An Efficient Data Structure for Feature Extraction in a Foveated Environment

Efri Nattel and Yehezkel Yeshurun *

The Department of Computer Science
The Raymond and Beverly Sackler Faculty of Exact Sciences
Tel-Aviv University
Tel-Aviv, 69978, Israel
{nattel, hezy}@math.tau.ac.il

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.

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 ([14,13,16]). 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 ([14]), and show how it is efficiently used for direct implementation of feature detection on foveated images. Following the classification of Jain (in [3]), we show how local, global, and relational (edge detection, Hough Transform and Symmetry detection, respectively) are implemented by our method.

Both our source images and the feature maps are foveated, based on Wilson's model ([16,15]). To achieve reasonable dimensions and compression rates, the 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, we reduced the field of view and the foveal resolution. Our simulations are done from initial uniform images with 1024×682 pixels, which are mapped to a *logimage* $L = u_{\mathrm{Max}} \times v_{\mathrm{Max}} = 38 \times 90$ *logpixels*.

* This work was supported by the Minerva Minkowski center for Geometry, and by a grant from the Israel Academy of Science for geometric Computing.

S.-W. Lee, H.H. Buelthoff, T. Poggio (Eds.): BMCV 2000, LNCS 1811, pp. 217–226, 2000.

2 The Complex Log Mapping

The complex log mapping was suggested as an approximation to the mapping of visual information in the brain ([10,11] 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 [12,13,15,16] and remove a circular area from the center of the source image, assuming that it is treated separately.

In addition to [15,16], 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

. The sampling model: According to [15,16] the retinal surface is spanned by partially overlapping, round-shaped receptive fields; Their centers form a complex log grid. The foveal area is excluded from the model. Using these assumptions – the eccentricity of the nth ring ($0 \le n \le u$) R_n is $R_0 e^{\frac{w \cdot n}{2\pi(1-o_v)}}$, where $w = \log\left(1 + \frac{2(1-o_v)c_m}{2-(1-o_v)c_m}\right)$, 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 nth ring is $\frac{c_m \cdot R_n}{2}$ and the number of receptive fields per ring is $v = \frac{2\pi}{c_m(1-o_v)}$. Ganglion cells appear to obey these assumptions ([9,16]). Following [11] and others, and extrapolating the model towards the $> 30°$ periphery, we can use ganglions as the modeled elements and let $o_v = 0.5$.

. Field of view: The retinal field of a single eye is $208° \times 140°$ ([5,8]). 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).

. Number of modules: 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 ([1]).

. Computing u, c_m and p: Following [16] we first model the spatial mapping of modules. For u and c_m, we solve $R_u = 104$ and $u \cdot v = 1875$, which gives $u = 33$, and $c_m = 0.221$, or a grid of 33×56.8 modules. If 750000 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 20×20 matrix, we get a grid of 660×1136 ganglions for the peripheral area.

2.2 Modeling Foveated Images

. Foveal resolution: Polyak's cones density in the foveaola matches Drasdo's foveal resolution of up to 30000 *ganglions/deg*^{2}, assuming a *ganglions/cones*

ratio of 2 ([6,2]). We therefore define $F = 28648\, ganglions/deg^2$ to be the maximal ganglions density (we ignore the density of 22918 $cones/deg^2$ in the very central $20'$ of the foveaola).

. **Selecting the portion of the view:** In practice images with a field of view of 208° are rarely used. Our suggested model uses the central $74° \times 53°$ of the human visual field. This portion has the same field as a $24mm$ camera lens, projected on a $35mm$ film ([4]). It is wider than the human eye's portion (around $39° \times 26°$, achieved using a $50mm$ lens), but not too wide to create distortions.

Viewing this portion with the resolution F we get 12525×8970 pixels; Excluding the foveal area leaves $\approx 1.11 \cdot 10^8$ pixels. The logarithmic mapping of the portion results $u \times v = 24 \times 56.8$ modules, or $480 \times 1136 \approx 545280$ ganglions (The v parameter remains the same and u is the minimal n such that $R_n > \frac{74}{2}$).

