Graph Extraction from Color Images

T. Lourens, K. Nakadai, H.G. Okuno, and H. Kitano

Japan Science and Technology Corporation
ERATO, Kitano Symbiotic Systems Project
M. 31 Suite 6A, 6-31-15 Jingumae, Shibuya-ku, Tokyo 150-0001, Japan
{*tino, nakadai, okuno, kitano*}*@symbio.jst.go.jp*

Abstract. An approach to symbolic contour extraction will be described that consists of three stages: enhancement, detection, and extraction of edges and corners. Edges and corners are enhanced by models of monkey cortical complex and end-stopped cells. Detection of corners and local edge maxima is performed by selection of local maxima in both edge and corner enhanced images. These maxima form the anchor points of a greedy contour following algorithm that extracts the edges. This algorithm is based on an idea of spatially linking neurons along the edge that will fire in synchrony to indicate an extracted edge. The extracted edges and detected corners represent the symbolic representation of the image. The advantage of the proposed model over other models is that the same low constant thresholds for corner and local edge maxima detection are used for different images. Closed contours are guaranteed by the contour following algorithm to yield a fully symbolic representation which is more suitable for reasoning and recognition. In this respect our methodology is unique, and clearly different from the standard edge detection methods.

1 Introduction

A human recognizes objects with great ease, although a distribution of light intensities can change considerably, e.g. when an object is scaled or placed in a different environment. 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 (Faugeras [5]).

Unfortunately, the functionality and representation space of the brain are only known partly. The existence of the so called simple, complex, and end-stopped cells found by Hubel and Wiesel [8] in early vision proves that edges and corners play an important role in mammalian vision. Graphs,[1] where edge

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

contours and corners represent edge contours and corners, respectively, have been proven to be useful for object recognition [3, 4, 9].

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, in order to detect line segments. After that an extraction algorithm should be applied to obtain a symbolic representation. A well known method based on this technique is from Canny [2], but the problem in this approach is setting a different threshold for every image.

In this paper, a greedy edge contour following algorithm will be proposed that avoids manual threshold and length of contour settings for every different image. The idea of following and extraction is based on neurons that fire in synchrony and activate neighboring cells along the contour. Standard algorithms for edge extraction based on a similar principle are the border-tracking algorithm [6] and following as graph searching [1], but need further complicated processing to obtain a graph.

The paper is organized as follows: Section 2 describes the enhancement of edges and corners based on models of complex and endstopped cells found in early vision. Section 3 contains the (attributed) graph extraction algorithm based on the idea of synchronously firing neurons along edge contours. In Section 4 results of the extraction algorithm are compared with the Canny [2]-CSS [10] and SUSAN edge and corner detectors [11]. Comparison is made possible by displaying the extracted symbolic results of our proposed algorithm as a two-dimensional image. The last section gives summary and discussion.

2 Edge enhancement

Corner detection and edge following are based on a Gabor wavelet transform of the image. These Gabor functions have been modified so that their integral vanishes and their one-norm (the integral over the absolute value) becomes scale independent. They provide a transform of the image via spatial convolution. Afterwards, only the amplitudes of the complex values, that represent the complex cell responses at a single scale and orientation, are retained for further processing. Starting from this representation, we have developed a biologically motivated method for corner detection, for details and motivation, see [12]. Our method for detecting corners yields position, sharpness, size, color, and contrast. It is based on a model of cortical end-stopped cells [7].

With a slight and biologically justified extension of the concept of complex and endstopped cells can be extended to color (red-green and blue-yellow opponent) channels. The amplitude for each of the grey value, red-green, and blue-yellow channels yields the edge operator

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

at a single scale. The corner operator is similar to (1), with the exception that for $\mathcal{C}$ one should substitute $\mathcal{E}$.

