Goldberg, editors, *Neural Networks in Robotics*, pages 347–374. Kluwer Academic Publishers, 1992.

[44] Carl Weiman and George Chaikin. Logarithmic spiral grids for image processing and display. *Computer Graphics and Image Processing*, 11, 1979.

[45] Carl F. R. Weiman. 3-d sensing with polar exponential sensor arrays. *SPIE Conference on Digital and Optical Shape Representation and Pattern Recognition*, April 1988.

[46] Carl F. R. Weiman. Exponential sensor array geometry and simulation. In *SPIE Conference on Digital and Optical Shape Representation and Pattern Recognition*, April 1988.

[47] Carl F. R. Weiman. Polar exponential sensor arrays unify iconic and hough space representation. In *SPIE: Intelligent Robots and Computer Vision VIII: Algorithms and Techniques*, pages 832–841, 1989.

[48] Hiroyuki Yamaguchi, Muneo Iida, Akira Tonomo, and Fumio Kishino. Picture quality of a large field visual field display with selective high resolution in foveal vision region. *ITEJ Technical Report*, 14(12), 1990. in Japanese.

[49] Hiroyuki Yamaguchi, Akira Tonomo, and Yukio Kobayashi. Proposal for a large field visual display employing eye movement tracking. *SPIE Optics, Illumination and Image Sensing for machine vision*, 1194, 1989.

[50] Isamu Yorizawa and Makato Kosugi. Depth image reconstruction from motion vectors using cortical image mapping. In *Proceedings of 1991 Electronics, Information and Communications Society Fall National Conference*, 1991. in Japanese.

[30] Azriel Rosenfeld and Avinash C. Kak. *Digital Picture Processing, 2nd Edition.* Academic Press, 1982.

[31] H. Samet. Region representation: quadtrees from boundary codes. *CACM*, 23:163–170, March 1980.

[32] Giulio Sandini, Paolo Dario, and I. Debusschere. Active vision based on space-variant sensing. In *5th International Symposium on Robotics Research*, August 1989.

[33] Gulio Sandini, Paolo Dario, and I. Debusschere. The use of an anthropomorphic visual sensor for motion estimation and and object tracking. In *Proc. OSA Topical Meeting on Image Understanding*, 1989.

[34] Eric L. Schwartz. Spatial mapping in primate sensory perception. *Biological Cybernetics*, 25, 1977.

[35] Akio Shio. An automatic thresholding algorithm based on an illumination-independent contrast measure. In *Conference on Computer Vision and Pattern Recognition*, pages 632–637, 1989.

[36] Michael J. Swain and Markus Stricker Ed. Promising directions in active vision. Technical report, University of Chicago, Chicago, August 1991.

[37] Akira Tonomo, Muneo Iida, and Yukio Kobayashi. A tv camera system which extracts feature points for non-contact eye movement detection. *SPIE Optics, Illumination and Image Sensing for machine vision*, 1194, 1989.

[38] Douglas B. Tweed and Tutis Vilis. The superior colliculus and spatiotemporal translation in the saccadic system. *Neural Networks*, 3:75–86, 1990.

[39] J. van der Spiegel et. al. A foveated retina-like sensor using ccd technology. In C. Mean and M. Ismail, editors, *Analog VLSI Implementations of Neural Networks.* Kluwer, 1989.

[40] Richard S. Wallace, Benjamin B. Bederson, and Eric L. Schwartz. Voice bandwidth visual communication through logmaps: The telecortex. In *Proceedings of the IEEE Workshop on Applications of Computer Vision*, November 1992.

[41] Richard S. Wallace, Ping-Wen Ong, Benjamin B. Bederson, and Eric L. Schwartz. Connectivity graphs for space-variant image processing. Technical Report VAI-1, Vision Applications, Inc., 1991.

[42] Richard S. Wallace, Ping-Wen Ong, Benjamin B. Bederson, and Eric L. Schwartz. Space-variant image processing. Technical Report 256, New York University, Computer Science, Robotics Research, October 1991.

[43] Richard S. Wallace, Ping-Wen Ong, Benjamin B. Bederson, and Eric L. Schwartz. Connectivity graphs for space-variant active vision. In George A. Bekey and Kenneth Y.

[15] Mark S. Franzblau. Log mapped focal driven procedural rendering: an application of the complex log map to ray tracing. Master's thesis, Courant Institute, New York University, 1991.

[16] J. Frazier and R. Nevatia. Detecting moving objects from a moving platform. In *IEEE International Conference on Robotics and Automation*, May 1992.

[17] B. K. P. Horn. *Robot Vision.* MIT Press, 1986.

[18] Hurlbert and T. Poggio. Do computers need attention? *Nature*, 321:651–652, June 1986.

[19] G. Kreider, J. Van der Spiegel, I. Born, C. Claeys, I. Debusschere, G. Sandini, P. Dario, and F. Fantini. A retina like space variant ccd sensor. *SPIE/Charge-Coupled Devices and Solid State Optical Sensors*, 1242, 1990.

[20] Marc Levoy and Ross Whitaker. Gaze-directed volume rendering. *Computer Graphics*, 24(2):217–223, 1991.

[21] R. A. Messner and H. H. Szu. An image processing architecture for real time generation of scale and rotation invariant patterns. *Computer Vision, Graphics, and Image Processing*, 31:50–66, 1985.

[22] A. Montanvert, P. Meer, and A. Rosenfeld. Hierarchical image analysis using irregular tesselations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):307–316, 1991.

[23] C. Narathong, R.M. Inigo, J.F. Doner, and E.S. McVey. Motion-vision architectures. In *Proceedings of Computer Vision and Pattern Recognition*, pages 411–416, 1988.

[24] Ping-Wen Ong. *Image Processing, Pattern Recognition and Attentional Algorithms in a Space-Variant Active Vision System.* PhD thesis, New York University, Courant Institute, 1992.

[25] Ping-Wen Ong, Richard S. Wallace, and Eric L. Schwartz. Space-variant optical character recognition. In *11th International Conference on Pattern Recognition*, August 1992.

[26] Pietro Perona and Jitendra Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, 1990.

[27] Alan Rojer and Eric L. Schwartz. Design of a space-variant sensor having a complex log geometry. In *10th International Conference on Pattern Recognition Volume 2*, 1990.

[28] Alan S. Rojer. *Space-variant computer vision with a complex-logarithmic sensor geometry.* PhD thesis, New York University, Courant Institute, 1989.

[29] A. Rosenfeld. Connectivity in digital pictures. *Journal of the ACM*, 17(1):146–160, 1970.

References

[1] Dana Ballard and Christopher Brown. *Computer Vision.* Prentice-Hall, 1982.

[2] Aijaz A. Baloch and Allen M. Waxman. A neural system for behavioral conditioning of mobile robots. *International Joint Conference on Neural Networks*, pages 723–728, June 1990.

[3] Benjamin B. Bederson. *A miniature space-variant active vision system: Cortex-I.* PhD thesis, New York University, Courant Institute, 1992.

[4] Benjamin B. Bederson. A miniature space-variant active vision system: Cortex-I. Technical Report 266, New York University, Computer Science, Robotics Research, June 1992.

[5] Benjamin B. Bederson, Richard S. Wallace, and Eric L. Schwartz. A miniaturized active vision system. Technical Report 255, New York University, Computer Science, Robotics Research, October 1991.

[6] Benjamin B. Bederson, Richard S. Wallace, and Eric L. Schwartz. Calibration of the spherical pointing motor. In *SPIE Conference on Intelligent Robots and Computer Vision*, November 1992.

[7] Benjamin B. Bederson, Richard S. Wallace, and Eric L. Schwartz. A miniature pan-tilt actuator: The spherical pointing motor. Technical Report 264, Courant Institute, NYU, April 1992.

[8] Benjamin B. Bederson, Richard S. Wallace, and Eric L. Schwartz. A miniaturized active vision system. In *11th International Conference on Pattern Recognition*, August 1992.

[9] Benjamin B. Bederson, Richard S. Wallace, and Eric L. Schwartz. Two miniature pantilt devices. In *IEEE International Conference on Robotics and Automation*, May 1992.

[10] C. Braccini, G. Gambardella, G. Sandini, and V. Tagliasco. A model of the early stages of the human visual system: Functional and topological transformations performed in the peripheral visual field. *Biological Cybernetics*, pages 57–58, 1982.

[11] David Casasent and Demetri Psaltis. Position, rotation, and scale invariant optical correlation. *Applied Optics*, 15(7):1795–1799, July 1976.

[12] G. M. Chaikin and C. F. R. Weiman. Image processing system. U.S. Patent No. 4,267,573, May 1981.

[13] N. Deo. *Graph Theory with Applications to Engineering and Computer Science.* Prentice-Hall, 1974.

[14] Eric L. Schwartz (ed.). Computational neuroscince: Applications of computer graphics and image processing to two and three dimensional modeling of the functional architecture of the visual cortex. *IEEE Computer Graphics and Applications*, 8(4), July 1988.

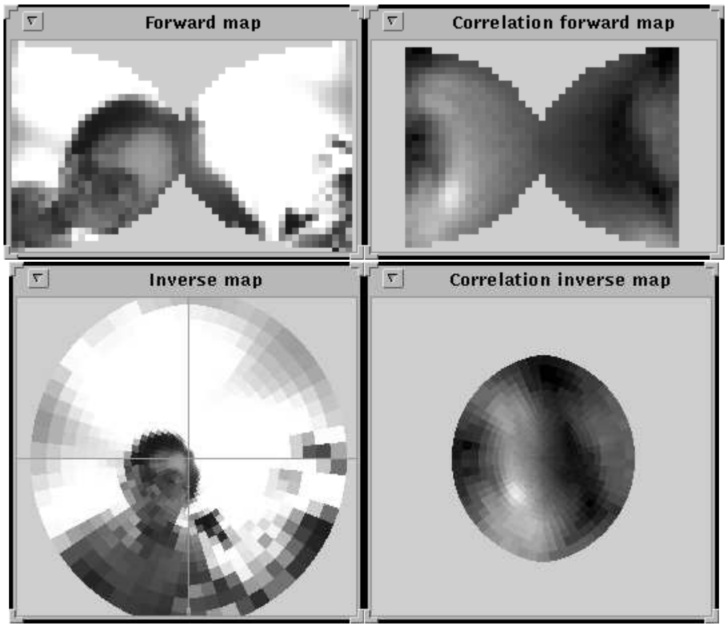


Figure 20: The left column shows the target image in the forward and inverse logmap coordinates. At the right is the result of template matching. The brightest pixel in the result map corresponds to the best matching position.

Transformation graphs are relatives of connectivity graphs that represent the effects of transformations such as translation and rotation in space variant images. Logmap images in particular are known to have elegant mathematical properties with respect to scaling and rotation, but translation of the data from these sensors has until now been problematic. If we have enough memory to represent all translation graphs for a particular sensor having k pixels, then we can compute a translated image in $O(k)$ steps. Our primary application of the translation graph is template matching. We show the results of template matching in a space variant image. The template matching operation is the basis for a target tracking program that requires processing only at the low data rate of the logmap sensor.