. **Lowering the foveal resolution:** We now adjust the above portion to the dimensions of a typical screen, which has a lower resolution. Dividing the uniform portion 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 takes $\approx 74° \times 53°$, when viewed from about $20cm$. 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.

Table 1. Numerical data about the human eye and our model. Additional constants are: $w = 0.11061$, $o_v = 0.5$, $c_m = 0.221$, $R_0 = 2.6$.

Parameter	Human eye	Portion of the eye	Model
Fov. resolution ($units/deg^2$)	28648	28648	180
Horizontal field	208	74	74
Vertical field	140	53	53
Horiz. field (w_u)	35205	12525	1024
Vert. field (h_u)	23696	8970	682
Uniform units	$8.34 \cdot 10^8$	$1.12 \cdot 10^8$	706910
Uniform units (ex. fovea)	$8.33 \cdot 10^8$	$1.11 \cdot 10^8$	703080
Log units (ex. fovea)	750000	545280	3408
Logmap's width (ex. 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

The following pictures (Figure 1) illustrate the sampling models we presented above. The left image shows our 780×674 input image and the middle image

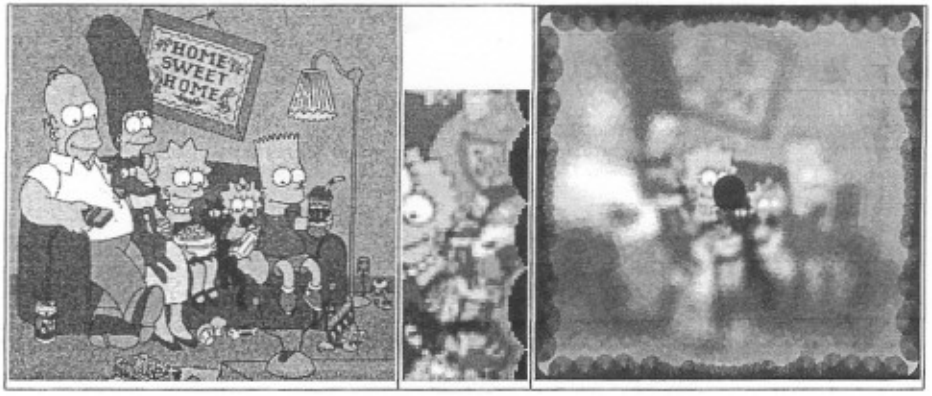

Fig. 1. Left: An input image; Middle: Its Logimage; Right: Reverse logmap

shows its 38×90 logimage. The center of the image was chosen to be the origin for our mapping. The right image shows the reverse mapping of that logimage (the central black area is the foveal area that is not mapped).

3 The Logmap and Operator Tables

A *logmap* is a table with $u_{\text{Max}} \times v_{\text{Max}}$ 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's kernel is just averaging, we can transform a uniform image with pixel values $\overline{z}$ for each z to a *logimage*. For every l with an attached list $\{z_1, \ldots z_n\}$, we assign the logpixel value $\overline{l}$ to be $\frac{1}{n}\sum \overline{z_i}$.

An *operator table* $Op_k(l)$ is defined $\forall l \in L$ as

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

where $L_i^{(k)} = \left(l_i^{(1)}, \ldots, l_i^{(k)}\right)$ is a k-tuple of $l_i^{(j)} \in L$, and k is constant.

Suppose that F is a function of k parameters. For any $l \in L$ we define Apply$(Op_k(l))$ 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 $Op_k(l)$ 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 on both tables will appear once in the resulting table with a summed weight.

