ELSEVIER

Pattern Recognition Letters 23 (2002) 1537–1548

Pattern Recognition Letters

www.elsevier.com/locate/patrec

# Direct feature extraction in a foveated environment

Efri Nattel, Yehezkel Yeshurun *

*The Raymond and Beverly Sackler Faculty of Exact Sciences, School of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel*

Received 17 August 2001; received in revised form 26 January 2002

**Abstract**

Foveated sampling and representation of images is a powerful tool for various vision applications. However, there are many inherent difficulties in implementing it. We present a simple and efficient mechanism to manipulate image analysis operators directly on the foveated image; a single typed table-based structure is used to represent various known operators. Using the complex log as our foveation method, we show how several operators such as edge detection and Hough transform could be efficiently computed almost at frame rate, and discuss the complexity of our approach. © 2002 Elsevier Science B.V. All rights reserved.

*Keywords:* Foveated vision; Log polar mapping; Active vision; Interest operators; Symmetry operators; Hough transform; Spatial masks; Edge detection

## 1. Introduction

Foveated vision, which was originally biologically motivated, can be efficiently used for various image processing and image understanding tasks due to its inherent compressive and invariance properties (Wallace et al., 1994; Tistarelli and Sandini, 1993; Yamamoto et al., 1996). It is not trivial, however, to efficiently implement it, since we conceptualize and design algorithms for use in a Cartesian environment. In this work, we propose a method that enables implementation of image operators on foveated images that is related to Wallace et al. (1994), and show how it is efficiently used for direct implementation of feature detection on foveated images. We achieve this efficiency by taking the "lookup table" concept to its extreme, and implementing, on top of the connectivity graph proposed in Wallace et al. (1994), a full direct access space variant operator lookup table. This approach requires a rather resource intensive preprocessing stage for each operator (due to the space variance), but this stage is carried only once for each operator, and results in a real time implementation of foveated operators on images. Following the classification of Jain et al. (1995), we show how local, global, and relational (edge detection, Hough transform and symmetry detection, respectively) are implemented using our method.

Both our source images and the feature maps are foveated, based on Wilson's model (Yamamoto et al., 1996; Wilson, 1983). To achieve reasonable dimensions and compression rates, the

* Corresponding author. Tel.: +972-3-640-9368; fax: +972-3-640-9357.

*E-mail addresses:* nattel@math.tau.ac.il (E. Nattel), hezy@math.tau.ac.il (Y. Yeshurun).

model's parameters are set in a way that follows biological findings—by imitating the mapping of ganglions between the retina and V1. In order to use camera-made images as inputs, we reduced the field of view and the foveal resolution. Our simulations are done from initial uniform images with $1024 \times 682$ pixels, which are mapped to a *logimage* $L = u_{\mathrm{Max}} \times v_{\mathrm{Max}} = 38 \times 90$ *logpixels*.

## 2. The complex log mapping

The complex log mapping is commonly used in the literature as an approximation to the mapping of visual information in the brain—see for example Schwartz (1977, 1980) and others. The basic complex-log function maps a polar coordinate to a Cartesian point by $(u(r,\theta), v(r,\theta)) = (\log r, \theta)$. We follow Tistarelli and Sandini (1992, 1993), Wilson (1983), Yamamoto et al. (1996) and remove a small circular area from the center of the source image. We assume that this area, which corresponds to the visual field of the fovea, is treated separately.

In addition to Wilson (1983) and Yamamoto et al. (1996), we select the relevant constants in the model in a way that follows biological findings. In order to achieve meaningful dimensions and compression rates, we follow the path and number of ganglions.

### 2.1. Modeling the human retina

#### 2.1.1. Choosing the sampling model

According to Wilson (1983) and Yamamoto et al. (1996) the retinal surface is spanned by partially overlapping, round shaped receptive fields. Their centers form a complex log grid of $(u \times v)$ elements. The foveal area is excluded from the model. Using these assumptions—the eccentricity of the $n$th ring $(0 \leqslant n \leqslant u)$ $R_n$ is $R_0 \exp(wn/2p(1-o_v))$, where $w = \log(1 + (2(1-o_v)c_m/(2-(1-o_v)c_m)))$, $R_0$ is the radius of the fovea (in degrees), $c_m$ is the ratio between the diameter (in degrees) of the receptive field and its eccentricity (in degrees), $o_v$ is an overlap factor and $p$ is the number of photoreceptors in a radius of one receptive field. The radius of the receptive field on the $n$th ring is $c_m R_n/2$ and the number of receptive fields per ring is $v = 2\pi/(c_m(1-o_v))$. Ganglion cells appear to obey these assumptions (Sakitt and Barlow, 1982; Yamamoto et al., 1996). Following Schwartz (1980) and others, and extrapolating the model towards the $>30°$ periphery, we can use ganglions as the modeled elements and let $o_v = 0.5$.

#### 2.1.2. Choosing the constants for the sampling model

The retinal field of a single eye is $208° \times 140°$ (Rojer and Schwartz, 1990). The foveal area is a circle with a radius of 2.6° in the center of this field. We therefore let $R_0 = 2.6°$, and note that $R_u = 104°$. (The number of ganglions in the extreme periphery is very small, thus we neglect the fact that the field of view is not really circular.)

There are $\approx 10^6$ neurons in the main optic nerve, 75% of them are peripheral. The number of modules (halves of hypercolumns) in one hemifield of V1 is 2500, 1875 of them represent the periphery (Carlson, 1991).

Following Yamamoto et al. (1996) we first model the spatial mapping of modules. For $u$ and $c_m$, we solve $R_u = 104$ and $uv = 1875$, which gives $u = 33$, and $c_m = 0.221$, or a grid of $33 \times 56.8$ modules. If 750 000 ganglions are equally divided to the receptive fields, we get 400 cells per receptive field. Roughly assuming that these cells are equally distributed inside every field in a $20 \times 20$ matrix, we get a grid of $660 \times 1136$ ganglions for the peripheral area.

### 2.2. Modeling foveated images

We now adjust the constants we derived above to fit data that resembles camera images. This is done by reducing the size of the output—by selecting a smaller portion of view, and by reducing the size of the input—using a maximal (foveal) resolution that is lower than the one used by humans.