Scale plays an important role in edge detection. In general it is hardly impossible to select a single optimal scale. We found that averaging the responses over a range of frequencies yields a much more robust corner detection operator:

$$\mathcal{E}^{\text{all}}_{\text{avg}}(x,y) = \frac{1}{S}\sum_{j=0}^{S-1}\mathcal{E}^{\text{all}}_{\sigma_j}(x,y). \tag{2}$$

3 Algorithm for attributed graph extraction

In standard edge detection, one global threshold is set for the whole image to determine whether a pixel belongs to an edge. In most cases the threshold needs to be adjusted for every image, manually. Thresholding can be avoided, if instead of performing the operations on the whole image, segments are detected by contour following. 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 we call *linking* cells, 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. This firing in synchrony is what we will call the *linking* of the (neighboring) neurons. Globally the algorithm (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

```
1   Function ExtractAttributedGraph (E^all_ω, C^all_σ)
2     C := Set of corners obtained by taking local maxima from E^all_ω
3     M := Set of local edge maxima obtained from C^all_σ
4     forall m ∈ M
5       forall n ∈ BestNeighborSelectedByResponseAndOppositeCoordinate (C^all_σ, m)
6         l := ExtractContour (C^all_σ, C, M, m, n)
7         Add (L, l)
8     forall c ∈ C
9       forall n ∈ EightNeighbors (c)
10        l := ExtractContour (C^all_σ, 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
```

Figure 1: Algorithm for attributed graph extraction. Variable ω is σ or avg.

3.1 Detection of corners and local maxima

Although setting a threshold in most algorithms depends strongly on the input image, we created a robust corner operator by averaging over multiple scales, hence setting one threshold at a reasonable value yields good results in almost all images. Selection of local edge maxima is a less critical process than marking corners, therefore a constant threshold at a single scale that is equal or even lower than that for marking corners gives satisfactory results. A position (x, y) is *marked* as a corner if the $\mathcal{E}^{\text{all}}_{\text{avg}}(x, y)$ response is larger than its neighbors, i.e. a local maximum, and above $T_{\mathcal{E}}$. Similarly (x, y) is a local edge maximum if $\mathcal{C}_\sigma(x, y)$ is larger than its neighbors and above $T_{\mathcal{C}}$.

3.2 Edge contour extraction

The second step in the algorithm is contour extraction. Local edge maxima and corners should be treated differently. At a corner several, but at most eight,[2] contours can start and in order to avoid missing any, a contour will be followed in every direction.A contour is always passing through a local maximum, otherwise a corner should have been marked there. Hence at a local maximum two opposite directions will be selected to be followed.

Contour following is based on a greedy algorithm: walking from one top, local edge maximum, to another and trying to keep as high as possible. The latter is done by selecting always that coordinate (within certain constraints) with the highest $\mathcal{C}^{\text{all}}_\sigma$-response. In the layer of linking cells this means that the neighbor with strongest input response is activated and linked by firing synchronously with the other activated neurons.

```
1  Function ExtractContour (C^all_σ, C, M, c, n)
2    p := n; contour := p; pp := c
3    repeat
4      if (NotAtBorderImage(p) and NoMember(contour, p) and C^all_σ(p) ≥ T and p ∉ C ∪ M)
5        p_1 := Neighbor of p to pp at 180°
6        p_2 := Neighbor of p to pp at 135°
7        p_3 := Neighbor of p to pp at 225°
8        pp := p;   p := p_j, where C^all_σ(p_j) = max(C^all_σ(p_i)) ∀i; i, j ∈ {1, 2, 3}
9        Add (contour, p)
10     else stop
11   until stop
12   if (p ∈ C ∪ M or Length (contour) ≥ Len)
13     return contour
```

Figure 2: Algorithm for extraction of a single contour from local edge maximum or corner to another local edge maximum or corner.

The algorithm is given in Figure 2. In this figure variable p denotes the current point or coordinate and pp previous point, which is initially a corner or local edge maximum. At every step there are three possible neighbors of p that can be selected. These neighbors form a 135, 180, and 225 degree angle

[2]The smallest unit in a digital image is a pixel, on a square grid of pixels there are (by definition) 8 neighboring pixels. Hence, at most 8 separate contours can connect a single pixel.

with p and pp (lines 5-7). From these three neighbors the one with the highest $\mathcal{C}_\sigma^{\text{all}}$-response is selected (line 8).

3.3 Connecting corners to contours

A third step is necessary because a contour does not necessarily have to pass through a corner, to get a fully symbolic representation, an additional step is taken that connects corners to contours if the closest distance between corner and contour is less than $d\sigma$. This value is derived from the single end-stopped operator.