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 $(-\arctan(y_0/x_0))$. 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 $\left(\frac{\partial F}{\partial x}\right)^2 + \left(\frac{\partial F}{\partial y}\right)^2 = \left(\frac{\partial f}{\partial s}\right)^2 + \left(\frac{\partial f}{\partial t}\right)^2$. Thus the *magnitude* of the gradient vector ∇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 ∇F, relative to the s-t coordinates. Since this system is rotated, the phase of L will be

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

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 $\left\{\left(\left\lceil x + m\lambda\sqrt{|R_l|}\right\rceil, \left\lceil y + n\lambda\sqrt{|R_l|}\right\rceil\right)\right\}$, where $|m| \le \hat{M}, |n| \le \hat{N}$ 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 $\left(g(\lceil m \rceil, \lceil n \rceil)/(\lambda^2 |R_l|^2)\right)$. λ 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 [14] this normalization is done in the "applying" stage.

Mask building and applying can be viewed in terms of translation tables (see [14]). A translation by (x, y) can be viewed as a spatial mask $T_{(x,y)}$, with $T(-x, -y) = 1$ and $T(i, j) = 0$ for every other (i, j). For each of the possible offsets we build $\bar{T}_{(x,y)} = T_{(x\lambda\sqrt{|R_l|}, y\lambda\sqrt{|R_l|})}$ Any mask operator G can now be defined as $G(u, v) = \sum_{m=-M}^{M}\sum_{n=-N}^{N} g(m, n)\cdot\bar{T}_{(m,n)}(u, v)$, giving $\text{Apply}(G(u, v)) = \sum_{m=-M}^{M}\sum_{n=-N}^{N} g(m, n) \cdot Apply(\bar{T}_{(m,n)}(u, v))$.

6 The Foveated Hough Transform

We construct a Hough map that detects lines in a given edge-logimage. Let Γ be a set of k angles, $\Gamma = \{\gamma_i | 1 \leq i \leq k, \gamma_i = \left(\frac{2\pi i}{k}\right)\}$, 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 λ_i passing through z and having angles of γ_i, respectively. We define (ρ_i, θ_i) (the coordinates of the normal vector of λ_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 $u_0 \leq \rho_i \leq u_{\text{Max}}$ and θ_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. Note that the actual results of the voting depend on the logpixels' *values*. They can be calculated once a logimage with the values $\overline{q}$ is given.

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

7 The Foveated Symmetry Operator

We show a foveated version of the Generalized Symmetry Transform ([7]), that detects corners or centers of shapes. 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 $k = 2$, $F(\overline{l_{ai}}, \overline{l_{bi}}) = \overline{l_{ai}} \cdot \overline{l_{bi}}$ in (1), (2)).

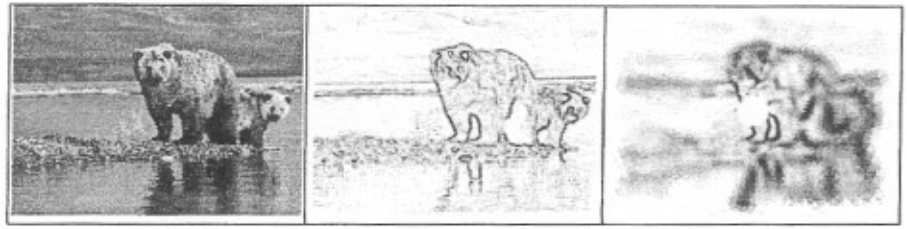

Fig. 2. Foveated edge detection

9 Results

Figure 2 demonstrates our edge detector. We transferred the left image to a logimage and applied the edge operator. The back-projection of the result is shown in the right image, and can be compared with the uniform equivalent (middle).

To demonstrate a spatial mask detector, we set a 5×5 mask that detects "Γ"-shaped corners. The image in Fig. 3(a) 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. 3(b).

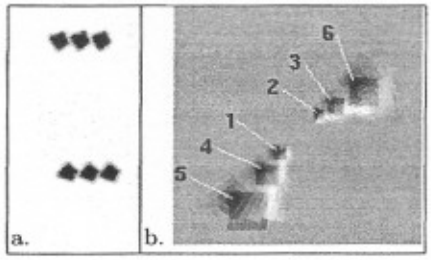

Fig. 3. Corner detection using a foveated spatial mask. Highest peaks designated by their order

The Hough operator is demonstrated on an image with 5 segments that was transferred into logimages using two fixation points (Fig. 4, left). The edge and hough operators were applied (middle) and the 5 highest local maxima were marked. The derived lines are drawn over the edge-logimage (right), they all had similar votes.