We first describe what the maximal resolution of the human vision is. Polyak's cones density in the foveaola matches Drasdo's foveal resolution of up to 30 000 ganglions/deg$^2$, assuming a ganglions/cones ratio of 2 (Polyak, 1957; Drasdo, 1989). We therefore define $F = 28\,648$ ganglions/deg$^2$ to be the maximal ganglions density (we ignore the density of 22 918 cones/deg$^2$ in the very central 20′ of the foveaola).

In practice images with a field of view of 208° are rarely used. We wish to model a field of view that corresponds to commonly used photo images, yet we would like the view not to be too narrow. Our suggested model uses the central 74° × 53° of the human visual field. This extent has the same field as a 24 mm camera lens, projected on a 35 mm film (Langford, 1978). It is wider than the human eye's projected extent (around 39° × 26°, achieved using a 50 mm lens), but not too wide to create distortions.

If we view this extent with the resolution $F$ we get 12 525 × 8970 pixels; Exclusion of the foveal area leaves $\approx 1.11 \times 10^8$ pixels. The logarithmic mapping of the extent yields $u \times v = 24 \times 56.8$ modules, or $480 \times 1136 \approx 545\,280$ ganglions. (The $v$ parameter remains the same and $u$ is the minimal $n$ such that $R_n > 74/2$.)

We now adjust the above extent to the dimensions of a typical screen, which has a lower resolution. Dividing the uniform extent by 12.66 in each dimension we fit it into 1024 × 682 pixels—a reasonable size for input images. On a 14″ monitor such an image spans ≈74° × 53°, when viewed from about 20 cm. We also reduce the sampling rate of the modules by the same factor—from $20^2$ ganglions to $1.6^2$. The 24 × 56.8 modules will now be composed of about $38 \times 89.8 \approx 3408$ ganglions.

### 2.3. Summary and example

Table 1 summarizes the dimensions of the human retina and our model. The middle column contains the information about the relevant extent of the human mapping, and the right column contains the data about the model itself (after the reduction in the foveal resolution).

The following pictures (Fig. 1) illustrate the sampling models we presented above. The top images show our 780 × 674 input image (left) and its 38 × 90 logimage (right). The center of the image was chosen to be the origin for our mapping. The bottom image shows the reverse mapping of that logimage (the central black area is the foveal area that was not mapped).

## 3. The structure of the logmap and operator tables

A logmap $L$ is a table with $u_{\mathrm{Max}} \times v_{\mathrm{Max}}$ entries, each containing a list of the uniform pixels that constitute the receptive field of this logpixel entry. Given that table, and if we assume (for simplicity) that the receptive field kernel is just averaging, we can transform a uniform image with pixel values $\bar{z}$ for each pixel $z$ to a logimage. For every logpixel $l$ with an attached list $\{z_1, \ldots, z_n\}$, we assign the logpixel value $\bar{l}$ to be $(1/n)\sum \bar{z_i}$.

An *operator table* $\mathrm{Op}_k(l)$ is defined $\forall l \in L$ as

$$\{(w_i, L_i^{(k)})\}_{i=1..N} \tag{1}$$

Table 1
Human and model's numerical data

| Parameter | Human eye | Eye's extent | Model |
|---|---|---|---|
| Fov. resolution (units/deg$^2$) | 28 648 | 28 648 | 180 |
| Horizontal field (deg) | 208 | 74 | 74 |
| Vertical field (deg) | 140 | 53 | 53 |
| Horizontal field in pixels ($w_u$) | 35 205 | 12 525 | 1024 |
| Vertical field in pixels ($h_u$) | 23 696 | 8970 | 682 |
| Uniform units | $8.34 \times 10^8$ | $1.12 \times 10^8$ | 706 910 |
| Uniform units (excluding fovea) | $8.33 \times 10^8$ | $1.11 \times 10^8$ | 703 080 |
| Log units (excluding fovea) | 750 000 | 545 280 | 3408 |
| Logmap's width (excluding fovea) | 660 | 480 | 38 |
| Logmap's height | 1136 | 1136 | 89.8 |
| Peripheral compression (x:1) | 1110 | 203 | 203 |
| $p$ | 10 | | 0.8 |
| $R_N$ | 104 | | 37 |

Additional constants are: $w = 0.11061$, $o_v = 0.5$, $c_m = 0.221$, $R_0 = 2.6$.

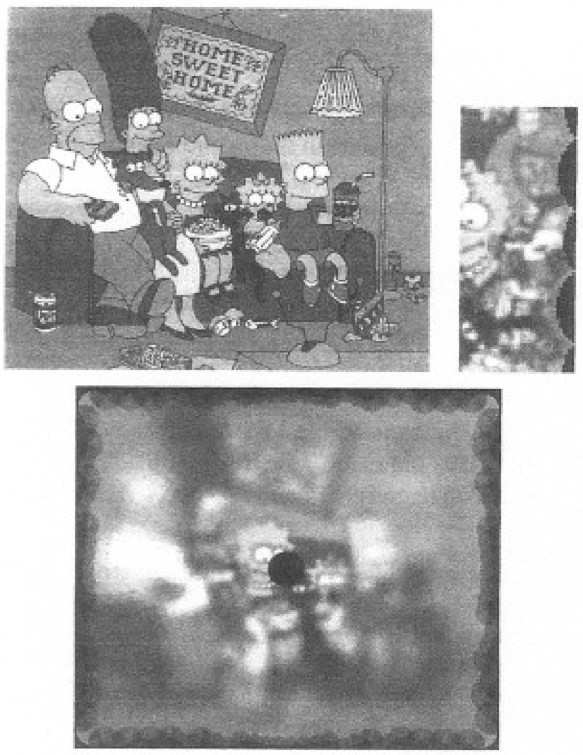


Fig. 1. Top left: an input image; top right: its logimage with a visual angle of 74°; bottom: reverse logmap.

where $L_i^{(k)} = (l_i^{(1)}, \ldots, l_i^{(k)})$ is a $k$-tuple of $l_i^{(j)} \in L$, $N$ is the number of elements in $l$'s operator table, and $k$ is constant.

These lists can be of different size (different values of $N$) for different logpixels. Assuming that $L_i^{(k)}$ occurs only once in such a list, $N$ is bound by $(u_{\text{Max}} v_{\text{Max}})^k$. However in practice $N$ is bound by much smaller constants, as will be shown.

