96, 1996.
49. E. K. Wong. Model matching in robot vision by subgraph isomorphism. In H. Bunke and A. Sanfeliu, editors, *Syntactic and Structural Pattern Recognition -Theory and Applications*, pages 381–414. World Scientific, 1990.
50. Ingo J. Wundrich, Christoph von der Malsburg, and Rolf P. Würtz. Image representation by the magnitude of the discrete Gabor wavelet transform. *IEEE Transactions on Image Processing*, 2002. In revision.
51. R. P. Würtz and T. Lourens. Corner detection in color images by multiscale combination of end-stopped cortical cells. In W. Gerstner, A. Germond, M. Hasler, and J. D. Nicoud, editors, *Proceedings of the International Conference on Artificial Neural Networks, ICANN'97*, volume 1327 of *Lecture Notes in Computer Science*, pages 901–906. Springer Verlag, October 1997.
52. R. P. Würtz and T. Lourens. Corner detection in color images through a multiscale combination of end-stopped cortical cells. *Image and Vision Computing*, 18(6-7):531–541, April 2000.
53. S. S. Yu and W. H. Tsai. Relaxation by the hopfield neural network. *Pattern Recognition*, 25:197–208, February 1992.

PREPRINT, published in
International Journal of Pattern Recognition and Artificial Intelligence,
Vol. 17, No. 7, pages 1279–1302 (2003)

Tino Lourens received the MS degree in computer science in 1993 and the PhD degree in 1998, both from the University of Groningen, The Netherlands. From 1998 to 1999 he was affiliated with the Netherlands organization for applied scientific research at the human factors research institute (TNO-HFRI) as a visual researcher. From 1999 to 2001 he was a researcher at the Japan Science and Technology Corporation (JST), ERATO, Kitano Symbiotic Systems Project in Tokyo, Japan. In 2001 he was invited to join Starlab, Brussels, Belgium as a senior researcher. Until 2002 he was with GMD-JRL, Kitakyushu, Japan. Currently he is a researcher at the Honda Research Institute (HRI) in Wako, Japan. His research interests include object recognition from real world images, biological models of vision, visual programming, brain like computing, and multimodal integration.

Rolf P. Würtz obtained his diploma in mathematics from the University of Heidelberg, Germany in 1986. After that, he was research assistant at the Max-Planck-Institute for Brain Research in Frankfurt, Germany. In 1990, he joined the Institute for Neurocomputing at the University of Bochum, Germany, where he received his Ph.D. from the Physics department in 1994. Until 1997, he was a postdoctoral researcher at the department of Computing Science at the University of Groningen, The Netherlands. He is currently a scientific staff member at the Institute for Neurocomputing in Bochum. Research interests include neuronal models and efficient algorithms for object recognition, hand-eye coordination, integration of visual and tactile information, and links to higher cognition. General organization problems and solution methods inspired by living systems are studied under the framework of "Organic Computing".