Image processing operations defined in the CG are independent of the sensor geometry. We developed a library of image processing routines that work on images from a variety of sensors. The CG image processing library will be useful as we continue to experiment with new sensor geometries, and as we begin to use future VLSI implementations of logmap sensors. [3]

---

[3] The authors would like to thank Dr. Ken Goldberg of the University of Southern California for his valuable comments on an early draft of this paper. We would also like to thank the anonymous referees for their substantive remarks and constructive criticisms. We gratefully thank Jack Judson of Texas Instruments for his technical assistance in the development of our logmap sensor.

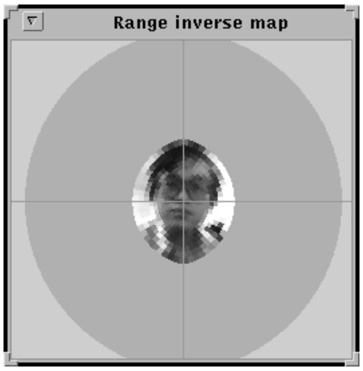


Figure 19: The template image.

7 Conclusion

The logmap sensor presents a special image processing problem because of its irregular pixel geometry. Pixels that are adjacent in the sensor may not be adjacent in an array representation of the sensor image. For example, in the logmap image defined by $\mathbf{w} = \log(\mathbf{z} + \alpha)$ the pixels along the vertical meridian are not connected to all of their neighbors in the array. Conventional image processing functions, defined over arrays, do not produce adequate results when applied to such space variant images. By explicitly representing the neighbor relations in space variant images, the connectivity graph data abstraction allows us to define image processing operations for space variant sensor data.

We have implemented a prototype real-time miniaturized active vision system based on the logmap sensor geometry. Even with the stringent memory limitations (less that 128Kbytes) of our prototype system, we implemented image operations using the CG. These operations became the components of the more complex applications to license plate reading, motion tracking, and actuator calibration using visual sensory feedback

Local image operators such as edge detectors and relaxation operators can be defined easily in the CG. These operations are independent of the sensor geometry. As a side effect, the local operators are defined everywhere in the space variant image, even at the image boundaries. Therefore, there are no special cases in CG local operator definitions for image boundary conditions. In this paper we showed local operators for edge detection and relaxation. The latter operator enabled us to define an image binarization operation for images having nonuniform illumination.

One slightly cumbersome issue is the definition of local operators involving more than one pixel and its immediate neighbors. We have shown that such operators can be defined in the CG, however, and they have the same generality as the immediate local operations. A smoothing operator was presented as an example of an enlarged local operation.

Building on the work of Montanvert et. al [22], we can also define pyramid structures over the CG. In this paper we touched upon the use of such pyramids, and showed how to implement a local binarization operator in a 2-level pyramid.

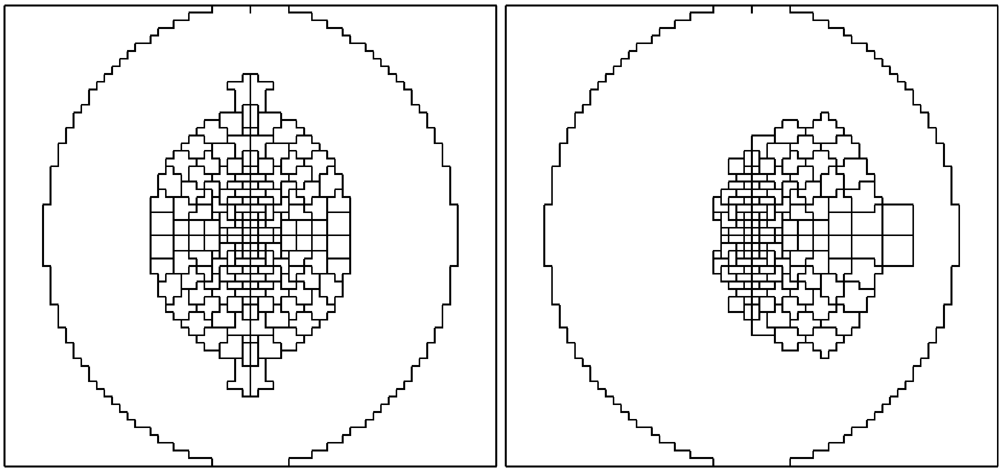


Figure 18: The concept of pixel *coverage* for translation.The left box depicts a masked region to be translated. The right box shows the pixel coverage for a translation along the x axis by 8 TV pixels

When we translate the template image by a vector $T = (\Delta i, \Delta j)$, it overlays another set of logmap pixels. Some of the second set fall completely under the translated template, but it covers others only partially. In order to define the matching error, we need to formalize this concept of *coverage*, illustrated by Figure 18. We define the image graph $G_{MT} = (V_{MT}, \emptyset)$ in the following way. A pixel $q \in V_{MT}$ if and only if its area

$$a(q) = \sum_{p \in V_M \text{ and } (p,q) \in E_T} \kappa_T(p,q).$$

We compute the translated template image using the translation graph:

$$L_{MT}(q) = \frac{1}{a(q)} \sum_{p \in V_M \text{ and } (p,q) \in E_T} \kappa_T(p,q) L_M(p)$$

An area weighted matching value between a translated template image and a data image $L(p)$ is defined by

$$\sum_{q \in V_{MT}} a(q)(L(q) - L_{MT}(q))^2 / \sum_{q \in V_{MT}} a(q)$$

The best matching position T is the one having minimum matching value.

Figure 19 shows a template image displayed in TV coordinates. Figure 20 shows the data image and the results of template matching. Note that we bounded the matching radius in order to get high pixel coverage.

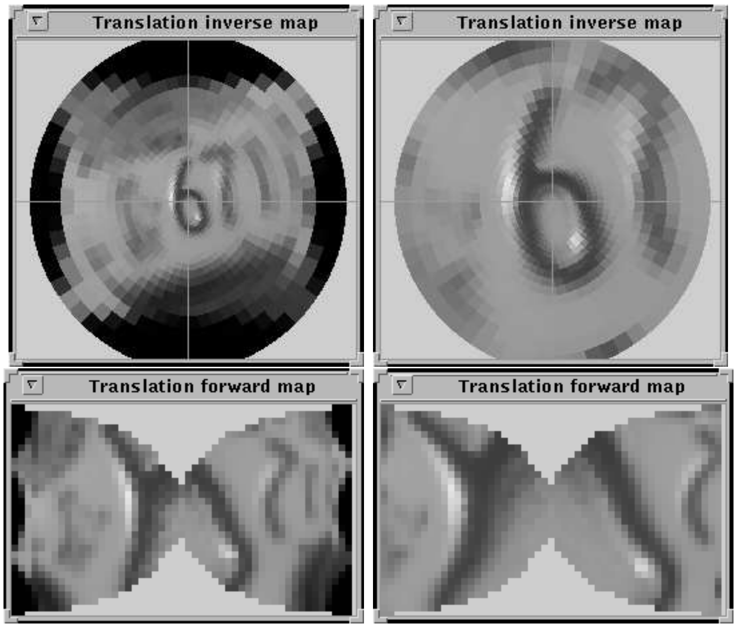

Translation inverse map
Translation inverse map
Translation forward map
Translation forward map

Figure 17: Scaling of logmap images. The left image pair is scaled down and the right image pair is scaled up.

6 Template matching

The translation graph enables us to solve the template matching problem for logmap images. Suppose we have a logmap image template given by the graph $G_M = (V_M, \emptyset)$. Call the logmap pixel values $L_M(p), p \in V_M$. In general, the vertices $V_M \subset V$. The edge set for the template model is empty, because to define the template model we need to consider only the pixels, not relations between them.

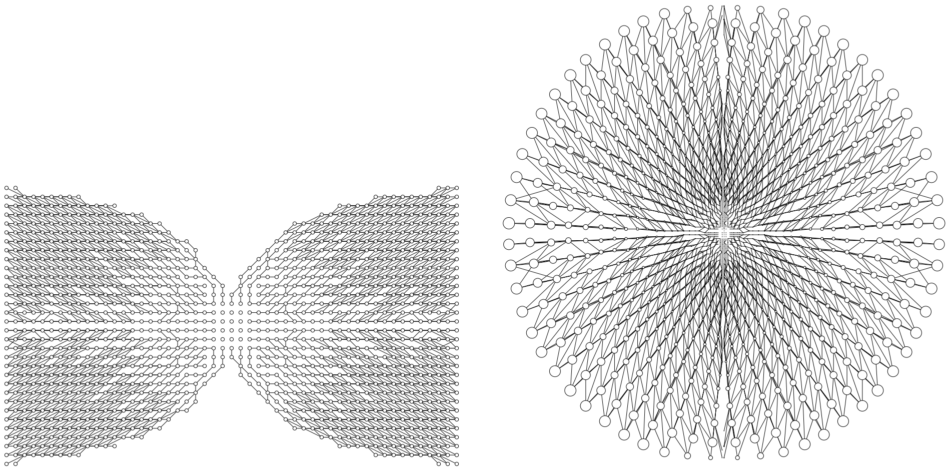


Figure 15: The scale graph shown in forward (left) and inverse (right) coordinates with one element ratio scale change

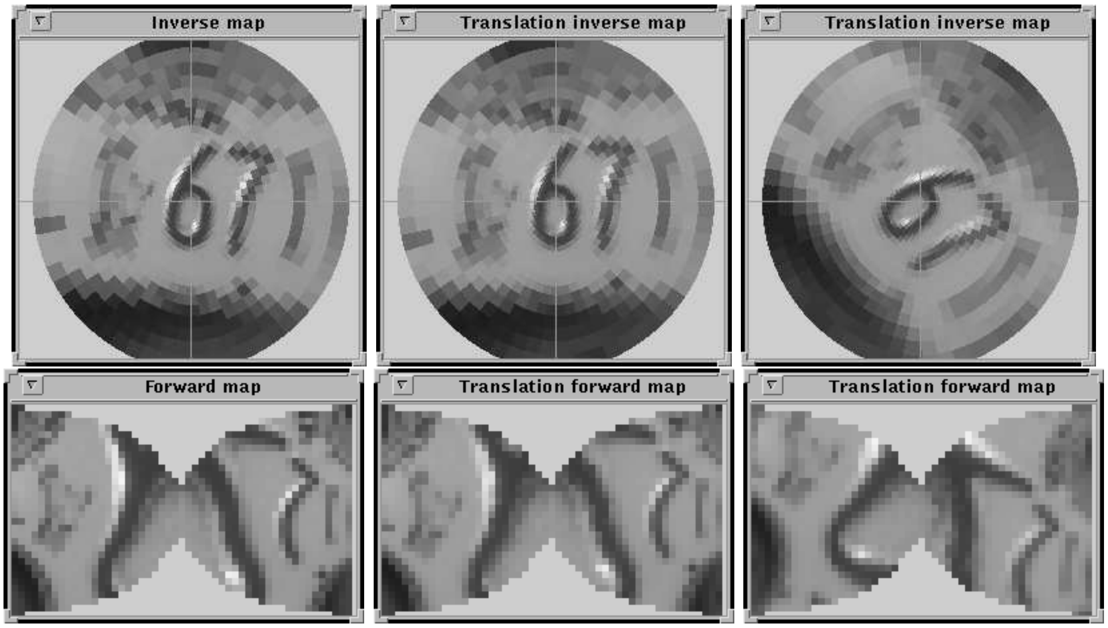

Inverse map
Translation inverse map
Translation inverse map
Forward map
Translation forward map
Translation forward map

Figure 16: Rotation of logmap images. The left image pair is the original logmap image. The center pair is rotated by one element angle. The right pair shows a larger rotation.