Suppose that $F$ is a function of $k$ parameters. For any $l \in L$ we define $\text{Apply}(\text{Op}_k(l))$ as

$$\sum_{i=1}^{N} w_i F(\overline{l_i^{(1)}}, \ldots, \overline{l_i^{(k)}}) \tag{2}$$

For $k = 1$ we use $F(\overline{l_i^{(1)}}) = \overline{l_i^{(1)}}$. Applying is the process of "instantiating" the operator on a source logimage, resulting in a target logimage. Preparing the tables can be done in a preprocessing stage, leaving minimal work to be done at applying time. Usually the preprocessing and applying stages can be carried out in parallel for each logpixel.

An operator table of $l$ can be normalized by multiplying every weight in the list of $\text{Op}_k(l)$ by a given constant. Since each $l$ has a different index list, we can multiply each list by a different constant, thus normalizing a global operator.

Two operator tables can be added. The resulting table for each $l$ will contain all the $k$-tuples from the source tables; $k$-tuples that appeared in both tables will appear once in the resulting table with a summed weight.

$L_i^{(k)}$ is an *ordered* $k$-tuple. In some operators, however, there is no meaning to the order of the logpixels in the $k$-tuple. In these cases we can accumulate weights of equivalent $k$-tuples and further decrease the size of the operator table.

## 4. Edge and phase maps

Let $(u_0, v_0)$ be an arbitrary logpixel, with a corresponding field center of $(x_0, y_0)$ in the uniform space. Let $(s, t) = (\varphi(x, y), \psi(x, y))$ be the transformation defined by a translation of a Cartesian coordinate by $(-x_0, -y_0)$ and a rotation by $(-\arctan(y_0/x_0))$ (see Fig. 5).

Projecting a logimage to the uniform space, we can express the result as functions of both the $x$–$y$ and the $s$–$t$ coordinate systems, by

$$F(x, y) = f(\varphi(x, y), \psi(x, y))$$

It can be easily shown that $(\partial F/\partial x)^2 + (\partial F/\partial y)^2 = (\partial f/\partial s)^2 + (\partial f/\partial t)^2$. Thus the *magnitude* of the gradient vector $\nabla F$ can be approximated using the $s$–$t$ system as follows: we define $G_s$, $G_t$ and $\nabla L(u, v)$ to be

$$\begin{aligned} G_s(u, v) &= \overline{(u+1, v)} - \overline{(u-1, v)} \\ G_t(u, v) &= \overline{(u, v+1)} - \overline{(u, v-1)} \\ \nabla L(u, v) &= (G_s^2 + G_t^2)^{0.5} \end{aligned} \tag{3}$$

$(G_t/G_s)$ approximates the tangent $\nabla F$, relative to the $s$–$t$ coordinates. Since this system is rotated, the phase of $L$ will be

$$\phi L(u, v) = \arctan\left(\frac{G_t}{G_s}\right) + \frac{2\pi v}{v_{\text{Max}}} \tag{4}$$

## 5. Spatial foveated mask operators

We suggest operators that use different masking scales: logpixels that reside on peripheral areas will have a larger effective neighborhood than central logpixels (using fixed-scaled neighborhood yields poor peripheral detection). Suppose that we are given a spatial mask $g$ with $M \times N = (2\hat{M}+1)\times(2\hat{N}+1)$ elements, and let $\lambda \in \Re^+$ be an arbitrary constant. We mark the rounded integer value of $x \in \Re$ by $\lceil x \rceil$, the set of pixels in $l$'s receptive field by $R_l$, and its size by $|R_l|$. For any uniform pixel $(x, y)$ we can find the closest matching logpixel $l$, and define a neighborhood $P$ around $(x, y)$ as

$$\{(\lceil x + m\lambda\sqrt{|R_l|}\rceil, \lceil y + n\lambda\sqrt{|R_l|}\rceil)\}$$

where $|m| \leqslant \hat{M}$, $|n| \leqslant \hat{N}$ are reals.

Every $p \in P$ corresponds to a logpixel $l_p \in L$. We add the logpixels $l_p$ to $l$'s operator list; Each addition of $l_p$ that corresponds to a pixel $(x + m\lambda\sqrt{|R_l|}, y + n\lambda\sqrt{|R_l|})$ will have a weight of

$$\frac{g(\lceil m \rceil, \lceil n \rceil)}{\lambda^2 |R_l|^2}$$

$\lambda$ is used to control our masking area (usually $\lambda = 1$). Normalizing the weight compensates for our sampling method: $|R_l|$ compensates for the several additions to the same $l_p$ that may occur using different $(x, y)$ pairs. An additional $\lambda^2|R_l|$ compensates for the different sizes of $P$. Note: in Wallace et al. (1994) this normalization is done in the "applying" stage.

The resulting table can be applied using the standard applying mechanism (see Eq. (2)).

Mask building and applying can be viewed in terms of translation tables (Wallace et al., 1994). A translation by $(x, y)$ can be viewed as a spatial mask $T_{(x,y)}$, where $T(-x, -y) = 1$ and $T(i, j) = 0$ for every other $(i, j)$. We build $\overline{T}_{(x,y)} = T_{(x\lambda\sqrt{|R_l|}, y\lambda\sqrt{|R_l|})}$ for each of the possible offsets. Any mask operator $G$ and its apply function can now be defined as

$$G(u, v) = \sum_{m=-M}^{M} \sum_{n=-N}^{N} g(m, n)\overline{T}_{(m,n)}(u, v)$$

$$\mathrm{Apply}(G(u,v)) = \sum_{m=-M}^{M} \sum_{n=-N}^{N} g(m,n)\mathrm{Apply}(\overline{T}_{(m,n)}(u,v))$$

## 6. The foveated Hough transform

We construct a Hough map that detects lines in a given edge-logimage. Let $\Gamma$ be a set of $k$ angles, $\Gamma = \{\gamma_i | 1 \leqslant i \leqslant k, \gamma_i = (2\pi i/k)\}$, let $z$ be an arbitrary pixel in a uniform image $Z$, and $q$ the respective logpixel in $L$.