For symmetry extraction we created a uniform image with 6 squares. The applied operator when $\lambda = 2$ and 4 are shown in Fig. 5. The two leftmost figures show the resulting logimages, the right figure shows the projection along with

For any 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 $mid(l_a, l_b)$ and add this pair to l's list if $mid(l_a, l_b) = l$. $mid(l_a, l_b)$ 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 σ and add a weight of $\mathcal{D}^*_\sigma$ to that pair in l's list. σ and $\mathcal{D}^*_\sigma$ are defined as –

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

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

λ is a constant that acts as the symmetry radius (σ in [7]). 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 λ 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 α_{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 [7]).

8 Complexity

Suppose that we are given an operator Op_k. We define the difference operator $Op\Delta_{nk}(u, v)$ as $Op_k(u, v\hat{+}n) - \text{Shift}(Op_k(u, v), n)$ where $\hat{+}$ means $+_{\text{mod}v_{\text{Max}}}$. We also define the $\text{Shift}(Op_k(u, v), n)$ of Op_k by n as the operator which is defined by $\left\{\left(w_i, (u_i, v_i\hat{+}n)^{(1)}, \ldots, (u_i, v_i\hat{+}n)^{(k)}\right)\right\}_{i=1..N}$

We say that Op_k has *radial invariance* if for every (u, v), the squared sum of weights in $Op\Delta_{nk}(u, v)$ is $\ll$ from the one of 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_{Max}.

For the Hough operator, it can be shown that a list with length of $O(u_{\text{Max}} + v_{\text{Max}})$ is attached for each l. Since the Hough operator is radial-invariant, the space complexity of the operator is $O(u_{\text{Max}} \cdot (u_{\text{Max}} + v_{\text{Max}}))$.

Similarly, it can be shown for the symmetry operator that the total space complexity is $O(u_{Max} \cdot v_{Max})$. The symmetry operator is radial invariant, thus it can be represented by $O(u_{Max})$ elements and the bound on the preprocessing time can be tightened to $O(u_{Max})$.

The preprocessing time for the mask operators is large (computations for each mask element is proportional to the size of the uniform image). However the resulting operator requires only $O(u_{Max} \cdot v_{Max})$ space.

In our model, a relatively small number of elements in each logpixel's list (< 25 for each translation mask, ≈ 440 for the Hough operator, $\approx 70/295$ for the symmetry with $\lambda = 2/4$) enables almost a frame rate processing speed. Note that since we build feature *maps*, the number and quality of interesting points we may extract from a map does not affect the extraction complexity.

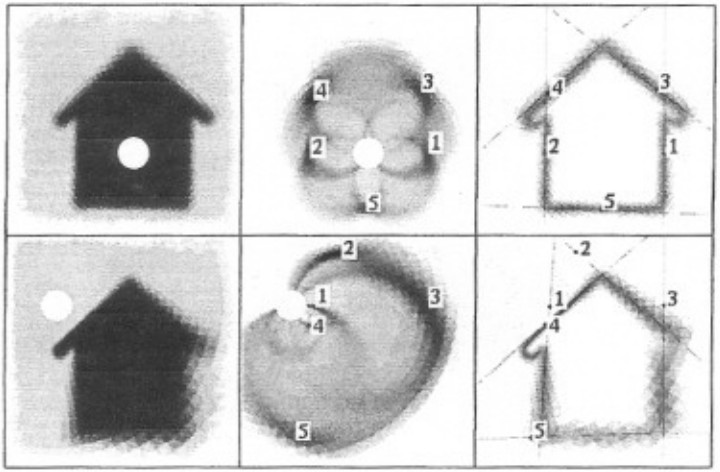

Fig. 4. The foveated Hough operator. See text for details

the source image, for $\lambda = 2$. The symmetry operators indeed detect the corners of the squares (for $\lambda = 2$) or their centers (for $\lambda = 4$).