These considerations make the transformation graph approach practical despite its quadratic memory cost.

5.2 Rotation and Scaling Graphs

The solution to the rotation and scaling problem for the general space variant sensor geometry is essentially the same as the solution for translation. After analyzing the gray level redistribution by rotating the original image, we can construct a *rotation graph*, defined analogously to the translation graph. To simplify the problem, let us assume that rotations must be integer multiples of the angular sampling period of the logmap sensor. Call the angle between adjacent spokes of the logmap the *element angle*. We consider only rotations that are integer number of element angles. If there are s spokes, we construct only the s rotation graphs for all possible rotations. Figure 14 shows a rotation graph which rotates the image by the one element angle.

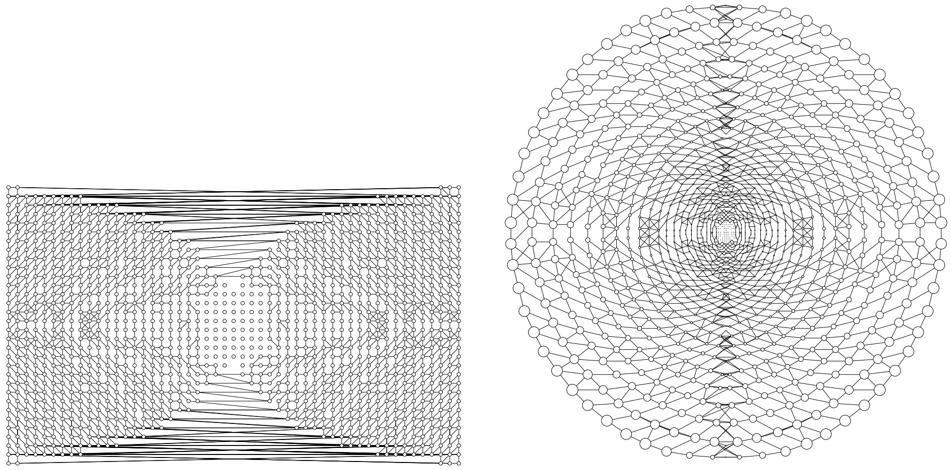


Figure 14: The rotation graph shown in forward (left) and inverse (right) coordinates of a rotation by one element angle.

By a similar analysis, we can construct the *scale graph*. Call the radius ratio of radii in two consecutive rings the element ratio. If the sensor has r rings, we need to compute the r scale graphs for scaling up the image and r scale graphs for scaling down. In figure 15 we show a graph which scales the logmap image by one element ratio. In figures 16 and 17, we show some examples of rotated and scaled logmap images.

The rotation and scaling graphs also contain a lot of redundant structure, and their number is only linear in the number of spokes or rings. But for our logmap sensor, much of the rotation and scaling graphs is redundant. What these representations do is to make explicit the exceptions to the mathematical equivalence of scaling and rotation to image shifting.

$$L_T(q) = \frac{1}{\sum \kappa_T(p,q)} \sum \kappa_T(p,q) L(p) \mid (p,q) \in E_T. \tag{5}$$

Figure 13 shows a plot of the graph G_T for the vector $T = (0,8)$.

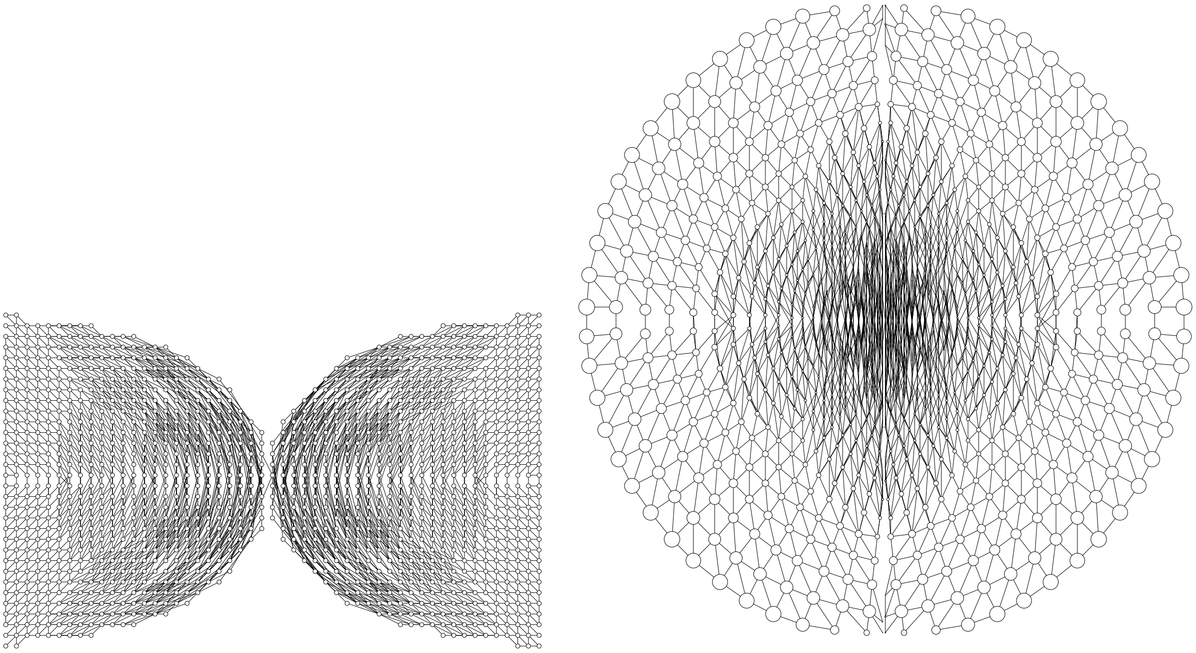


Figure 13: Translation graph for the translation vector $T = (0,8)$.

Any given vertex p in G_T is associated with a set of edges $E_T(p) = \{(p,q) \in E_T\}$. Assuming that $|E_T(p)|$ is bounded above by a constant, then computing a translation by (5) takes $O(k)$ steps. The drawback to the TG is that storing all of them requires a large amount of memory. If we allow translations to every TV pixel (i,j), then we need to store mn graphs of size $O(k)$. Allowing translation only to the k centroids $\boldsymbol{\mu}(p), p \in V$ requires us to store k graphs, giving a total storage requirement of $O(k^2)$ locations.

But in this case it is important to look at the constant factors. The combination of a sensor with thousands of pixels with a computer having millions of memory locations is quite feasible at very low cost. Moreover, the transformation graphs contain a lot of redundant structure, indicating that a memory representation of them could be compressed. Also, many tracking and matching applications require only a small set of cached possible translations.

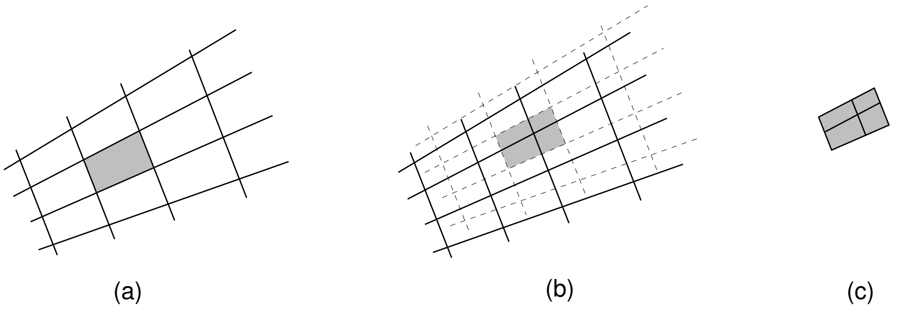


Figure 12: (a) a TV image region corresponding to a logmap pixel. (b) the effect of pixel translation (c) the four pixels whose gray value will be affected by this translation.

5.1 Translation Graph

The transformation graph methods for translation, rotation and scaling are all similar. All three are special cases of more general image mappings, but we limit the discussion to these three for simplicity of exposition. To simplify even further, let us first focus on the problem of image translation.

One approach is to remap the image back to TV coordinates, using the mapping described in section 1.2, then to translate the TV image and then map back to sensor coordinates. The drawback to this method is that it takes time proportional to the size of the TV image. In this section we show how to use a set of graphs to implement translation in time proportional to the number of space variant image pixels, which can be from as much as four orders of magnitude smaller than a TV image having the same field of view[27].

For every translation $T = (\Delta i, \Delta j)$ we construct a translation graph $G_T = (V, E_T)$ from the connectivity graph $G = (V, E)$. In order to translate the space variant image by the offset T we need to compute the redistribution of gray levels in the translated image. Figure 12 shows that the translation of a pixel redistributes its gray value to possibly several other pixels.

Given a translation $T = (\Delta i, \Delta j)$ the edge set E_T contains (p, q) provided

$$\begin{aligned} \exists(i,j) \mid \quad & R(i,j) = u(q) \text{ and } S(i,j) = v(q) \text{ and} \\ & R(i-\Delta i, j-\Delta j) = u(p) \text{ and } S(i-\Delta i, j-\Delta j) = v(p) \end{aligned}$$

Associated with every edge $(p, q) \in E_T$ is a number

$$\begin{aligned} \kappa_T(p,q) = \textstyle\sum_{i,j} 1 \mid \quad & R(i,j) = u(q) \text{ and } S(i,j) = v(q) \text{ and} \\ & R(i-\Delta i, j-\Delta j) = u(p) \text{ and } S(i-\Delta i, j-\Delta j) = v(p) \end{aligned}$$

indicating the number of TV pixels shifted by T from p to q.