4 Edge detection methods

In this section, we compare our method with the Canny edge detector combined with the curvature scale space (CSS) detector, and with the SUSAN edge and corner detector. Although edge detection and extraction are not the same, we can compare results by displaying the extracted data in a two dimensional image, by treating it as if it is detected. As mentioned in the introduction already, extraction of edge contours is very difficult for both Canny and SUSAN, since the contours sometimes contain gaps. For both Canny-CSS and SUSAN we set the thresholds manually for every image to get the optimal results for both methods. Results of the methods are in Figure 3.

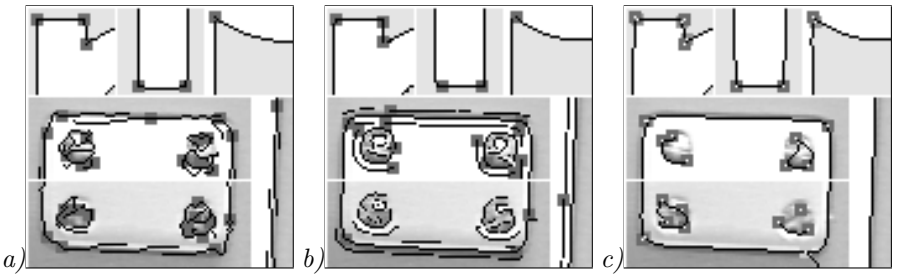

Figure 3: Comparison with other detection methods. *a)* SUSAN edge and corner detector. *b)* Canny edge detector combined with the CSS corner detector. *c)* Results obtained with our method for $\sigma = 2.36$ and default parameters ($T_{\mathcal{E}}$=10, $T_{\mathcal{C}}$=10, T=1, and Len=15). Corners are represented as grey squares and detected contours are marked in black. In our method: little brighter grey squares are corner edge connections and bright squares denote a junction (due to extraction) or the end of a contour.

5 Summary and discussion

In this paper we described an approach to multiscale corner and edge extraction, based upon complex and endstopped cells. Extraction of enhanced edges from an image is based upon an idea and possible existence of, which we termed, linking neurons. These neurons which receive their input from the complex and

end-stopped cells are activating each other along enhanced edges and start firing in synchrony, indicating that they formed a linked chain.

For corner and edge detection we used low constant thresholds for different images. In this respect our method is very robust and outperforms the other edge detection methods, where setting a threshold often strongly influences the results. The extraction algorithm guarantees contours without gaps and avoids global threshold settings. The presented method performs equally well or better compared to the SUSAN and Canny-CSS detection methods, but is especially a valuable choice when further processing for symbolic reasoning is used.

In the current approach there are shortcomings compared to what humans perceive as edge contours, this is because only a feedforward mechanism for solely form features is used. Results might be improved by using other visual cues, like color, texture, and motion. Also a feedback mechanism could improve and facilitate edge extraction. Another very important factor for edge extraction that is not used, is knowledge, e.g., often true edges are ignored because they are not relevant in the current scene. Future research needs to incorporate these cues to improve the results.

References

[1] D. H. Ballard and C. M. Brown. *Computer Vision*. Prentence-Hall Inc., 1982.

[2] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(9):679–698, November 1986.

[3] C. Dillon and T. Caelli. Learning image annotations: The CITE system. *Videre: Journal of Computer Vision Research*, 1(2):90–121, Winter 1998.

[4] 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.

[5] O. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. The MIT Press, Cambridge, Massachusetts; London, England, 1993.

[6] R. M. Haralick and L. G. Shapiro. *Computer and Robot Vision*. Addison-Wesley, 1992.

[7] F. Heitger, L. Rosenthaler, R. von der Heydt, E. Peterhans, and O. Kübler. Simulation of neural contour mechanisms: from simple to end-stopped cells. *Vision Research*, 32(5):963–981, 1992.

[8] D. H. Hubel. *Eye, Brain, and Vision*. Scientific American Library, New York, 1988.

[9] 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.

[10] 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.

[11] 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.

[12] 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.