For each $z \in Z$ we find the parameterization of the $k$ lines $\lambda_i$ passing through $z$ and having angles of $\gamma_i$, respectively. We define $(\rho_i, \theta_i)$ (the coordinates of the normal vector of $\lambda_i$ that passes through 0) as these parameterizations. For an arbitrary $i$ we observe $(u(\rho_i, \theta_i), v(\rho_i, \theta_i))$. For $u_0 \leqslant \rho_i \leqslant u_{\mathrm{Max}}$ and $\theta_i$ there is a logpixel $p \in L$ with these coordinates. Thus we can add the coordinates of $q$ to $p$'s operator table, which will function as the voting plane of the Hough transform (see Fig. 2). Note that the actual results of the voting depend on the logpixels' *values*. They can be calculated once a logimage with the values $\bar{q}$ is given.

The operator table of a single $p$ is made of a series of logpixels which lie on a "band" in $Z$ (see Fig. 3). That band passes through its parameterizing source logpixel, and is orthogonal to the line $(\rho_i, \theta_i)$. We define the *thickness* of that band as the number of pixels along $(\rho_i, \theta_i)$ that intersect $p$. As

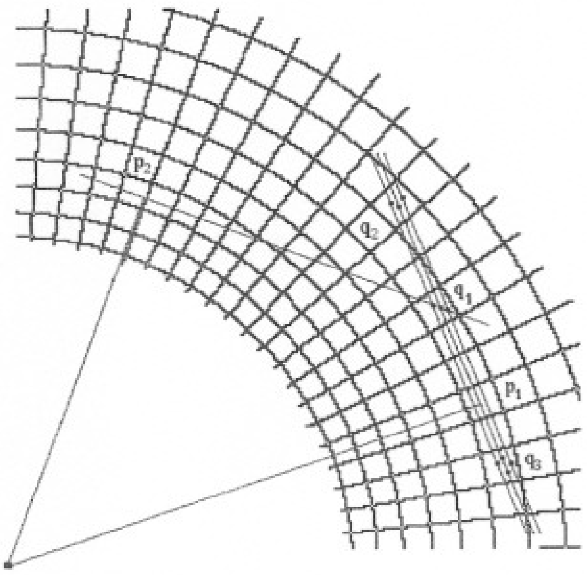


Fig. 2. Contribution of logpixels to the foveated Hough map.

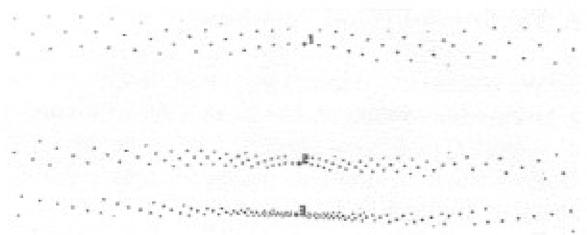


Fig. 3. Visualization of contributers to three logpixels in the foveated Hough map.

the $u$-value of a logpixel $p$ gets larger, more parameterizations fall into the same $p$: the number of these contributers increases linearly with $p$'s thickness, which can be shown to be proportional to the diameter of its receptive field. We therefore normalize the table during its construction, by dividing every contribution to a logpixel $p$ by $p$'s diameter.

Fig. 2 shows an example of contributions to the Hough map. The pixels that are in $q_1$'s field can participate in many lines that pass through many logpixels. As an example, three pixels were chosen. Through these pixels, three lines in one direction and one line in a different direction are shown. The parameterization of the three lines leads us to $p_1$: the normal vector to these lines (also drawn in the figure) intersects the lines at $p_1$. The parameterization of the other line points to $p_2$. Thus the operator table of $p_1$ will contain the coordinates of $q_1$, and likewise, the operator table of $p_2$ will contain the coordinates of $q_1$ too. The operator table of $p_1$ will also contain the coordinates of $q_2$ and $q_3$, because the three drawn lines pass through their fields as well.

Fig. 3 shows a visualization of the Hough operator for three logpixels (marked 1…3). The figure shows the upper half of a uniform image, with every logpixel projected back to this plane. The three logpixels have the same value of $v$. Each logpixel that is a member in the operator's weighted list (of one of 1…3's lists) is shown as a dark dot in the figure. As can be seen, the operator table of each of the logpixels 1…3 is made of a series of logpixels that lie on a band. That band passes through its corresponding source logpixel, and is orthogonal to the line that connects that logpixel with the origin.

## 7. The foveated symmetry operator

The generalized symmetry transform (Reisfeld et al., 1995) and other related mechanisms DiGesu et al. (1997), use symmetry in order to detect regions of interest. These operators specifically detect corners or centers of objects. We implement here the foveated version of Reisfeld et al. (1995). As in the case of mask operators, our operator is *scale dependent*; It detects both corners and centers, smaller near the fovea and larger in the periphery. Our input is a set of logpixels $l_k = (u_k, v_k)$, from which an edge logmap $(r_k, \theta_k) = (\log(1 + \|\nabla(e_k)\|), \arg(\nabla(e_k)))$ can be obtained. We define $L^{-1}(l)$ as the uniform pixel that corresponds to the center of $l$'s receptive field. Our operator table for a logpixel $l$ will be of the form $\{w_i, (l_{ai}, l_{bi})\}_{i=1..N}$ (assigning $k = 2$, $F(\overline{l_{ai}}, \overline{l_{bi}}) = \overline{l_{ai}}\,\overline{l_{bi}}$ in Eqs. (1) and (2)).

For any logpixel $l$ we find all the pairs $(l_a, l_b)$ of its circular neighborhood $\Gamma(l)$: we traverse all the possible logpixels-pairs $(l_a, l_b)$, find $\mathrm{mid}(l_a, l_b)$ and add this pair to $l$'s list if $\mathrm{mid}(l_a, l_b) = l$. $\mathrm{mid}(l_a, l_b)$ is defined to be the closest matching logpixel to the pixel $((L^{-1}(l_a) + L^{-1}(l_b))/2)$. When adding a pair, we compute $\sigma$ and add a weight of $\mathscr{D}^{\star}_{\sigma}$ to that pair in $l$'s list. $\sigma$ and $\mathscr{D}^{\star}_{\sigma}$ are defined as

$$\sigma(l_a, l_b) = \lambda\sqrt{|R_{\mathrm{mid}(l_a, l_b)}|}$$

$$\mathscr{D}^{\star}_{\sigma}(l_a, l_b) = \begin{cases} \mathrm{e}^{-\frac{\|L^{-1}(l_a) - L^{-1}(l_b)\|^2}{2\sigma^2(l_a, l_b)}} & \text{if it is} > \mathrm{e}^{-1} \\ 0 & \text{otherwise} \end{cases}$$

$\lambda$ is a constant that acts as the symmetry radius (similar to $\sigma$ in Reisfeld et al. (1995)). After the construction is done, we divide each weight by the number of elements in its respective list. The number of elements in each list is approximately the same for all the lists in a single table. Thus the normalization is done only to equalize weights of tables with different $\lambda$ values.