Using the translation graph (TG), we compute the translated image by the following expression:

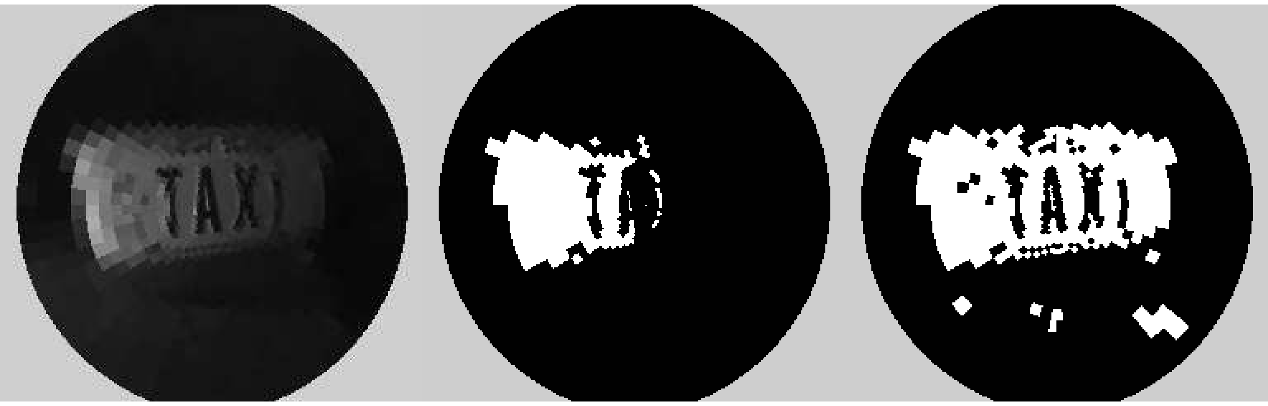


Figure 11: Example of binarization using a 2-level pyramid. The left image depicts a scene containing a license plate under uneven illumination. The middle image shows the result of applying a single global threshold. The right image shows the result of applying an adaptive local binarization procedure.

It propagates reliable threshold values in level 1 first. Then it propagate these values in level 0.

Mark those vertices p at level 1 having reliable thresholds as seed points. Set the gray values $L_0(p)$ equal to a threshold value such as the mean $\mu(P)$. Then, apply the soap bubble algorithm given by equation (4) until convergence. We propagate these values to the seed points at level 0, i.e. $L(q) = L(p)$ if $parent(q) = p$. We assign a constant gray value to the remaining graph vertices at level 0, then again apply algorithm (4) to obtain individual threshold values for every pixel in the graph. Using these thresholds, we binarize the image. The image in Figure 11 is a license plate under uneven lighting conditions. We show the results of applying global and local thresholding to it.

5 Transformation graphs

Log polar images have mathematical properties that make them attractive for certain applications such as optical flow and navigation [11] [12] [16] [21] [26] [28] [27] [32] [33] [34] [39] [44] [46] [45] [47] [50] [44]. For an image sensor having a pixel geometry given by $\mathbf{w} = \log(\mathbf{z})$, image scaling is equivalent to radial shifting, and image rotation is equivalent to annular shifting. These facts alone have motivated much of the work to date on foveated sensors.

But the space variant image *translation* problem has until now remained cumbersome. Also, for a logmap sensor based on defined by $\mathbf{w} = \log(\mathbf{z} + \alpha)$, the shift properties of scaling and rotation are only approximations of the actual effects. In response to these problems, we developed a graph-based method for image transformations. The graph's vertices are pixels and its edges represent the redistribution of gray levels effected by a transformation. For this reason, the graph is called a *transformation graph*.

$$
\begin{array}{ll}
(p, q) \in E(\ell+1) & \text{provided } p \in V(\ell+1) \text{ and } q \in V(\ell+1) \text{ and} \\
 & \exists\ p_0, q_0 \text{ such that} \\
 & \mathit{parent}(p_0) = p \text{ and } \mathit{parent}(q_0) = q \text{ and} \\
 & (p_0, q_0) \in E(\ell).
\end{array}
$$

4.1 Sensor data partitions

A 2-level pyramid defines a partition $P_1, \ldots, P_{|V(1)|}$ of the sensor pixels, where P_h is the set $\{p \mid \mathit{parent}(p) = p_h\}$. We call the region formed by any of these partitions P_h the *local region.* The broad definition of the recursive graph reduction represents the many ways to partition the sensor data. If we want to perform operations on partitions that depend on the number of pixels in them, we partition the sensor into connected regions having approximately the same number of pixels. Another choice is to partition the sensor data into sets of pixels having approximately the same area. The first choice is better when we need a statistically significant set of pixels in order to find, for example, a reliable local threshold value. One method recursively uses the centroid of all the pixels under consideration as a reference point and repeatedly splits the sensor data into four smaller sets, terminating when it reaches a minimum pixel count. This pyramid is similar topologically to the quadtree decomposition [31] for conventional video images.

4.2 Adaptive local binarization

One application of the pyramid is local binarization. To binarize an image, we compute a local threshold value for each local region, based on the method proposed by Shio [35]. He used an illumination-independent contrast measure to verify the reliability of a local threshold values. His reliability index $r(P)$ for a given local region P is defined as

$$r(P) = \frac{\sigma(P)}{\mu(P) - \beta}$$

where $\sigma(P)$ and $\mu(P)$ are the mean and standard deviation of intensity, respectively.

The parameter β depends on the video camera and A/D converter used. To simulate our space variant sensor for this experiment, we used a Sony XC-77RR CCD camera and an Analogic DASM frame grabber. From an experiment similar to the one reported in [35], we measured the value β at very close to 0 for our equipment.

Larger values of $r(P)$ indicate evidence for reliable local thresholds. We use a relaxation method to propagate threshold values from regions having reliable values to the regions having unreliable ones. If there is no reliable threshold value, we resort to a global binarization method, but this situation occurs rarely in real scenes. For those local regions which have reliable threshold values, we use the region mean or the median as the threshold value, both of which perform well. The algorithm described below computes threshold values for every pixel.

The definitions given so far have expressed functions of a pixel and only its immediate neighbors. Sometimes a local operator may map from a larger neighborhood, however. One helpful function in defining such operations is the $\mathcal{N}_2$ function.

$$\mathcal{N}_2(p) = \{b \mid b \in \mathcal{N}(q) \textbf{ and } q \in \mathcal{N}(p) \textbf{ and } b \notin \mathcal{N}(p)\}.$$

The 2 in $\mathcal{N}_2(p)$ denotes the fact that these graph nodes are two edges away from the vertex p. In this way, we say that $\mathcal{N}_1(p) = \mathcal{N}(p)$. We can go on to define $\mathcal{N}_3(p)$, $\mathcal{N}_4(p)$ etc. similarly.

Now we can define a smoothing operator $c(p)$ over an extended neighborhood:

$$c(p) = \frac{1}{|\mathcal{N}_1(p)| + |\mathcal{N}_2(p)| + 1}(L(p) + \sum_{q \in \mathcal{N}_1(p) \cup \mathcal{N}_2(p)} L(q))$$

The local operations defined above are independent of the sensor geometry and its size. In software simulations, we have experimented with a variety of space variant sensors. Because they are all defined by the lookup tables R and S, our program can automatically compute the CG and apply any of the local operations to data from any of the sensors.

4 Pyramid operations

Following the method of Montanvert et al. [22], we can recursively define a pyramid structure, each layer of which is a general CG. This structure gives us a natural way to define partitions for a space variant sensor. A pyramid structure over anirregular tesselation may be visualized and represented by a graph. Each layer of the pyramid is similar to a connectivity graph, but each of the vertices p in the pyramid are connected between levels by a directed edge called *parent*(p).

We call the bottom layer of the pyramid level 0 and call each successive layer $1, 2\ldots$ etc. The parent edges at level ℓ are directed from level ℓ to level $\ell + 1$. We denote the vertex set at level ℓ with $V(\ell)$. The function *parent*(p) has the property that if $q = \mathit{parent}(p)$ and $p \in V(\ell)$ then $q \in V(\ell + 1)$. Moreover, every vertex p has exactly one *parent*(p) except for the vertices at the highest pyramid level, where *parent*(p) is undefined.

Given the graph $G(\ell) = (E(\ell), V(\ell))$ at level ℓ , we define a recursive graph reduction to obtain $G(\ell + 1) = (E(\ell + 1), V(\ell + 1))$ at level $\ell + 1$. The graph $G(0)$ is the same as the CG, except that each vertex $p \in V(0)$ has the new edge relation *parent*. The vertices in successive levels have the property that

$$q \in V(\ell + 1) \text{ iff } \exists p \in V(\ell) \mid \mathit{parent}(p) = q.$$

It follows that $|V(\ell + 1)| \leq |V(\ell)|$, but we usually restrict our attention to the case that $|V(\ell + 1)| < |V(\ell)|$.

To obtain the connectivity (i.e. non-parent) edges we apply the recursive graph reduction procedure given in [22] :

Another simple example is the pseudo-Laplacian $f(p)$, illustrated in Figure 10, given by

$$f(p) = L(p) - \frac{1}{|\mathcal{N}(p)|} \sum_{q \in \mathcal{N}(p)} L(q).$$

Of course, computing the *actual* Laplacian on the sensor array requires transforming expressions like (1) and (2) into analytic functions and solving for the second derivatives.

We can generalize these definitions to take into account other pixel properties such as the relative differences in pixel size $a(p)$, or the pixel mean $\boldsymbol{\mu}(p)$. For example we could have defined

$$f(p) = L(p) - \sum_{q \in \mathcal{N}(p)} \omega(p,q) L(q) / \sum_{q \in \mathcal{N}(p)} \omega(p,q)$$

where $\omega(p,q) = a(q)/|\boldsymbol{\mu}(p) - \boldsymbol{\mu}(q)|$.

A slightly more complicated definition is illustrated by a plane-fitting edge operator. Given the $d = |\mathcal{N}(p)| + 1$ points

$$\{q_1, \ldots q_d\} = \{p\} \cup \mathcal{N}(p)$$

and a vector of their gray values $\mathbf{A} = (L(q_1), \ldots, L(q_d))$, fit a plane $h(i,j) = \mathbf{a} \cdot (i,j,1)^T$ to minimize some error measurement. Applying the least squares error criteria for linear equations [17], we can find a plane by solving for

$$\mathbf{A} = \mathbf{a}\mathbf{B} \text{ where}$$

$$\mathbf{B} = \begin{pmatrix} \boldsymbol{\mu}(q_1) & \ldots & \boldsymbol{\mu}(q_d) \\ 1 & \ldots & 1 \end{pmatrix}$$

The solution of $\mathbf{a}$ for $|\mathcal{N}(p)| \geq 2$ is

$$\mathbf{a} = \mathbf{A}\mathbf{B}^T(\mathbf{B}\mathbf{B}^T)^{-1}$$

If not all points are colinear, $(\mathbf{B}\mathbf{B}^T)^{-1}$ always exists.

Since $\mathbf{M} = \mathbf{B}^T(\mathbf{B}\mathbf{B}^T)^{-1}$ is constant for each pixel p, we precompute and store this matrix $\mathbf{M}(p)$ for every graph vertex p. At run time, we only need to compute $\mathbf{AM}(p)$ for every graph node p.

A simple iterative relaxation procedure, the soap bubble algorithm, is also defined naturally on the CG. If p is a seed point, then $L_{t+1}(p) = L_0(p)$, otherwise

$$L_{t+1}(p) = \frac{1}{1 + |\mathcal{N}(p)|}(L(p) + \sum_{q \in \mathcal{N}(p)} L_t(q)). \qquad (4)$$

3 Local neighborhood operations

Using the connectivity graph, we can define local neighborhood operations independently of the number of neighbors, and therefore without arbitrary boundary effects. Our CG approach generalizes local image processing operations to arbitrary sensor geometries.

