Automatic Graph Extraction from Color Images

T. Lourens, 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, 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 contours and corners. Contours and corners are enhanced by models of monkey cortical complex and endstopped cells. Detection of corners and local contour maxima is performed by selection of local maxima in both contour and corner enhanced images. These maxima form the anchor points of a greedy contour following algorithm that extracts the contours. This algorithm is based on an idea of spatially linking neurons along a contour that will fire in synchrony to indicate an extracted contour. The extracted contours 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 contour 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) contour detection methods. The results of the extracted contours (when displayed as being detected) show similar or better results compared to the SUSAN and Canny-CSS detectors.

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 [6].

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 (see, e.g., the book of Hubel [9]) in early vision proves that contours and corners play an important role in mammalian vision. Graphs,[1] where edges and vertices represent contours and corners, respectively, have been proven to be useful for object recognition [3, 5, 14]. By adding attributes (which are implicitly available when visual objects are extracted), the graph can be made invariant under changes of scale, rotation, and translation. A fast graph matching algorithm using these attributes even recognized objects that are deformed and incomplete [13] and was applied successfully in a humanoid that "simulated" selective attention by integrating audio and video cues [12].

A problem that has been tackled party is the extraction of contours 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 contour detection in image processing are based upon enhancement of contours 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]. The problem in this approach is setting a threshold, which is called a *plague* by Faugeras [6]. To his knowledge thresholding is unavoidable, and should be tackled with courage. Very recently, Jarvis [10] stated, that even when sophisticated contour 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 contour following algorithm will

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

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 contour extraction based on a similar principle are the border-tracking algorithm [7] and following as graph searching [1], but need further complicated processing to obtain a graph.

The paper is organized as follows: Section 2 gives a mathematical framework for contour and corner enhancement based on the 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 contours. In Section 4, results of the extraction algorithm are compared with the Canny-CSS [15] and SUSAN contour and corner detectors [16]. Comparison is made possible by displaying the extracted symbolic results of our proposed algorithm as a 2D image. Section 5 gives a summary and discussion.

2 Contour enhancement

Corner detection and contour enhancement are based on a Gabor wavelet transform of the image. Complex-valued Gabor functions at scale σ and orientation θ are defined as

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

These Gabor functions have been modified so 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 a transform of the image $I(x,y)$ via spatial convolution. Afterwards, only the amplitudes of the complex values are retained for further processing:

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

This representation, which models the responses of complex cells, is the basis of all subsequent processing. 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: $\theta_i = \frac{i\cdot 2\pi}{N}, i = 0\ldots N-1$, $\sigma_j = \sigma_0 + j\Delta\sigma, j = 0\ldots S-1$.

2.1 Robust corner detection

Starting from the local energy representation, a biologically motivated method for corner detection was developed, which is described here only briefly. For details and motivation of the constants, see [17]. The method for detecting corners yields position, sharpness, size, color, and contrast. It is based on a model of cortical endstopped cells [8].

The first step towards an endstopped operator is an approximation of the *first* (given by the $\mathcal{E}^s$-operators which represent the single endstopped cell responses) and *second* (given by the $\mathcal{E}^d$-operators which represent the double endstopped cell responses) derivative of the $\mathcal{C}$-operator in the direction orthogonal to that of the line segment in question:

$$\hat{\mathcal{E}}^s_{\sigma,\theta}(x,y) = \mathcal{C}_{\sigma,\theta}(x+s,y-c) - \mathcal{C}_{\sigma,\theta}(x-s,y+c) \quad (3)$$

$$\hat{\mathcal{E}}^d_{\sigma,\theta}(x,y) = \mathcal{C}_{\sigma,\theta}(x,y) - 0.5\mathcal{C}_{\sigma,\theta}(x+2s,y-2c) - 0.5\mathcal{C}_{\sigma,\theta}(x-2s,y+2c), \quad (4)$$

where $s = d\sigma\sin\theta$ and $c = d\sigma\cos\theta$. These two operators are both inhibited by a tangential ($\mathcal{I}^t$) and a radial ($\mathcal{I}^r$) inhibiting operator:

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

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

where % denotes the modulus function, $x_1 = x + d\sigma\cos\theta_i$, and $y_1 = y + d\sigma\sin\theta_i$. Half-wave rectification is obtained by $[z]^{\geq 0}$ which is equal to 0 for negative z and equal to z elsewhere. Constant w_r is set to 4 to suppress all false responses. The corner operators on a single scale in a single direction then are:

$$\mathcal{E}_{\sigma,\theta_i} = \left[\left[\hat{\mathcal{E}}_{\sigma,\theta_i}\right]^{\geq 0} - g(\mathcal{I}^t_\sigma + \mathcal{I}^r_\sigma)\right]^{\geq 0}. \quad (5)$$

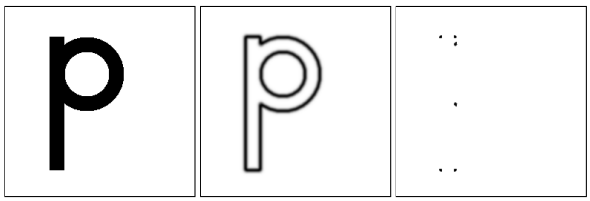
Constant $g = 2$ is a gain factor and for $\mathcal{E}$ one can substitute $\mathcal{E}^s$ or $\mathcal{E}^d$. Orientations and single and double endstopped operators are combined by a maximum operator, while orientations for contours are combined by amplitude:

$$\mathcal{E}_\sigma = \max_{i=0}^{2N-1}(\max(\mathcal{E}^s_{\sigma,\theta_i}, \mathcal{E}^d_{\sigma,\theta_i})) \quad \mathcal{C}_\sigma = \sqrt{\sum_{i=0}^{N-1}(\mathcal{C}_{\sigma,\theta_i})^2}. \quad (6)$$

Combining contours by amplitude yields better results than combining with a maximum or averaging operator. Results of the $\mathcal{C}_\sigma$- and $\mathcal{E}_\sigma$-operators are illustrated in Figure 1.

2.2 Combining color channels

With a slight and biologically justified extension of the concept of complex and endstopped cells can be extended to color channels (red-green and blue-yellow opponent), which is described in detail in [11].



a) Image *b)* $\mathcal{C}_\sigma$-operator *c)* $\mathcal{E}_\sigma$-operator

Figure 1. *a)* **Synthetic input image.** *b)-c)* **Results of the $\mathcal{C}_\sigma$- and $\mathcal{E}_\sigma$-operators applied at a single scale (S = 1) to the input image. The parameters used are σ = 2.36 and N = 8.**

Combining the achromatic and two color opponent (red-green and blue-yellow) channels by maximum, sum, or amplitude all result in a strongest response at the contour. In this respect any arbitrary (maximum, sum, or amplitude) combining operator can be chosen. The amplitude is chosen because the strength of the response at the contour corresponds to the Euclidean color distance between two areas with color (r_a, g_a, b_a) and (r_b, g_b, b_b) which formed the contour. The amplitude for each of the grey value ($\mathcal{C}_\sigma$), red-green ($\mathcal{C}_\sigma^{r,g}$), and blue-yellow ($\mathcal{C}_\sigma^{b,y}$) channels yields the final contour operator at a single scale:

$$\mathcal{C}_\sigma^{\text{all}} = \sqrt{(\mathcal{C}_\sigma)^2 + \left(\frac{1}{2}\mathcal{C}_\sigma^{r,g}\right)^2 + \left(\frac{1}{2}\mathcal{C}_\sigma^{b,y}\right)^2} \qquad (7)$$

Since $\mathcal{C}_\sigma^{e,i} = \mathcal{C}_\sigma^{i,e}$, one channel for every opponent pair is sufficient.The final corner operator is obtained in a similar way, its equation is identical to (7) if $\mathcal{E}_\sigma$ is substituted for $\mathcal{C}_\sigma$.