Applying the symmetry operator for $l$ results in

$$\sum_{i=1}^{N} w_i P(a_i, b_i) r_{a_i} r_{b_i},$$

where $\alpha_{ij}$ is the angle between the $x$-axis and the line $(a_i, b_i)$; and $P(a_i, b_i) = (1 - \cos(\theta_{a_i} + \theta_{b_i} - 2\alpha_{ij}))(1 - \cos(\theta_{a_i} - \theta_{b_i}))$ (for a detailed definition, see Reisfeld et al. (1995)).

## 8. Complexity

### 8.1. Radial invariance

Suppose that we are given an arbitrary operator $\mathrm{Op}_k$. We define the difference operator $\mathrm{Op}\Delta_{nk}(u, v)$ as

$$\mathrm{Op}_k(u, v \hat{+} n) - \mathrm{Shift}(\mathrm{Op}_k(u, v), n)$$

where $\hat{+}$ stands for $+_{\mathrm{mod}\, v_{\mathrm{Max}}}$. We also define the shift operator of $\mathrm{Op}_k$ by $n$ as

$$\mathrm{Shift}(\mathrm{Op}_k(u, v), n) = \left\{ \left( w_i, (u_i, v_i \hat{+} n)^{(1)}, \ldots, (u_i, v_i \hat{+} n)^{(k)} \right) \right\}_{i=1..N}$$

We say that $\mathrm{Op}_k$ has *radial invariance* if for every $(u, v)$, the squared sum of weights in $\mathrm{Op}\Delta_{nk}(u, v)$ is $\ll$ from the one of $\mathrm{Op}_k$. If an operator is radial invariant, only one representative for each $u$ in the table needs to be stored. This cuts the storage complexity and preprocessing time by a factor of $v_{\mathrm{Max}}$. We usually choose a fixed $v_0$ and take all the representatives to be of the form $(u, v_0)$.

For example, it can be seen from Fig. 4 that the symmetry operator is radial invariant: the figure shows a logpixel $l_1$ and a pair in its contributers list $(l_{1a}, l_{1b})$. If we rotate $l_1$ to the position of $l_2$, the contribution pair is rotated respectively (rotation by the same angle) to $(l_{2a}, l_{2b})$.

### 8.2. Operator complexities

Our operators are usually built in a preprocessing stage by an exhaustive run over uniform pixels, in order to produce tables that are small and quick to traverse. As can be seen from Eq. (2), the time complexity of applying an operator for a single logpixel is equal to its space complexity. We bind the number of logpixels $n$ in the table of an arbitrary logpixel by factors that are proportional to the small size of the logimage ($u$ and $v$). Using the radial invariance of operators, we reduce the number of logpixels whose tables we keep in memory, resulting a total space complexity of $O(nu_{\mathrm{Max}})$ for an operator table. Time complexity is

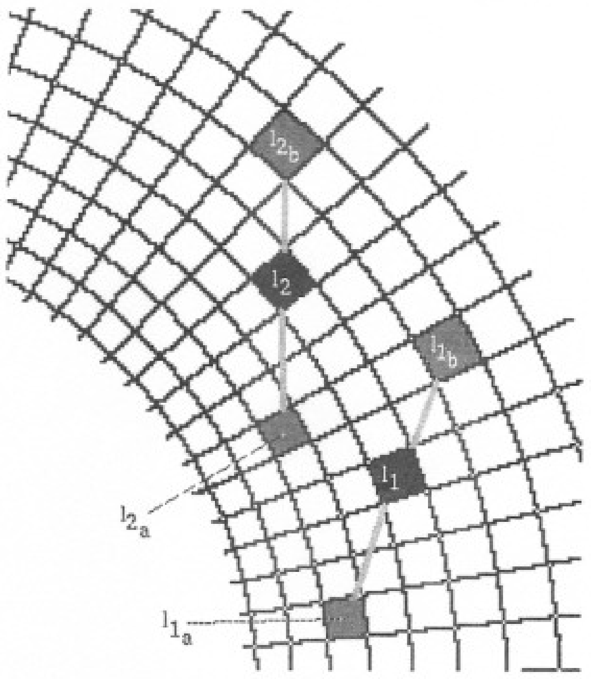


Fig. 4. Contributers to the symmetry operators. See text inside.

therefore $O(nu_{\mathrm{Max}}v_{\mathrm{Max}})$ if applying is done serially, and $O(n)$ if $u_{\mathrm{Max}}v_{\mathrm{Max}}$ processors apply a table simultaneously.

#### 8.2.1. Hough operator

For the Hough operator, the preprocessing time is $O(|Z|^{1.5})$, derived by naively taking all $z \in Z$ and walking along the $k$ parameterization lines for each $z$.

To yield the operator's space complexity we note that a list with a length of $O(u_{\mathrm{Max}} + v_{\mathrm{Max}})$ is attached for each $l$: for each of the logpixels, the Hough table is produced by walking along a line that corresponds to an arbitrary Hough value in $l$. We super-impose the line on the uniform image $Z$, and count the number of times that the uniform pixels moved to a different receptive field. We hold two counters, charging the first one when the line moves to a field with a different $u$ coordinate, or the second one when the line moves to a field with a different $v$ coordinate. It is clear that the line cannot cause the first counter to grow more than $2u_{\mathrm{Max}}$ times, and cannot cause the second counter to grow more than $v_{\mathrm{Max}}/2$ times. The number of such lines that will contribute to the list of $l$ depends on $u$ (which determines the thickness of the contributing "band" (see Fig. 3). However, since this number is linearly proportional to $u$, the change in the number of contributing *logpixels* as $u$ varies can be bound by a constant factor.

The space complexity of a list for a single logpixel $l$ is therefore $O(u_{\mathrm{Max}} + v_{\mathrm{Max}})$. Since the Hough operator is radial-invariant, the space complexity of the whole operator is $O(u_{\mathrm{Max}}(u_{\mathrm{Max}} + v_{\mathrm{Max}}))$.

In our model representation, the number of elements in a single list of $l$ was bound by 140, with an average number of 110, i.e. the binding complexity constant is $\approx 1$.