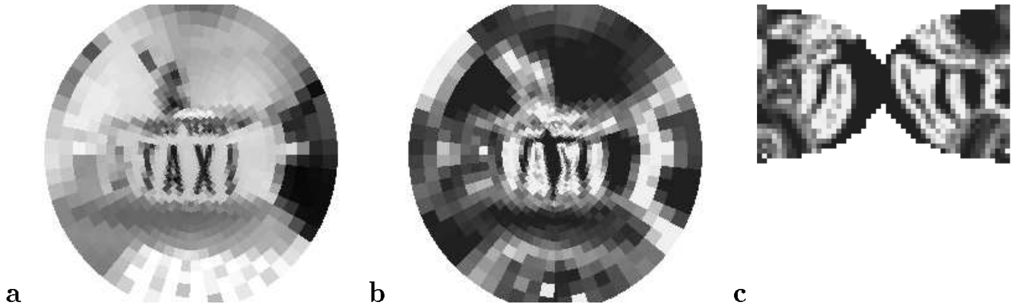


Figure 9: (a) A logmap image having 1376 pixels and (b) the result of a CG local edge magnitude operation. (c) The same edge image image in forward map coordinates.

We can define a simple edge detector, the effect of which Figure 9 illustrates, as

$$e(p) = \frac{1}{|\mathcal{N}(p)|} \sum_{q \in \mathcal{N}(p)} (L(p) - L(q))^2.$$

Note that the definition of $e(p)$ contains no special cases for pixels having different number of neighbors, and thus applies equally to pixels at the boundary of the image and to those in the interior.

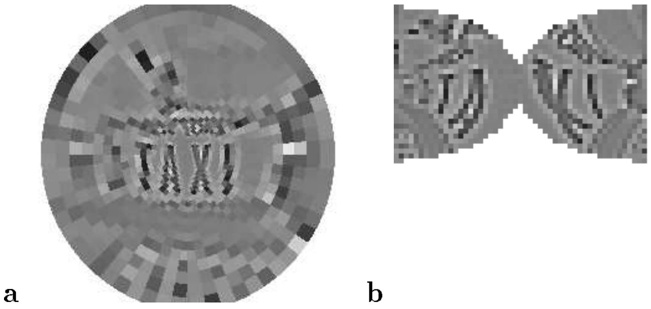


Figure 10: (a) The result of a CG pseudo-Laplacian local operation. (b) The same image image in forward map coordinates.

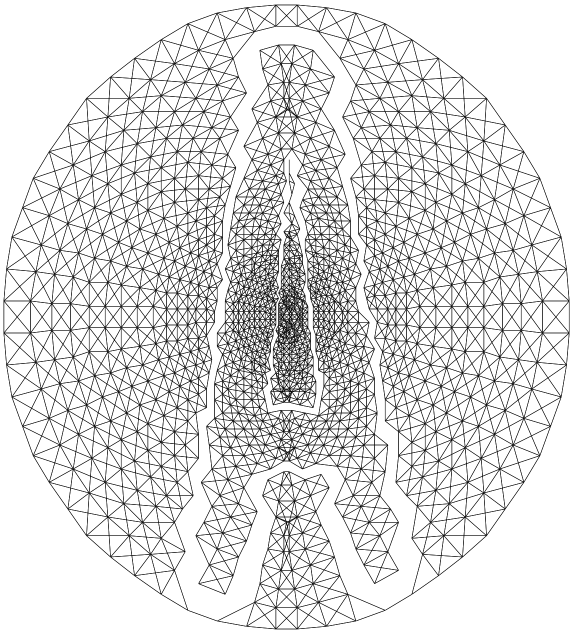


Figure 8: The CG reduces the image connected components problem to its graph theoretic counterpart. Here we show the foreground-background segmentation of an image containing the letter "A".

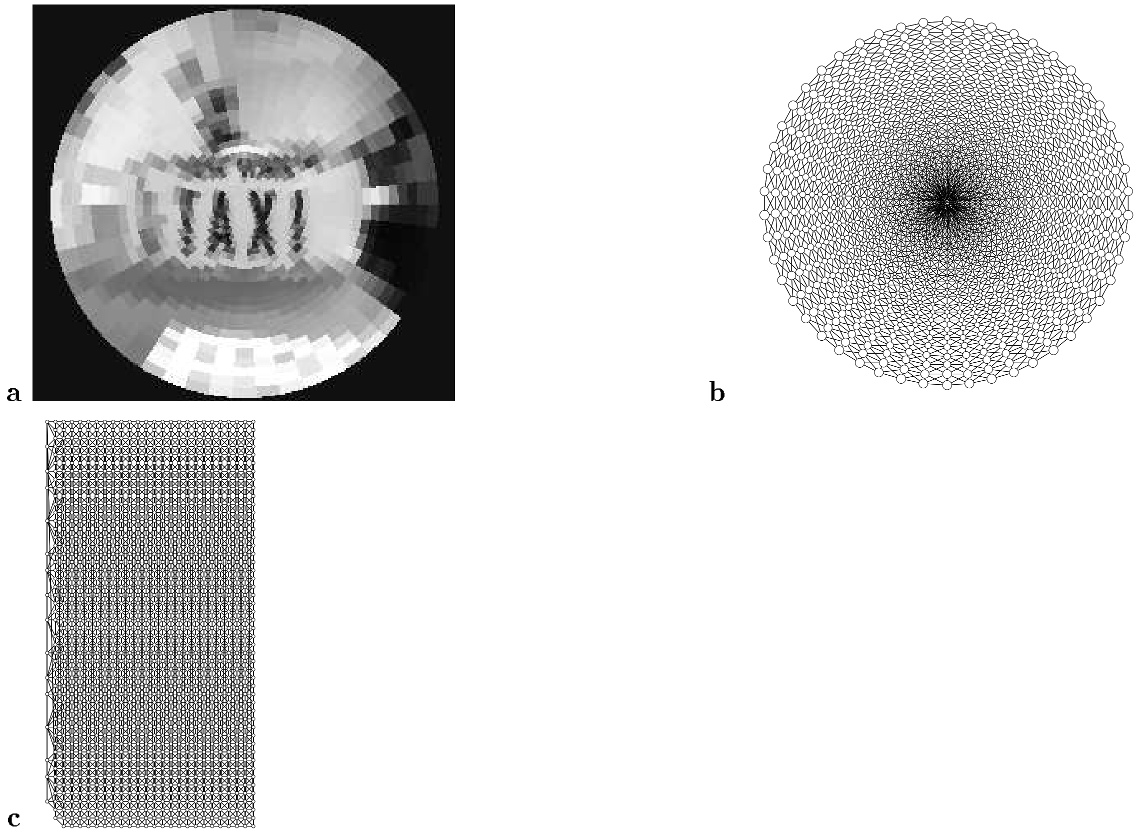


Figure 7: The connectivity graph of a log polar sensor given by the mapping $\mathbf{w} = \log(\mathbf{z})$, where the origin is not in the domain of the mapping. The connectivity graph is plotted in inverse map coordinates (b) and in forward map coordinates (c). For this type of sensor, a horizontal shift in forward coordinates is equivalent to an image scaling, and a vertical shift is equivalent to an image rotation. Note that the CG representation makes explicit the wraparound necessary for image rotation, and also the sparseness of pixels near the origin.

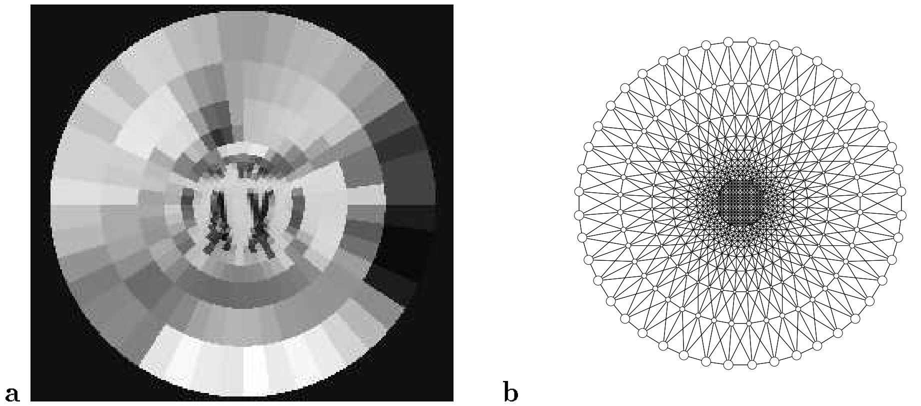


Figure 6: The connectivity graph (b) for a CMOS logmap sensor made by Synaptics, Inc. The center pixels form a rectangular grid, and these are surrounded by annular rings of increasing size. The image (a) shows a simulation of this sensor.

$a(u(p), v(p))$, and the number of neighbors $|\mathcal{N}(p)|$. The pixel centroid $\boldsymbol{\mu}(p)$ is given by

$$\boldsymbol{\mu}(p) = \frac{1}{a(p)} \sum_{i,j} (i,j)^T \; |R(i,j) = u(p) \text{ and } S(i,j) = v(p)$$

We and others have designed and fabricated a variety of space variant sensors [12] [28] [27] [32] [33] [39] . As illustrated in Figures 6 and 7 there exists a CG for each of these architectures. We will use the CG to show, in the next section, how to develop a library of image processing routines that can run on data from any of these sensors, and therefore that the CG is a generic solution to image processing problems under an unconstrained image sensor geometry. But first, we will describe the image processing problem that led us to connectivity graphs in the first place, namely, connected components analysis.

2.2 Connected components

The connected components problem begins with a given predicate F, for example the foreground-background function

$$F(p,q) = \begin{cases} \text{true} & L(p) = L(q) \\ \text{false} & \text{otherwise} \end{cases}$$

true in a binary image when the two pixels p and q are either both black or both white. The algorithm to find all connected components of pixels satisfying F is well-known [1] [17] [30] . In fact we use the familiar graph theoretic algorithm [13]. First we break the CG into subgraphs by removing those edges between pixels where $F(p,q)$ does not hold. Then, by the connectedness and components algorithm for graphs, our program extracts the connected regions. Figure 8 shows the result.

pairs are 8-connected both in the logmap and in the TV coordinate system, but other pairs are connected only in the logmap. We would like the connectivity graph edge set E to include both sets of neighbor relations. Let E_F be the set of edges representing connections between pixels in the forward mapping. An edge $(p, q) \in E_F$ provided $(u(p), v(p))$ is an 8-(4-) neighbor of $(u(q), v(q))$.