2.3 Multiple scales

Scale plays an important role in contour detection. In general it is hardly impossible to select a single optimal scale, since there exists no scale of threshold which can be defined a priori to detect contours [4]. 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 operator:

$$\mathcal{E}_{\text{avg}}^{\text{all}}(x,y) = \frac{1}{S}\sum_{j=0}^{S-1} \mathcal{E}_{\sigma_j}^{\text{all}}(x,y) \ . \qquad (8)$$

For contour detection and extraction multiple scales are used as well, but contours are extracted for every scale separately.

3 Algorithm for attributed graph extraction

In standard contour detection, one global threshold is set for the whole image to determine whether a pixel belongs to a contour. 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 contour maxima as starting points, and then to follow a contour to another corner or local maximum by selecting the strongest contour 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 contours will be the final result. This firing in synchrony is what we call the *linking* of the (neighboring) neurons.

```
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 contour maxima obtained from C_σ^all
4    forall m ∈ M
5      forall n ∈ BestNeighborSelectedByResponseAnd-
       OppositeCoordinate (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
```

Figure 2. Algorithm for attributed graph extraction. Variable ω is σ or avg.

Globally the algorithm (Figure 2) contains 3 stages:

1. detecting corners and local contour 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 a contour

3. connecting corners with edge contours and each other (lines 13-16); also activating and linking neighboring neurons along a contour

3.1 Detection of corners and local contour 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 contour maxima is less critical than marking corners, therefore a constant threshold $T_{\mathcal{C}}$ at a single scale that is equal or even lower than threshold $T_{\mathcal{E}}$, which is used 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 contour maximum if $\mathcal{C}_\sigma(x, y)$ is larger than its neighbors and above $T_{\mathcal{C}}$.

3.2 Contour extraction

The second step in the algorithm is contour extraction. Local contour 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.[3] 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 contour 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.

The algorithm is given in Figure 3. 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 with p and pp (lines 4-6). From these three neighbors the one with the highest $\mathcal{C}^{\text{all}}_\sigma$-response is selected (line 7).

[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.

[3]Following in every direction is the simplest way, but it is not strictly necessary, since directions can be derived from the $\mathcal{E}_{\sigma,\theta}$-operator.

```
1  Function ExtractContour (C^all_σ, C, M, c, n)
2    p := n; contour := p; pp := c
3    while (NotAtBorderImage (p) and NoMember (contour, p)
           and C^all_σ(p) ≥ T and p ∉ C ∪ M)
4      p1 := Neighbor of p to pp at 180°
5      p2 := Neighbor of p to pp at 135°
6      p3 := Neighbor of p to pp at 225°
7      pp := p;  p := p_j, where C^all_σ(p_j) = max(C^all_σ(p_i))
                                          ∀i;   i, j ∈ {1, 2, 3}
8      Add (contour, p)
9    if (p ∈ C ∪ M or Length (contour) ≥ Len)
10     return contour
```

Figure 3. Algorithm for extraction of a single contour from a contour maximum or corner to another contour maximum or corner.

3.3 Connecting corners to contours

A third step is necessary because a contour does not necessarily have to pass through a corner. In order to get a fully symbolic representation, which is needed for symbolic reasoning, an additional step is taken that connects corners to contours if the closest distance between corner and contour is less than $d\sigma$. The value $d\sigma$, is the distance where the complex cell operator influences the endstopped operator.