### 8.2.2. Symmetry operator

Similarly, it can be shown that the total space complexity for the symmetry operator is $O(u_{\mathrm{Max}} v_{\mathrm{Max}})$. We first show that the number of elements in each logpixel table can be bound by a constant.

The operator table for each logpixel $l = (u, v)$ has a finite circular support $S$ whose center is at $l$'s center in the uniform space (because $\mathscr{D}$ is a Gaussian with a finite support). Its radius in pixels is $k\lambda\sqrt{|R_l|}$, where $k$ is constant. We now bind the number of logpixels that cover $S$. Let $l_{-i} = (u-i, v)$, for any $0 \leqslant i \leqslant u$, and let $c = |R_{l_{-i}}|/|R_{l_{-(i+1)}}|$ ($c > 1$ is a constant regardless of $i$ and $u$). The radius of $S$ with the maximal number of logpixels resides along the line from $l$ to the origin (because the size of the logpixels' receptive fields along this radius will shrink in the fastest rate)—thus its respective logpixels would be $\{l, l_{-1}, l_{-2}, \ldots\}$. Keeping in mind that the fields partially overlap—we sum the number of uniform pixels along this line until it gets outside $S$:

$$k\lambda\sqrt{|R_l|} < (1 - o_v)\sqrt{\frac{|R_l|}{\pi}}\left(1 + \frac{1}{c} + \frac{1}{c^2} + \cdots + \frac{1}{c^{\mu}}\right)$$

$\mu$ is *constant*, regardless of $u$. Therefore the number of logpixels in a radius of $S$ is bound by $(\mu + 1)$ and therefore $S$ is bound by a constant number of logpixels ($\approx \pi(\mu + 1)^2$).

The symmetry operator is radial invariant, thus it can be represented by $O(u_{\mathrm{Max}})$ elements and the bound on the preprocessing time can be tightened to $O(u_{\mathrm{Max}})$.

In our implementation the maximal number of elements in a single table entry of the symmetry operator is 89 for $\lambda = 2$ and 315 for $\lambda = 4$. The average numbers are 72 and 290, respectively.

### 8.2.3. Mask operator

The preprocessing time for the mask operators is large (computations for each mask element is $O(|Z|)$). However, by definition, the rectangular neighborhood (in the uniform space) $P$ of any logpixel $l$ can be covered by a constant number $(2\hat{M}2\hat{N})$ of logpixels. Therefore the resulting mask operators require only $O(u_{\mathrm{Max}} v_{\mathrm{Max}})$ space.

In our model representation, a logpixel in a $5 \times 5$ translation table had a maximum of 25 elements in its table. In an example of a $5 \times 5$ corner operator (see Fig. 12)—the maximal number of elements in a table of a single logpixel was 71 and the average number was 64.

## 9. Results

### 9.1. Edge detection

Fig. 5 demonstrates our edge operator. We transferred the upper left image to a logimage and applied the edge operator. The back-projection of the result is shown in the bottom right image, and can be compared with the uniform equivalent (bottom left). An example of an $s$–$t$ system for a specific logpixel $l$ (marked by 5) is shown on the upper right side.

### 9.2. Hough operator

Fig. 6 shows the uniform visualizations of a Hough logmap (middle image) of an input image with a horizontal edge (left image). Note that the centers of the images are positioned at the center of the fovea's hole. Thus the most prominent line can be derived by taking the line that is orthogonal to the vector between the origin and the marked point (this normal is drawn in the right image). The right image in Fig. 6 shows the superpositions of the relevant edge with its Hough map. The Hough map indeed points to the correct line.

Fig. 7 shows several input logimages (shown at the left side in every triple) and their Hough maps.

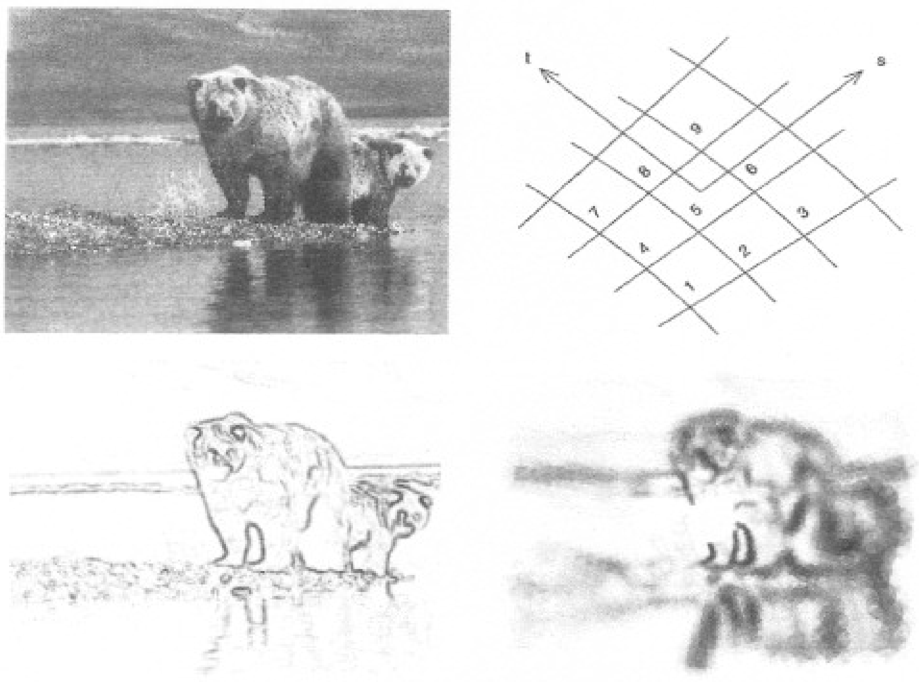


Fig. 5. Foveated edge detection.

The edge operator was applied on each logimage. Each edge logimage corresponds to a uniform image of a horizontal line in a different location. The log-edges have different length and thickness, although their uniform representations are similar. The Hough operator was applied on the edge logmaps (shown in the right side of every triple). The logpixels that got the largest number of votes (marked in each logmap) indeed point to the correct lines. They reside on the same orientation (have an approximately equal $v$ value), with decreasing $u$-values. The voting scores of the points are similar.