For the logmap sensor in Figure 1, E_F does not contain the connections across the vertical meridian. To obtain all the connections in the sensor, we place the *inverse image* adjacency relations in the edge set E_I:

$$(p,q) \in E_I \text{ provided } \begin{array}{l} \exists\ i, j, i_0, j_0 \text{ such that} \\ S(i,j) = u(p), R(i,j) = v(p) \text{ and} \\ S(i_0,j_0) = u(q), R(i_0,j_0) = v(q) \text{ and} \\ (i,j) \text{ is an } 8- \ (4-)\text{neighbor of } (i_0,j_0). \end{array} \tag{3}$$

For the logmap image in Figures 1 and 2 the CG edge set is $E = E_I \cup E_F$.

For an arbitrary pixel geometry sensor illustrated in Figure 4, the construction of the edge set E differs slightly. The forward map for an k pixel sensor of this type is simply a k-element array, but neighbor relations in that array are not necessarily related to connectedness of the sensor pixels. When the sensor mapping is given by lookup tables S and R that do not preserve adjacency in the forward map, like the one illustrated in Figure 4, we set the CG edge set $E = E_I$.

Given the lookup tables S and R, the algorithm to compute the connectivity graph is simple. One vertex $p \in V$ is allocated for each pixel (u, v). The graph edges in E_F are just the $8-$ or $4-$ edges from the forward map image. The graph edges in E_I are computed by scanning the lookup tables $S(i, j)$ and $R(i, j)$ with a 2×2 operator.

The CG is a data abstraction for space variant image data, and as such it may be implemented in a variety of ways, including the computation of neighbor relations on line. When the logmap geometry is given by functions such as (1) and (2), we can transform the them into analytic functions by removing the $\lfloor\cdot\rfloor$ integer transform, and then obtain neighbor relations by searching for pixel indexes satisfying (3). Another possibility is computing all the neighbor relations in advance, then storing them in discrete graph structures. Our most efficient implementation of CG operations uses a hybrid of both approaches. Examination of Figure 5 shows that it is only the boundary pixels that have special cases of connectivity relations, whereas the interior pixels, which cover most of the logmap area, all have the usual 8-connectedness. Therefore at run time a single branch instruction determines whether a pixel is interior or exterior, and for the former it calculates neighbor storage locations by efficient autoincremental addressing. Computations on the boundary of the CG use pre-calculated lists of neighbors to retrieve data from them efficiently.

Other operations we define on the graph node p are the pixel array coordinates $(u(p), v(p))$, the pixel intensity [2] $L(p) = L(u(p), v(p))$, the pixel centroid $\boldsymbol{\mu}(p)$, the pixel area $a(p) =$

---

[2] We have intentionally overloaded the definition of $L()$. The two-argument $L(u, v)$ is pixel brightness in the array representation, but the one-argument $L(p)$ is pixel brightness in the CG representation.

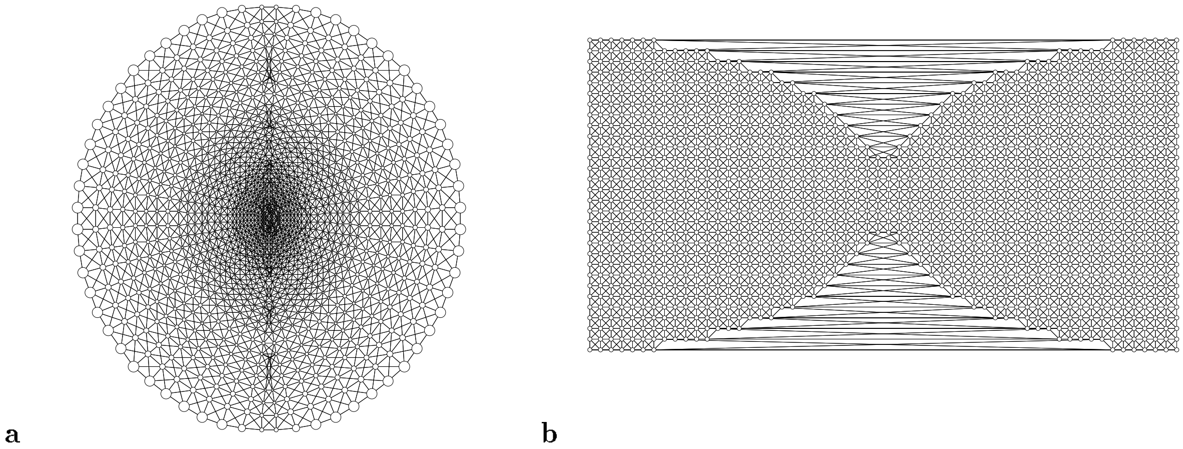


Figure 5: The CG for the sensor depicted in Figure 1. Circles stand for graph vertices and lines stand for graph edges. The plot (a) shows the vertices in logmap array coordinates, and the plot (b) shows the same graph in sensor coordinates. The graph explicitly represents the neighbor relations across the vertical meridian.

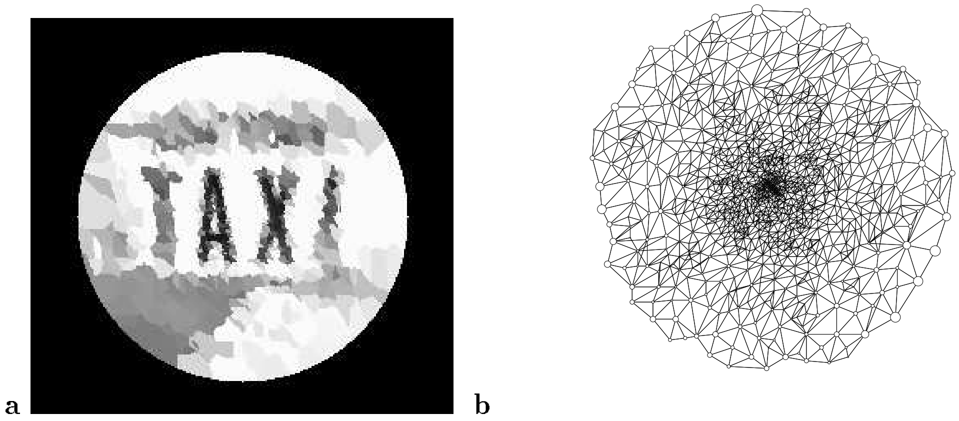


Figure 4: The image from a sensor having an arbitrary pixel geometry (left). The CG for this sensor (right).

pixel architecture, but also in those hybrid sensors composed of a $\log(\mathbf{z})$ architecture combined with a central foveal area having uniform resolution.

More generally, suppose we have an image sensor in which the pixels have no regular pattern, such as the one illustrated in Figure 4 . In this case there is no obvious way to map the pixels into a two dimensional array. Yet if we had such a sensor, we might still be motivated to do image processing with it. In this paper we show that image processing operations can in fact be defined on such images, given a systematic approach to the connectivity problem.

2 The connectivity graph

In this section we give a definition of the CG and then show how to use it to solve the connected components problem in a logmap image.

2.1 Defining the connectivity graph

The CG is a graph $G = (V, E)$ whose vertices V stand for sensor pixels and whose edges E represent the adjacency relations between pixels. Figure 5 illustrates the CG for our logmap sensor geometry. Associated with a vertex p is a logmap pixel address (u, v). Thus we write $(u(p), v(p))$ for a pixel coordinate identified by its graph vertex, or $p = \phi(u, v)$ for a vertex identified by its pixel coordinate. Then, $V = \{p_0, \ldots, p_{k-1}\}$, where $|V| = k$ is both the number of logmap sensor pixels and the number of vertices in V. We use the expression $\mathcal{N}(p)$ to refer to the set of pixels adjacent to p:

$$\mathcal{N}(p) = \{q \mid (p, q) \in E\}.$$

We discussed the example of a logmap generated by the expression $\mathbf{w} = \log(\mathbf{z} + \alpha)$ in the previous section. The lookup table depicted in Figure 2 illustrates that some logmap pixel

$$\rho = |(x + \alpha + \tfrac{1}{2},\ y + \tfrac{1}{2})|$$

so that

$$R_Q(x, y) = \left\lfloor \frac{r}{2} \frac{(\log(\rho) - g_{\min})}{(g_{\max} - g_{\min})} \right\rfloor . \tag{1}$$

We also let

$$\theta = \tan^{-1}(x + \alpha + \tfrac{1}{2},\ y + \tfrac{1}{2})$$

so that

$$S_Q(x, y) = \left\lfloor \frac{s(\theta - \theta_{\min})}{\pi - 2\theta_{\min}} \right\rfloor \tag{2}$$

Where $R_Q(x, y) \geq \frac{r}{2}$ we change the values of R_Q and S_Q to *undefined.* A 4-way symmetry procedure obtains the full frame logmap lookup tables S and R in TV coordinates:

$\forall (i, j) | i \in \{0, \ldots, \frac{m}{2} - 1\}$ **and** $j \in \{0, \ldots, \frac{n}{2} - 1\}$ **and** $R_Q(j, i)$ is defined let

$$\begin{aligned}
S(\tfrac{m}{2} - 1 - i,\ \tfrac{n}{2} - 1 - j) &= S(\tfrac{m}{2} - 1 - i,\ \tfrac{n}{2} + j) &&= S_Q(j,\ i), \\
S(\tfrac{m}{2} + i,\ \tfrac{n}{2} - 1 - j) &= S(\tfrac{m}{2} + i,\ \tfrac{n}{2} + j) &&= s - 1 - S_Q(j,\ i), \\
R(\tfrac{m}{2} - 1 - i,\ \tfrac{n}{2} - 1 - j) &= R(\tfrac{m}{2} + i,\ \tfrac{n}{2} - 1 - j) &&= \tfrac{r}{2} - 1 - R_Q(j,\ i), \\
R(\tfrac{m}{2} - 1 - i,\ \tfrac{n}{2} + j) &= R(\tfrac{m}{2} + i,\ \tfrac{n}{2} + j) &&= \tfrac{r}{2} + R_Q(j,\ i).
\end{aligned}$$

We make the undefined region 4-way symmetric similarly. For more general mappings, we can think of the image in of Figure 1(b) as what the sensor sees and the one in Figure 1(c) as a representation of that image inside computer memory, that is, as an array. Not all cells in the array correspond to sensor pixels: the cells between the "butterfly wings" of the complex log mapping do not correspond to any pixel in the sensor. Moreover, those pixels along the vertical meridian of the sensor are adjacent across the meridian, but they are separated in the memory image. If we try to apply conventional image processing routines to data like that from this sensor, they produce undesirable artifacts along the vertical meridian.