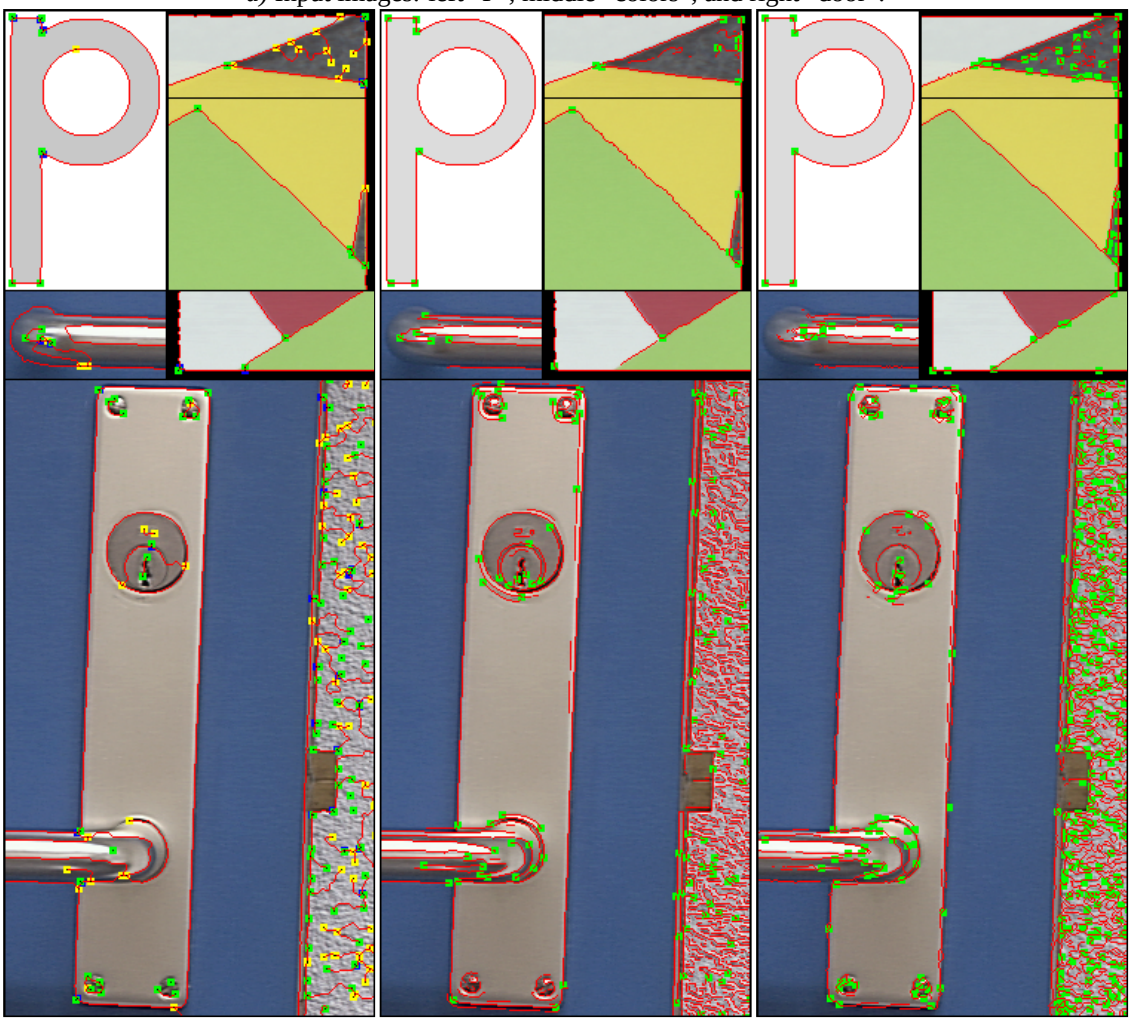
4 Evaluation of contour detection methods

In this section we evaluate our method by comparing it with the Canny edge detector combined with the curvature scale space (CSS) detector and with the SUSAN edge and corner detector. Although contour 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 already mentioned in the introduction, extraction of 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.

Figure 4b-d (3 top-left squares) illustrate that all three methods are able to detect corners and contours accurately in the synthetic P-image. Our method found correctly the 7 vertices (6 corners and 1 local contour maximum at the circle) and 7 contours, see Figure 5. For both synthetic and real world images we used the default parameters, a single scale with one small σ, and did not do any additional postprocessing, to give an objective comparison between the methods.

