From: Roland Chin and Ting-Chuen Pong (eds.), Computer Vision — ACCV'98. Lecture Notes in Computer Science vol. 1352, Springer Verlag, 1998, pp. II-193 – II-200.

# Object Recognition by Matching Symbolic Edge Graphs

Tino Lourens[1] and Rolf P. Würtz[2]

[1] Computing Science, University of Groningen, The Netherlands,
http://www.cs.rug.nl/~tino/tino.html
[2] Institute for Neurocomputing, Ruhr-University Bochum, Germany,
http://www.neuroinformatik.ruhr-uni-bochum.de/ini/PEOPLE/rolf/

**Abstract.** We present an object recognition system based on symbolic graphs with object corners as vertices and outlines as edges. Corners are determined in a robust way by a multiscale combination of an operator modeling cortical end-stopped cells. Graphs are constructed by line-following between corners. Model matching is then done by finding subgraph isomorphisms in the image graph. The complexity is reduced by adding labels to corners and edges. The choice of labels makes the recognition system invariant under translation, rotation, and scaling.

## 1 Introduction

Labeled graph matching is a method used successfully for, e.g., invariant face recognition [1]. In that system, vertices are assigned local texture elements and edges carry the geometrical information about the relative locations. Applicability of that 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 outlines. In this paper we present a somehow complementary matching scheme that matches corners and connecting edges with the idea that a combination of both approaches will yield a fairly general recognition method.

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 can currently not handle significant occlusion.

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, which is NP-complete in the number of vertices. In our approach, the complexity is drastically reduced by demanding that the labels of matched vertices and edges, respectively, must be roughly equal. We will show experimentally that the problem is so reduced to a tractable size.

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 the graph extraction method [2]. A neuronal model for graph matching was proposed in [1]. 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. In [3], e.g., it is shown that partial contour deletion only impedes object recognition if it is accompanied by altering corner attributes.

## 2 Symbolic edge graphs

We will describe objects and scenes as graphs with corners as vertices and outlines as edges. The most important prerequisite for such a method is a robust corner detector.

### 2.1 Robust corner detection

Our method for detecting corners yields position, sharpness, size and color and grey-scale distribution (contrast). The subtended angle can be determined a posteriori by following the line segments that constitute the corner (see section 2.2). It is based on a model of cortical end-stopped cells [4]. These model cells are a nonlinear combination of the amplitudes of well-known Gabor functions [1] applied to an image. We denote those Gabor responses by $\mathcal{C}_{\sigma\theta}$ at scale $\sigma$ and orientation $\theta$.

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. The whole algorithm is described in [5]. Here, it suffices to mention that that algorithm yields the corner positions with sufficient accuracy and reliability. With a slight and biologically justified extension of the concept of complex cells, line and corner detection can be extended to color channels, which is also described in detail in [5].

### 2.2 Line following

A simple method to get the outlines of objects are binarized responses of appropriate edge detectors. Those, however, have turned out to be very sensitive to changes in local contrast and size of thresholds. We are using a method that starts from corners and collects evidence for a line to connect this corner with another one. Thus, the resulting edge graphs can be called “symbolic”. The edge detector consists of the Gabor moduli $\mathcal{C}_{\sigma\theta}$. The complete graph extraction algorithm is shown in figure 1.

**Orientation selection** A corner (or line-end) contains implicit information that one or more lines start from this position. Therefore we start searching for

```
Procedure ExtractGraph (C, I, G)
    C' := Corner cluster elimination C
    forall c ∈ C'
        O := all orientations at distance d from c where a segment is found
        forall o ∈ O
            repeat
                follow segment with intial orientation o
            until stop criterion fulfilled
            if stop criterion is another corner
                store detected segment
    optimize detected segments
    G := represent detected segments as a 2-D planar graph
```

**Fig. 1.** Algorithm for graph extraction.

possible lines using a circle with the corner at the center and a certain radius $d$. On this circle a number of equidistant samples are taken and the response of the $\mathcal{C}_\sigma$-operator, which is the average of $\mathcal{C}_{\sigma\theta}$ for all $\theta$, is determined. In order to assure that the algorithm indeed follows a line, the operator is also averaged over all scales, yielding the operator $\mathcal{A}$. A line segment in orientation $\phi$ is selected if the following conditions are satisfied:

1. The response of the $\mathcal{A}$-operator is above a threshold $T$.
2. The response is 20% higher than the weakest response from the samples of the $\mathcal{A}$-operator on the circle.
3. It is a local maximum among the samples taken.
4. The angle $\theta_0$ where the response of $\mathcal{C}_{\sigma\theta}$ is maximal lies closer to $\phi$ than the sampling stepsize of the orientations. This angle is called the *principal orientation* $\mathcal{O}_\sigma$ at the current image location.

**Following a line** The subsequent steps of line following are simpler than the starting one, because the line must roughly continue in the same direction. If a line diverts too much from that heuristic the corner detector must find a corner there. Thus, the circles around the corner are replaced by arcs of twice the sampling stepsize around the current line orientation.