Fig. 8 shows that different positions of segments have the same contribution to the Hough map. The left figures show segments that were taken from a

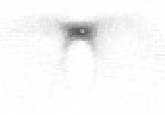
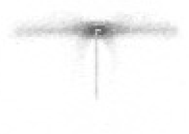
| Image | Hough | Hough+Edge |
|---|---|---|
| |  |  |

Fig. 6. Example of an input image that contains a line and the resulting Hough map.

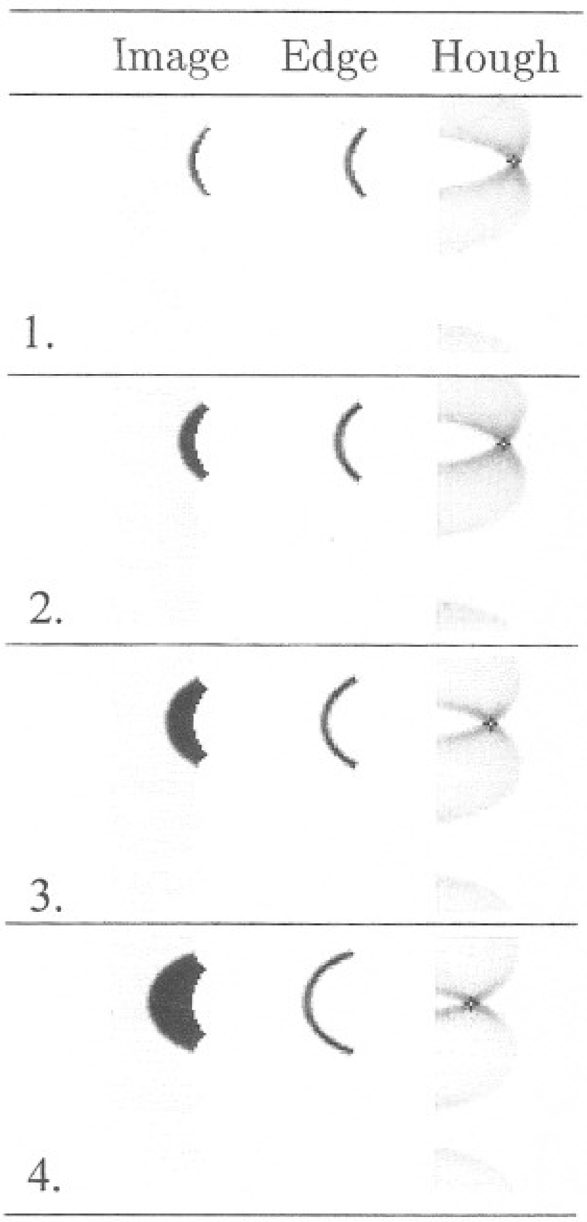


Fig. 7. Foveated Hough maps of horizontal lines-based log-images.

single horizontal bar. The segments' $x$ position begins at the near-left area of the image and ends near the origin (the other half is symmetric). The middle figures show the edges of the segments. Note that segments that are closer to the fovea have finer boundaries. The right figures show the

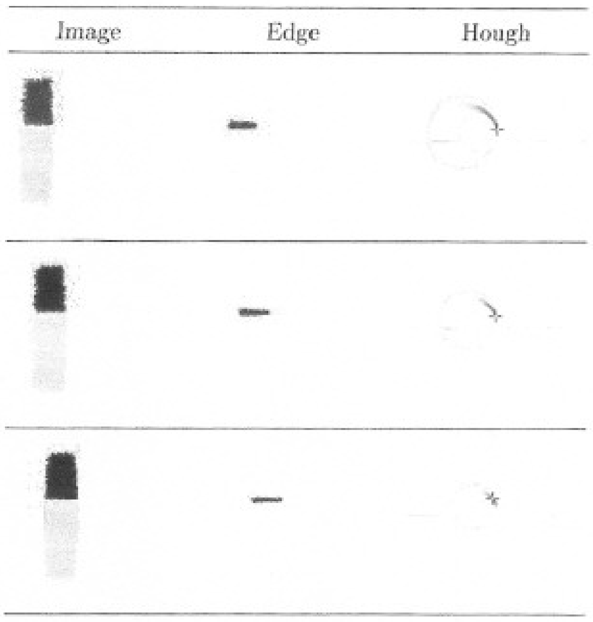


Fig. 8. Foveated Hough maps of line segments.

Hough maps of the line segments, with the best voting cells marked. The same line is detected for all segments, although the difference in contributions causes different distribution of votes. The total votes are very similar: 11 939, 11 901, 11 591.

The Hough operator is also demonstrated on an image with five segments that was transferred into logimages using two fixation points (Fig. 9, top). The edge and Hough operators were applied (middle) and the five highest local maxima were marked. The derived lines are drawn over the edge-logimage (bottom), they all had similar votes.

## 9.3. Symmetry operator

For symmetry extraction we created a uniform image with six squares. The squares were placed at different distances from the fovea, and their size was proportional to their eccentricity. The applied operators when $\lambda = 2$ and 4 are shown in Figs. 10 and 11, respectively. The left images in each of these figures show the resulting logimages, the right images show the projection along with the source image. The symmetry operators indeed detect the corners of the squares (for $\lambda = 2$) or their centers (for $\lambda = 4$).

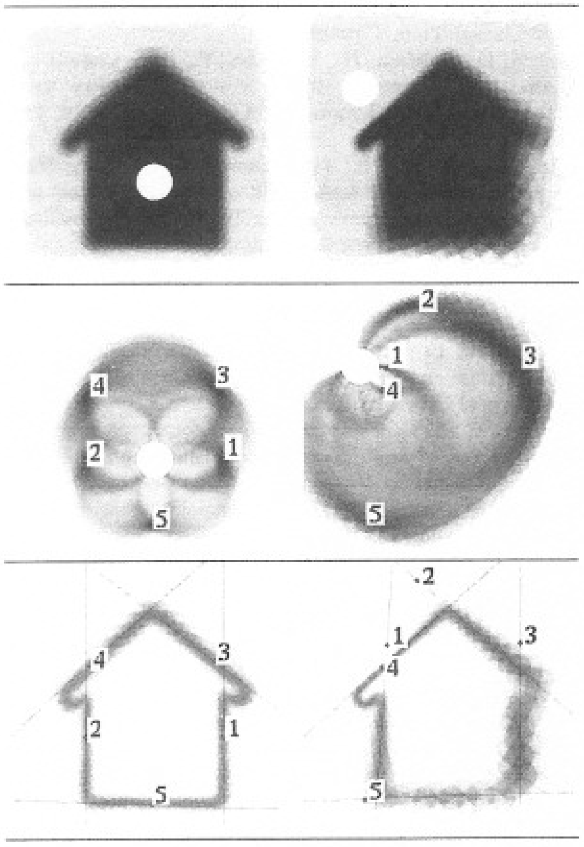


Fig. 9. The foveated Hough operator. See text for details.