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

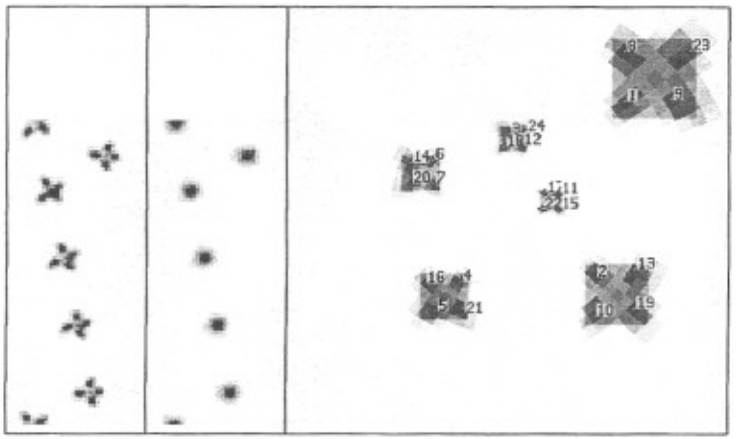

Fig. 5. Foveated corner and center detection using the symmetry operator. Highest peaks designated by their order

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.

References

1. N.R. Carlson. *Physiology of behavior, forth edition.* Allyn and Bacon, Maryland, 1991.
2. N. Drasdo. Receptive field densities of the ganglion cells of the human retina. *Vision Research*, 29:985–988, 1989.
3. R. Jain, R. Kasturi, and B.G. Schunk. *Machine Vision.* McGraw Hill, New York, 1995.
4. M. Langford. *The Step By Step Guide to Photography.* Dorling Kindersley, London, 1978.
5. M.D. Levine. *Vision in Man and Machine.* McGraw-Hill, New York, 1985.
6. S. Polyak. *The Vertebrate Visual System.* The university of Chicago press, 1957.
7. D. Reisfeld, H. Wolfson, and Y. Yeshurun. Context free attentional operators: the generalized symmetry transform. *International Journal of Computer Vision*, 14:119–130, 1995.
8. A. Rojer and E. Schwartz. Design considerations for a space-variant visual sensor with complex logarithmic geometry. In *Proceedings of the 10th IAPR International Conference on Pattern Recognition*, pages 278–285, 1990.
9. B. Sakitt and H. B. Barlow. A model for the economical encoding of the visual image in cerebral cortex. *Biological Cybernetics*, 43:97–108, 1982.
10. E.L. Schwartz. Spatial mapping in the primate sensory projection: Analytic structure and relevance to perception. *Biological Cybernetics*, 25:181–194, 1977.
11. E.L. Schwartz. Computational anatomy and functional architecture of striate cortex: A spatial mapping approach to perceptual coding. *Vision Research*, 20:645–669, 1980.
12. M. Tistarelli and G. Sandini. Dynamic aspects in active vision. *Journal of Computer Vision, Graphics, and Image Processing: Image Understanding*, 56(1):108–129, July 1992.
13. M. Tistarelli and G. Sandini. On the advantages of polar and log-polar mapping for direct estimation of time-to-impact from optical flow. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 15(4):401–410, April 1993.
14. R. Wallace, P. Wen Ong, B. Bederson, and E. Schwartz. Space variant image processing. *International Journal of Computer Vision*, 13(1):71–90, 1994.
15. S.W. Wilson. On the retino-cortial mapping. *International journal of Man-Machine Studies*, 18:361–389, 1983.
16. H. Yamamoto, Y. Yeshurun, and M.D. Levine. An active foveated vision system: Attentional mechanisms and scan path covergence measures. *Journal of Computer Vision and Image Understanding*, 63(1):50–65, January 1996.