One possible solution to this boundary-effect problem is to duplicate the meridian pixels in the image array. This approach is especially attractive for CCD sensors based on the $\log(\mathbf{z})$ mapping, because timing requirements of CCD imaging result in a cheap readout technique to store sr pixels duplicated in a size $2sr$ memory buffer. But this buffer does not capture connectivity across the sensor's center. Examination of Figure 2 shows that for the more general $\log(\mathbf{z} + \alpha)$, array adjacency alone cannot capture *all* the neighbor relations. Duplicate-pixel representations also pose undue complexities for iterative relaxation procedures like the soap bubble algorithm described below. These connectivity problems arise not only in the $\log(\mathbf{z} + \alpha)$

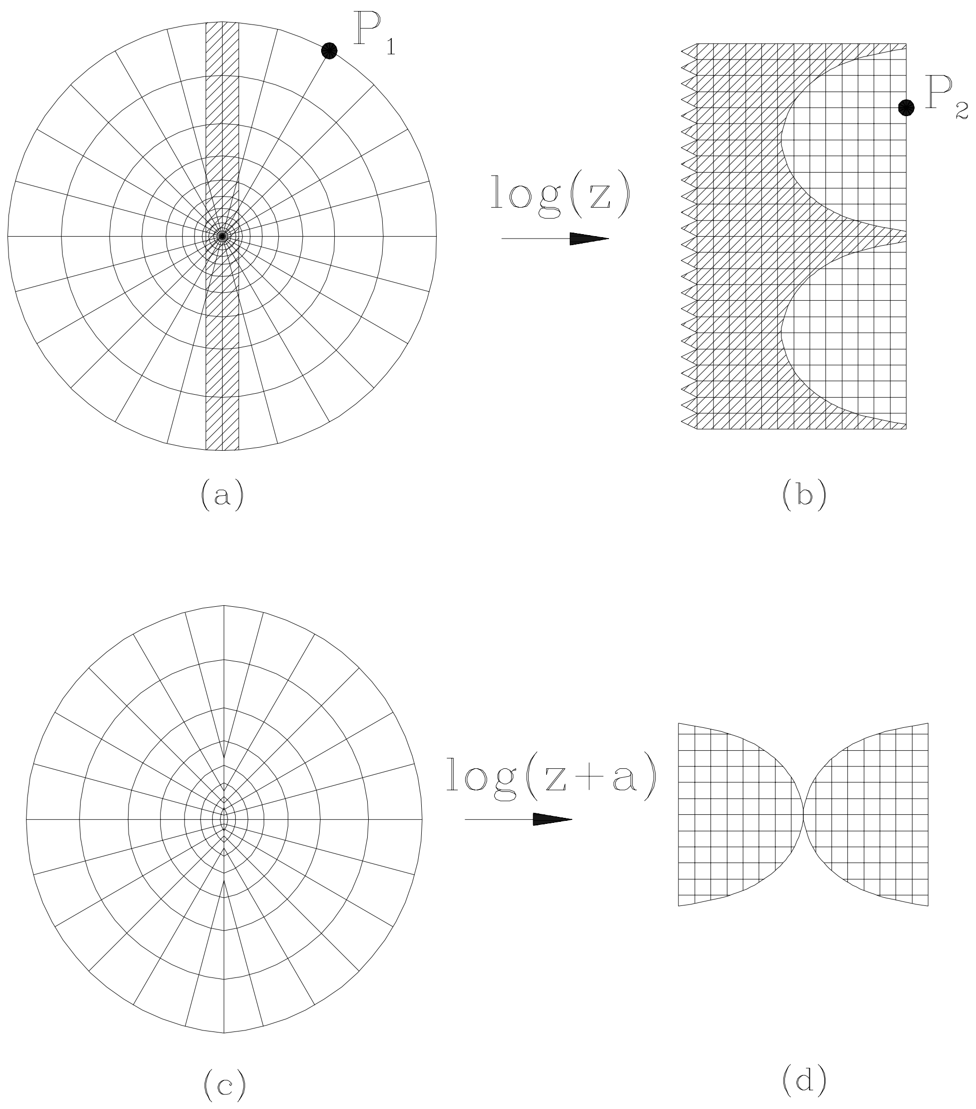


Figure 3: Two types of space variant sensor defined by the mapping $\mathbf{w} = \log(\mathbf{z} + \alpha)$ (see text). The mapping transforms the point P_1 into the point P_2. One way to easily visualize the $\mathbf{w} = \log(\mathbf{z} + \alpha)$ sensor is to think of cutting a strip, shown by the hatched area of (a), out of the middle of a sensor geometry defined by $\mathbf{w} = \log(\mathbf{z})$.

adopt the convention that for those pixels $a(u,v) = 0$. We observe that if k is the number pixels for which $a(u,v) > 0$, then $k \leq sr$. The logmap image is defined where $a(u,v) > 0$ by

$$L(u,v) = \frac{1}{a(u,v)} \sum_{i,j} I(i,j) \mid S(i,j) = u \text{ and } R(i,j) = v,$$

in other words, the average of aggregate sets of TV pixels. The inverse logmap is

$$L^{-1}(i,j) = L(S(i,j), R(i,j)),$$

provided $S(i,j)$ and $R(i,j)$ are defined. The inverse logmap function converts a logmap image to TV coordinates for display.

To see how to construct the lookup tables S and R, we look at the example of a logmap sensor given by the complex mapping $\mathbf{w} = \log(\mathbf{z} + \alpha)$. Figure 3 illustrates the difference between a sensor defined by $\mathbf{w} = \log(\mathbf{z})$ and one defined by $\mathbf{w} = \log(\mathbf{z} + \alpha)$. For simplicity, we restrict our attention to logmap sensors having four-way symmetry, *i.e.* the pixel geometry is reflected about the vertical and horizontal meridians. We construct the 4-way symmetric logmap by first computing the one-quadrant mapping given by a pair of lookup tables $S_Q(x,y)$ and $R_Q(x,y)$, then obtaining S and R by a simple symmetry procedure. The lookup tables $S_Q(x,y)$ and $R_Q(x,y)$ have their indicies $x \in \{0, \ldots, \frac{n}{2} - 1\}$ and $y \in \{0, \ldots, \frac{m}{2} - 1\}$ transposed from matrix coordinates to Cartesian coordinates.

The basic idea behind the sensor geometry is to treat each TV pixel coordinate (x,y) as a complex number $\mathbf{z} = x + \mathrm{i}y$, then to calculate the complex point $\mathbf{w} = \log(\mathbf{z} + \alpha)$, and then to normalize $\mathbf{w}$ to the integer array indexes for S_Q and R_Q.

If we assume that there are no more rows than columns ($m \leq n$), the minimum value of the log function occurs in the pixel $(0,0)$ and its maximum value is where the outermost ring intersects the y axis. We denote the maximum and minimum values as

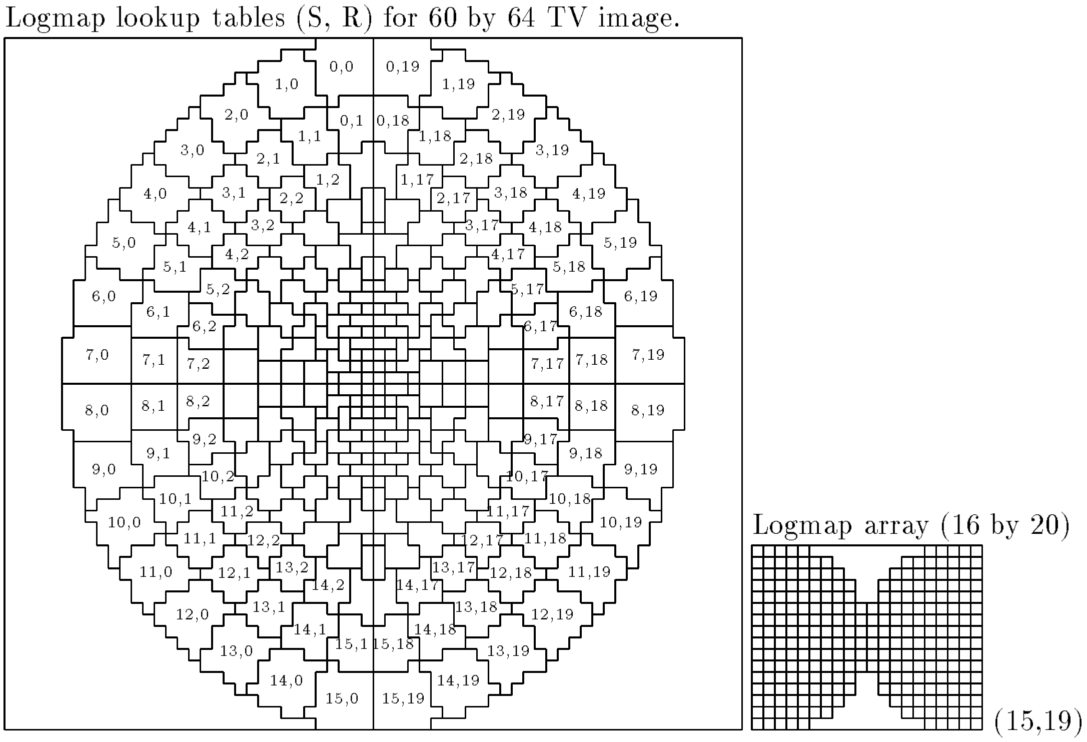
$$\begin{aligned} g_{\max} &= \log|(\alpha + \tfrac{1}{2},\ \tfrac{m}{2} - \tfrac{1}{2})|, \quad \text{and} \\ g_{\min} &= \log|(\alpha + \tfrac{1}{2},\ \tfrac{1}{2})| \end{aligned}$$

respectively. We add the offsets of $\frac{1}{2}$ to compute the lookup table from the *center* of each pixel, rather than the corner. This has the beneficial effect of preventing the axial pixels, and the origin, from being replicated when we compute the 4-way symmetric tables.

The minimum value of the spoke angle occurs on the y axis and is given by

$$\theta_{\min} = \tan^{-1}(\alpha + \tfrac{1}{2},\ \tfrac{m}{2} - \tfrac{1}{2})$$

Then $\forall (x,y) | x \in \{0, \ldots, \frac{n}{2} - 1\}$ **and** $y \in \{0, \ldots, \frac{m}{2} - 1\}$ we let



Logmap lookup tables (S, R) for 60 by 64 TV image.

Logmap array (16 by 20)

0,0
0,19
1,0
1,19
2,0
0,1
0,18
2,19
1,1
1,18
3,0
2,1
2,18
3,19
1,2
1,17
4,0
3,1
2,2
2,17
3,18
4,19
4,1
3,2
3,17
4,18
5,0
5,19
4,2
4,17
5,1
5,18
5,2
5,17
6,0
6,1
6,18
6,19
6,2
6,17
7,0
7,1
7,2
7,17
7,18
7,19
8,0
8,1
8,2
8,17
8,18
8,19
9,2
9,17
9,1
9,18
9,0
9,19
10,2
10,17
10,1
10,18
11,2
11,17
10,0
10,19
11,1
12,2
12,17
11,18
13,2
13,17
11,0
12,1
12,18
11,19
14,2
14,17
13,1
13,18
12,0
12,19
14,1
14,18
15,1
15,18
13,0
13,19
14,0
14,19
15,0
15,19
(15,19)

Figure 2: The lookup table of logmap pixels from TV pixels. The logmap has 16 spokes and 20 rings, but a total of only 272 pixels. The shift factor alpha = 4. (The lookup indexes are only plotted for logmap pixels having area at least 10 TV pixels.) The pixel (0, 1) has 8 neighbors, pixel (1, 1) has 7 neighbors, but pixel (1, 0) has only 4 neighbors. Logmap pixel (0, 0) is connected to (0, 19) across the vertical meridian, but unconnected in the Logmap array.