**Stop criteria** At every step the following five stop criteria are checked, and following the current line terminates if one of them is satisfied.

1. Another corner is found.
2. One of the samples falls outside the image.
3. There is no sample where the output of the $\mathcal{A}$-operator is above the threshold $T$ and the principal image orientation falls inside the possible range.
4. The orientation at coordinate $(x_i, y_i)$ in the line does not correspond with the preferred orientation $\mathcal{O}_\sigma$.

5. The response of the $\mathcal{A}$-operator differs too much from the response at the previous step.
6. The length of the line exceeds twice the image size.

**Line optimization** The line following has a certain step size which may not be chosen too large. Consequently, straight lines or lines of small curvature get represented by too many intermediate points.

During the process of line following a point $(x_i, y_i), ) \leq i \leq n$, is calculated at every step. $(x_0, y_0)$ is the position of the corner where the line starts from and $(x_n, y_n)$ is the end-point of the line, which is usually another corner. If a straight line can be drawn between the two corners, all coordinates $(x_1, y_1), \ldots, (x_{n-1}, y_{n-1})$ can be dropped. In general, if we have three coordinates $(x_{i-1}, y_{i-1}), (x_i, y_i)$, and $(x_{i+1}, y_{i+1})$ then $(x_i, y_i)$ can be eliminated if the angle at $(x_i, y_i)$ is smaller than $\delta'$. In the simulations we used $\delta' = 5°$. When the point with index $i$ is eliminated the same procedure is repeated for points with indices $i-1, i+1$, and $i+2$. If coordinate $(x_i, y_i)$ is not removed, the procedure is repeated for coordinates with indices $i, i+1$, and $i+2$.

We apply the line following algorithm separately at every scale. This minimizes the chance for missing a line, but now, a line may be detected more than once. When this is the case, or more precisely, if several lines connect the same pair of corners and the average distance between them becomes smaller than a threshold, all but the shortest one are eliminated.

### 2.3 Graph labels

After all these steps we end up with a graph that has corners as vertices and curve segments described by a series of points as its edges. 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.

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). This choice of labels automatically yields invariance under translation, rotation and changes in size.