Evaluation of results is subjective, since there are no simple criteria that can evaluate which method is better. The main focus of the real world images will be on properly detected corners and contours. For simplicity we will assume that additional processing can eliminate falsely detected corners and contours.

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



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

Figure 4. *a)* **Top row input images used as benchmarks and bottom row enlarged results of parts of the input images.** *b)* **Results obtained with our method with σ = 2.36 and default parameters ($T_{\mathcal{E}}$=10, $T_{\mathcal{C}}$=10, T=1, and Len=15). Corners are represented as green squares and detected contours are marked in red. Blue squares are corner contour connections. Yellow squares denote a junction (due to extraction) or the end of a contour.** *c)* **Results of the Canny detector combined with the CSS corner detector.** *d)* **Results of the SUSAN edge and corner detector.**

4.1 SUSAN

For the SUSAN detector the brightness parameter is very sensitive and needs to be tuned for every image, while the distance parameter seems to have hardly any influence for values larger than 4. This corner detector underperformed in the real world images of Figure 4a. For example at the door image in Figure 4d, detected contours are (slightly S-shaped) shifted segments that are not likely to be closed using simple post-processing mechanisms. In the same image, corners are detected at absolutely straight contours of the rectangular metal part. Also, slightly rounded corners are not detected, instead corners are marked at the beginning and end of the arc of the corner.

vertices		start	end	attributes	
V_i	V_j	(x_i, y_i)	(x_j, y_j)	len	score
10	8	(79, 219)	(81, 132)	87	293.41
9	1	(60, 219)	(61, 40)	179	292.48
6	*6*	(103, 61)	(103, 61)	168	291.84
9	**10**	(60, 219)	(79, 219)	19	297.08
0	1	(60, 39)	(61, 40)	1	***
2	3	(80, 40)	(79, 41)	1	***
5	4	(80, 50)	(81, 48)	2	294.67
7	8	(80, 130)	(81, 132)	2	294.67
8	4	(81, 132)	(81, 48)	198	292.88
1	3	(61, 40)	(79, 41)	18	296.26
4	3	(81, 48)	(79, 41)	7	309.04

Figure 5. Symbolic representation of the extracted graph from the P input image. Corners, junctions, and contour ends are denoted by V_k, where $k \in [0-10]$ in this example. Every V_k has an (x_k, y_k) coordinate to indicate its position in a 2D image. The length len of contour V_i-V_j is the distance in pixels between V_i and V_j. The score gives an indication of the "quality" of the contour, * denotes that the contour contains coordinates (x_i, y_i) and (x_j, y_j), only. Corners are represented in boldface, the "contour end" of the circle is emphasized, and the others represent the connections from a corner to an extracted chain. Note that the extracted chains, which are coded as Freeman indices, are omitted in this figure, but they are displayed (in red) in Figure 4b. In that figure corners, contour ends, and others are represented in green, yellow, and blue, respectively. The used parameters to obtain this graph are are $T_{\mathcal{E}} = 10$, $T_{\mathcal{C}} = 10$, $T = 1$, $\sigma = 2.36$, and Len=15.**

4.2 Canny-curvature scale space

The CSS corner detector has a sensitive minimum length parameter and needs as input detected contours, hence results depend strongly on detected contours. The Canny edge detector has been proven to be a useful detector, certainly when all detailed edges in an image need to be detected. The detector gives good results in the image with the different colored rectangles (Figure 4c). Known problems in the detector are doubly detected lines and gaps between segments (even at corners in synthetic images), but it outperforms the SUSAN edge detector. The multiscale approach of the CSS corner detector shows good performance for the colored rectangles, where corners are detected properly. These results are obtained by setting the high threshold adjustment T_{high} to 20. In the door image of Figure 4c, the CSS corner detector shows satisfactory results with exception of the contours between the blue door and white wall, where corners are not detected.

5 Summary and discussion

In this paper we described an approach to corner and contour extraction and presented an algorithm that consists of three stages: enhancement, detection, and extraction.