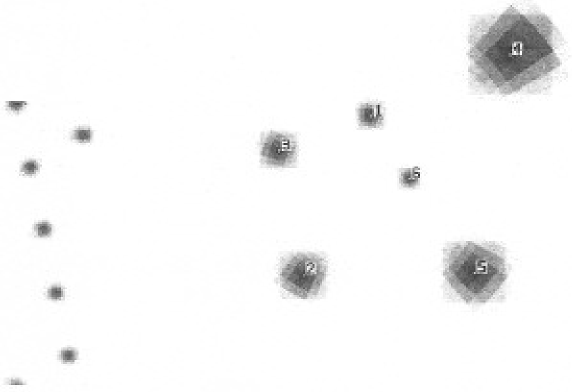


Fig. 11. Foveated centers detection using the symmetry operator. Highest peaks designated by their order.

## 9.4. Spatial mask operators

To demonstrate a spatial mask detector, we set a 5 × 5 mask that detects "$\Gamma$"-shaped corners. The image in Fig. 12 (left) is used as our input; It is a logmap of uniform squares with sizes that are proportional to the eccentricity of their upper-left corners.

We constructed a set of translation operators and used them to construct a corner-detector. The result of applying the operator (along with the original image) is shown in Fig. 12 (right).

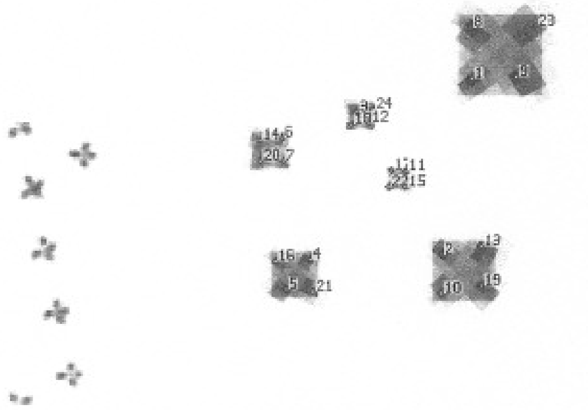


Fig. 10. Foveated corner detection using the symmetry operator. Highest peaks designated by their order.

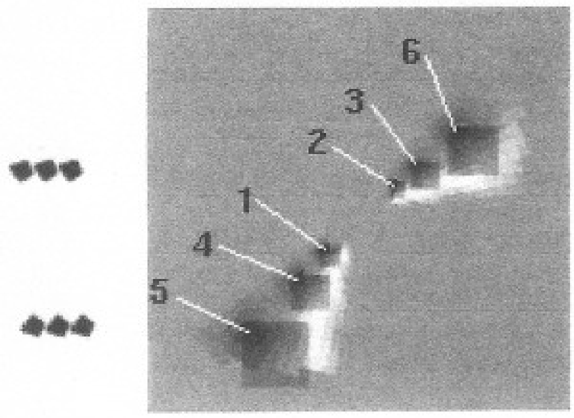


Fig. 12. Corner detection using a foveated spatial mask. Highest peaks designated by their order.

## 10. Conclusions

In this paper we have presented an efficient mechanism that can be used to implement various image analysis operators directly on a foveated image. The method is based on a space variant weighted data structure. Using this approach, we show how some common global and local operators can be implemented directly, at almost frame rate, on the foveated image.

## Acknowledgements

Supported by the ISF center of excellence for Geometry, by the Minkowski Minerva center for applied Geometry, and by the MOST.

## References

Carlson, N., 1991. Physiology of behavior, fourth edition Allyn and Bacon, Maryland.

DiGesu, V., Valenti, C., Strinati, L., 1997. Local operators to detect regions of interest. Pattern Recognition Letters 18, 1077–1081.

Drasdo, N., 1989. Receptive field densities of the ganglion cells of the human retina. Vision Research 29, 985–988.

Jain, R., Kasturi, R., Schunk, B., 1995. Machine Vision. McGraw Hill, New York.

Langford, M., 1978. The Step by Step Guide to Photography. Dorling Kindersley, London.

Polyak, S., 1957. The Vertebrate Visual System. The University of Chicago Press, Chicago.

Reisfeld, D., Wolfson, H., Yeshurun, Y., 1995. Context free attentional operators: the generalized symmetry transform. International Journal of Computer Vision 14, 119–130.

Rojer, A., Schwartz, E., 1990. Design considerations for a space-variant visual sensor with complex logarithmic geometry. In: Proceedings of the 10th IAPR International Conference on Pattern Recognition, pp. 278–285.

Sakitt, B., Barlow, H.B., 1982. A model for the economical encoding of the visual image in cerebral cortex. Biological Cybernetics 43, 97–108.

Schwartz, E., 1977. Spatial mapping in the primate sensory projection: analytic structure and relevance to perception. Biological Cybernetics 25, 181–194.

Schwartz, E., 1980. Computational anatomy and functional architecture of striate cortex: a spatial mapping approach to perceptual coding. Vision Research 20, 645–669.

Tistarelli, M., Sandini, G., 1992. Dynamic aspects in active vision. Journal of Computer Vision, Graphics, and Image Processing: Image Understanding 56 (1), 108–129.

Tistarelli, M., Sandini, G., 1993. On the advantages of polar and logpolar mapping for direct estimation of time-to-impact from optical flow. IEEE Transactions Pattern Analysis and Machine Intelligence 15 (4), 401–410.

Wallace, R., Ong, P.W., Bederson, B., Schwartz, E., 1994. Space variant image processing. International Journal of Computer Vision 13 (1), 71–90.

Wilson, S., 1983. On the retino-cortial mapping. International Journal of Man–Machine Studies 18, 361–389.

Yamamoto, H., Yeshurun, Y., Levine, M., 1996. An active foveated vision system: attentional mechanisms and scan path convergence measures. Journal of Computer Vision and Image Understanding 63 (1), 50–65.