## 3 Graph matching

A brute force approach to test if two graphs are isomorphic is to exhaustively test every one-to-one mapping between the vertices of the graphs, which implies that every possible permutation is tested. We are using a modified version of the algorithm for subgraph isomorphism is from Ullman [6] based on tree search with backtracking. To cut down evaluation expenses the above mentioned labels 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 model may appear several times in the image graph.

```
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. 2.** Algorithm for graph matching.

Consequently, we are interested in all "copies" of a model graph $G_m$ in the given image graph $G$.

We assume that the model graphs are either constructed by hand or extracted from "clean" images. Thus, they contain *all* edges, wheras in the image graph some may be missing. Our matching process allows for this by allowing every non existing edge between two different vertices to be added at a cost of 1, the total cost being limited by a parameter, which has been set to 2 in the experiments. Thus, our graph matching model finds both exact and inexact copies of the models in the image graph.

The algorithm for graph matching is illustrated in figure 2. 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 on 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 we have a complete match (line 14). If the match is complete, the maximum and average relative length differences to the model are calculated, if both are below a certain threshold the match is accepted and displayed. If we do not have a complete match we go to 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 we parse the missing model vertex by adding it to the list (line 26) and find all possible vertices in the image graph (lines 27–30). A vertex $v$ is added if it is not matched yet and if the cost $EC$ of adding edge $(L_{cv}, v)$ is smaller than the allowed cost.

The speed of the algorithm depends mainly on the condition of 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 $\delta r$ between two pairs of edges is obtained by scaling the edge pairs 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 chapter. The absolute difference of the model edge with its corresponding image edge is taken, when there was no edge between $v_b$ 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 3 shows (what used to be) a color image and the extracted image graph. We have used the model graph illustrated in figure 3c) to find all the markers in the image.

The result is illustrated in figure 3d). We allowed at most two out of the seven edges in the model graph to be added and a maximal $\delta r$ of 5. We tolerated an angle difference of at most 36° and also an average angle difference of at most 36°, since there are 10 angles in the model graph this means that the summed angular difference should not exceed 360 degrees. 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 are approximately 1 second.

## 5 Discussion

We have presented a graph matching scheme for object recognition based on corners and outlines 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. This makes these graphs true symbolic information.

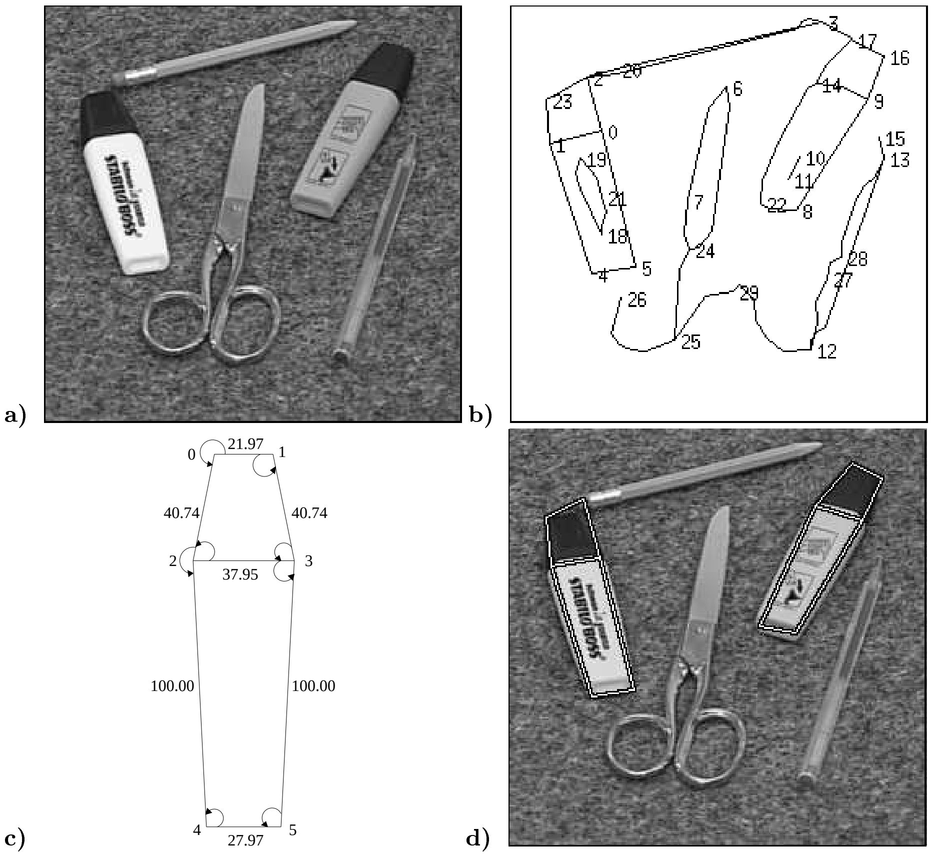


**Fig. 3. a)** Input image. **b)** Image graph extracted from **a)**, with numbered vertices. **c)** A model graph with implicit attributes, such a the (relative) length of an edge and the angle between two edges. **d)** Found matches of the markers. The used parameters are: maximum five edges to be added, ratio between two edge pairs $\delta r = 5$, average angle tolerance of 10%, maximum angle tolerance of 10%, maximum length tolerance of 50%, and an average length tolerance of 10%.

The NP-complete problem of subgraph matching has been greatly simplified by assigning angles between edges and relative sizes of edges as labels to the corners. 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 complexity of the algorithm is $O(N^4)$, 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.

The current version of the graph matching system is only the first working prototype. We are currently testing the robustness on many 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 lines starting and ending at corners. This causes the poor representation of the scissors in figure 3b).

We did not use curved lines for matching but help vertices which are used to give a proper description of the curve. It can be used to get a better description of the model and gives a more accurate cost of a match in the image graph. When additional coordinates are used we can apply a curve matching by surface difference algorithm.

Further developments will try and integrate texture and color features in order to combine this approach with the one from [1]. A further extension could use three-dimensional model graphs for matching into two-dimensional images and could yield a truly 3D object recognition system.

**Acknowledgments.** Financial support from the NEUROS-Project by the German Research Minister (01 IN 504 E 9) is gratefully acknowledged.

## References

1. Martin Lades, Jan C. Vorbrüggen, Joachim Buhmann, Jörg Lange, Christoph von der Malsburg, Rolf P. Würtz, and Wolfgang Konen. Distortion invariant object recognition in the dynamic link architecture. *IEEE Transactions on Computers*, 42(3):300–311, 1993.
2. Lothar Weitzel, Klaus Kopecz, Claus Spengler, Reinhard 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'th International Conference on Computer Analysis of Images and Patterns,Kiel, Germany, September 10-12, 1997*, 1997.
3. Irving Biedermann. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94(2):115–147, 1987.
4. Friedrich Heitger, Lukas Rosenthaler, Rüdiger von der Heydt, Esther Peterhans, and Olaf Kübler. Simulation of neural contour mechanisms: from simple to end-stopped cells. *Vision Research*, 32(5):963–981, 1992.
5. Rolf P. Würtz and Tino Lourens. Corner detection in color images by multiscale combination of end-stopped cortical cells. In Wulfram Gerstner, Alain Germond, Martin Hasler, and Jean-Daniel Nicoud, editors, *Artificial Neural Networks – ICANN '97*, volume 1327 of *Lecture Notes in Computer Science*, pages 901–906, Berlin, Heidelberg, New York, 1997. Springer Verlag.
6. J. R. Ullman. An algorithm for subgraph isomorphism. *Journal of the Association for Computing Machinery*, 23(1):31–42, 1976.