Enhancement of contours is obtained by convolving the input image with Gabor filters at multiple orientations which represents a model for the well known simple and complex cells, that are found in monkey primary visual cortex. The model for corner enhancement is based on the responses of endstopped cells. *Detection* of corners is accomplished by taking the local maxima in the corner enhanced image. Likewise the local maxima in the contour enhanced image are marked. *Extraction* of enhanced contours 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 endstopped cells are activating each other along enhanced edges and start firing in synchrony, indicating that they formed a linked chain.

The advantages of the proposed method are:

- *A fully symbolic representation* is obtained that is more suitable for artificial intelligence than a discrete 2D image with detected corners and contours.
- *Results are similar or better* compared to the SUSAN and Canny-CSS detection methods.
- *Thresholds are the same for different images*. Due to the following mechanism global threshold settings are avoided. Thresholds are set to avoid contour extraction in areas where responses are caused by noise. In the other two methods threshold settings strongly influence the results. To obtain satisfactory results with these two methods, threshold tuning is needed, which is a tedious and time consuming job.

- *Every contour is closed* due to the following mechanism of the extraction algorithm. Hence, no further (complicated) postprocessing is necessary to obtain symbolic contours.
- *The extraction algorithm is fast*, since at most all pixels in the image, but in most images only a fraction of all pixels will be visited.

Although, the extraction algorithm is fast, the preprocessing for contour and corner enhancement is computationally expensive, due to its many convolutions. For contour and corner *detection* in synthetic images, where parameter tuning is not necessary, the SUSAN detector is the best choice, due to its speed. In case high sensitivity and accuracy is required in slightly more complicated images the Canny-CSS method is the best choice, since it is faster than our method. In cases when it comes to natural images, reasoning, or recognition, the proposed method is the best choice because of the resulting symbolic representation and its robustness to parameter settings.

5.1 Future work

Combining multiple scales works well for corner detection, but combining scales for enhanced contours by a max-, avg-, or ampl-operator is not the best solution. Probably the best method is to extract relevant contours from different scales, or first extract contours from all scales separately, and use further processing to decide which are true or falsely detected contours.

In the current approach there are shortcomings compared to what humans perceive as 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 contour extraction. Another very important factor for contour extraction that is not used, is knowledge, e.g., often true contours are ignored because they are not relevant in the current scene. Future research needs to incorporate these cues to improve the results.

5.2 Concluding remarks

Even while there are many ways to improve the results, the symbolic extraction algorithm is sufficient to let a humanoid robot perform vision tasks, successfully [12]. A successful example is selective attention in a (restricted) natural environment by symbolic object recognition, which is a step towards true intelligent artificial vision systems.

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] J. H. Elder and S. W. Zucker. Local scale control for edge detection and blur estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(7):699–716, July 1998.

[5] M. A. Eshera and K.-S. 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.

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

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

[8] 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.

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

[10] R. 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.

[11] T. Lourens. *A Biologically Plausible Model for Corner-based Object Recognition from Color Images*. Shaker Publishing B.V., Maastricht, The Netherlands, March 1998.

[12] T. Lourens, K. Nakadai, H. G. Okuno, and H. Kitano. Selective attention by integration of vision and audition. In *Proceedings of the First IEEE-RAS International Conference on Humanoid Robots, Humanoids2000*, pages 20, file: 44.pdf, The Massachusetts Institute of Technology, Boston, U.S.A., September 2000.

[13] T. Lourens and R. P. Würtz. Object recognition by matching symbolic edge graphs. In R. Chin and T. C. Pong, editors, *Proceedings of the Third Asian Conference on Computer Vision, ACCV '98*, volume 1352 of *Lecture Notes in Computer Science*, pages 193–200. Springer-Verlag, January 1998.

[14] B. T. Messmer and H. Bunke. A new algorithm for error-tolerant subgraph isomorphism detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):493–504, 1998.

[15] F. Mokhtarian and R. Suomela. Robust image corner detection through curvature scale space. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 20(12):1376–1381, December 1998.

[16] 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.

[17] 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.