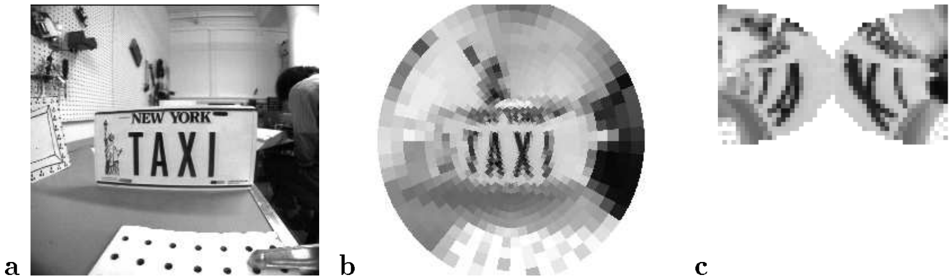


Figure 1: (a) a TV image, $I(i,j)$, (b) The inverse logmap image $L^{-1}(i,j)$, and (c) The forward logmap image $L(u,v)$. We enlarged the printed pixels in the logmap (c). Each logmap pixel requires one byte of memory, and the logmap (c) represents a data reduction, compared with (a), of 30 : 1.

1.2 Space variant image sensors

The CG allows us to write image processing routines that are independent of a particular sensor geometry. To formalize the idea of "sensor-independent" algorithms, we need to make a distinction between three types of images. Figure 1(a) is a conventional raster, video, or TV image. Figure 1(b) is called, for reasons given below, an *inverse* logmap image. The (forward) logmap image appears in Figure 1(c). These three types of image are related by a pair of lookup tables S and R, which map TV image coordinates to logmap array coordinates. If we obtain the logmap image from a conventional video sensor, as in our miniaturized system [8], then the lookup tables S and R tell us which aggregate set of TV pixels to average in order to form a logmap pixel.

But it is worth emphasizing that even if we had a VLSI logmap sensor, we would like to *display* the logmap images in an inverse logmap format, like the one in Figure 1 (b). Therefore for all conventional raster displays, and for firmware logmap readouts, we need to think of a logmap in relation to a TV screen or a a TV image sensor. Moreover, the TV pixel representation of the logmap sensor geometry is the basis for an algorithm, introduced below, to construct the CG for an arbitrary sensor geometry.

We define the logmap formally as a mapping from a TV raster image $I(i,j)$ where $i \in \{0,\ldots,m-1\}$ and $j \in \{0,\ldots,n-1\}$ to a logmap array $L(u,v)$, with $u \in \{0,...,s-1\}$ and $v \in \{0,...,r-1\}$. The mapping is specified by two lookup tables $S(i,j)$ and $R(i,j)$, whose names are chosen as mnemonics for "spoke" and "ring" logmap indexes (see Figure 2) Let $a(u,v)$ be the area (in TV pixels) of a logmap pixel (u,v).

$$a(u,v) = \sum_{i,j} 1 \mid S(i,j) = u \text{ and } R(i,j) = v.$$

Logmap pixels (u,v) for which $\not\exists (i,j) \mid S(i,j) = u$ and $R(i,j) = v$ are undefined and we

1.1 Background

Since the size and cost of a machine vision system, and the motors which drive it if it is active, are scaled by the numbers of pixels it must process, space variant active vision provides the possibility of radical reductions in system size and cost. We have built a 32 frames per second (fps) real-time computer vision hardware system based on the logmap pixel geometry [5] [8] [3] [7] [4]. The choice of the logmap had two direct architectural implications. First, there is a tremendous reduction in the sensor data rate. The low data rate enables real-time logmap image processing routines to run on modest microprocessors, and allows inter-processor image data communication on low capacity channels. The low data rate also enables our consumer video phone to transmit logmap images at 4 fps over ordinary voice telephone lines [40]. The second major impact of the logmap results from the fact that the peripheral pixels are in effect low pass filters. Threfore, we can allow the periphery to be blurry, with the consequence the logmap sensor utilizes a smaller lens than a video sensor, which reduces the size and cost of the camera and its actuator.

With the sensor mounted on a very low cost camera pointing actuator, the Spherical Pointing Motor [9] [6], the total system cost was two orders of magnitudes lower than a comparable system based on conventional video sensors and more elaborate actuators.

We want to state clearly that our system *emulates* a logmap sensor with an embedded DSP and a conventional CCD. Specifically, an AD 2101 DSP reads pixel values digitized from the 192×165 Texas Instruments CCD chip TC211, and averages CCD pixels to form a 1376 pixel logmap. But the DSP is fast enough to run the readout at 32 fps, so the emulated sensor is, in effect, a sensor. The sensor emulator contained only commodity integrated circuits, which eliminated the need for costly custom VLSI.

Constructed at very low cost, systems of this type may open up new niches for machine vision, and can only be built, at present, using space variant sensing to limit the size and cost of the motors, memory, CPUs, and interprocessor communications. For these reasons, space variant sensors in general, and logarithmic sensors in particular, have begun to attract attention in the machine vision community [2] [11] [12] [16] [18] [19] [23] [21] [26] [28] [27] [32] [33] [39] [36] [44] [46] [45] [47] [50]. The sensor geometry motivating much of the work in this paper is given by Rojer [28] [27] .

The basis of our approach to space variant image processing is the use of a CG to represent the neighbor relations between pixels in space variant images. Rosenfeld was the first to apply the topological notion of connectivity to image processing [29]. The standard image processing texts (e.g. [30] vol. 2 p. 208, [17] p. 67ff) discuss the connectivity paradox for digital images, namely that the Jordan curve theorem does not hold when the foreground and background are both four or eight connected. These texts also mention connectivity under a hexagonal pixel geometry. Rojer [28] [27] was apparently the first to use a connectivity graph to implement local image processing operations in a space variant image. The application of graphs to represent arbitrary neighbor relations for region segmentation is also discussed in the standard texts (e.g. [1] p. 159ff). Recently, Montanvert et al. [22] have shown how to define an image pyramid recursively from an irregular tesselation.

1 Introduction

We are accustomed to thinking of an image as a rectangular grid of rectangular pixels. In such an image, connectivity and adjacency are simply defined in terms of the four or eight neighbors of each pixel. But in biological vision systems [10] [34] [38] as well as *space variant* artificial image sensors, based on CCD [19] [33] [39] , MOS [8] [50], and firmware [7] implementations, there exists a pattern of photodetectors having spatially changing size, shape and connectivity across the image array. One motivation for the study space variant sensing derives from its prominent role in the architecture of the human visual system, and so space variant sensors are sometimes called *foveated* or *retinal.* Their elegant mathematical properties for certain visual computations also motivates the development of space variant sensors, and so they are sometimes called *log polar*, *log spiral*, or *logmap* sensors. Except where we note explicitly, we use the term *logmap* to refer to a visual sensor whose outstanding characteristic is that a wide angle workspace may be covered, with the capability for high resolution sensing at the center of the array, incurring a small pixel sensing and processing burden proportional only to the logarithm of the size of the workspace [28] [27] [14] . The special relationship of logmap sensors to the human visual system is a constant of nature analogous to the frame rate constants in motion pictures and TV, and some researchers have exploited this property to and build multiresoultion head-tracking viewer-centered displays [37] [49] [48] and low-cost computer graphic displays based on the logmap [15] [20]. We have applied the logmap architecture to a very low cost moving car license plate identification and tracking system [24] [25] and to a prototype consumer video telephone [40].

A fundamental problem with logmap sensors arises from their varying connectivity across the sensor plane. Pixels which are neighbors in the sensor are not necessarily neighbors once a computer reads the data into an array, making it difficult or impossible to apply conventional image array operations. Some sensors integrate a central uniform resolution region inside an annulus of logarithmic structured pixels, leading the problem of how to combine image data from the two grids. This paper develops a data abstraction called the connectivity graph (CG), which represents explicitly the connectedness relations in arbitrary space variant images (see also [41] [42] [43]). Using the graph, we define a variety of *sensor-independent* image processing operations. The connected components problem for space variant images then reduces to its graph theory counterpart. Local image operations are defined as a function of a pixel and its neighbors, which turns out to have the additional effect of eliminating special cases for boundary conditions. We discuss typical local operations such as edge detection, smoothing, and relaxation. Building on the work reported by Montanvert et al. [22], we define a pyramid structure for space variant images and apply it to a local binarization operation. Finally, we introduce a class of variations of the CG which we call *transformation graphs*, and which represent the effects of image transformations such as translations, rotations and scalings. We demonstrate an implementation of template-matching using the transformation graphs.

Abstract

This paper describes a graph-based approach to image processing, intended for use with images obtained from sensors having space variant sampling grids. The connectivity graph (CG) is presented as a fundamental framework for posing image operations in any kind of space variant sensor. Partially motivated by the observation that human vision is strongly space variant, a number of research groups have been experimenting with space variant sensors. Such systems cover wide solid angles yet maintain high acuity in their central regions. Implementation of space variant systems pose at least two outstanding problems. First, such a system must be active, in order to utilize its high acuity region; second, there are significant image processing problems introduced by the non-uniform pixel size, shape and connectivity. Familiar image processing operations such as connected components, convolution, template matching, and even image translation, take on new and different forms when defined on space variant images. The present paper provides a general method for space variant image processing, based on a connectivity graph which represents the neighbor-relations in an arbitrarily structured sensor. We illustrate this approach with the following applications: (1) Connected components is reduced to its graph theoretic counterpart. We illustrate this on a logmap sensor, which possesses a difficult topology due to the branch cut associated with the complex logarithm function. (2) We show how to write local image operators in the connectivity graph that are independent of the sensor geometry. (3) We relate the connectivity graph to pyramids over irregular tessalations, and implement a local binarization operator in a 2-level pyramid. (4) Finally, we expand the connectivity graph into a structure we call a transformation graph, which represents the effects of geometric transformations in space variant image sensors. Using the transformation graph, we define an efficient algorithm for matching in the logmap images and solve the template matching problem for space variant images. Because of the very small number of pixels typical of logarithmic structured space variant arrays, the connectivity graph approach to image processing is suitable for real-time implementation, and provides a generic solution to a wide range of image processing applications with space variant sensors.

Space Variant Image Processing[1]

Richard S. Wallace

Courant Institute of
Mathematical Sciences
New York University
New York, NY

Ping-Wen Ong

AT&T Bell Laboratories
Middletown, NJ

Benjamin B. Bederson

Bell Communications Research
Morristown, NJ

Eric L. Schwartz

Department of
Cognitive and Neural Systems
Boston University
Boston, MA

---

[1] *This research was supported by DARPA/ONR #N00014-90-C-0049 and AFOSR Life Sciences #88-0275. Please address all correspondence to Richard S. Wallace, NYU Robotics Research Laboratory, 715 Broadway 12th Floor, New York, NY 10003. This report is copyright (c) 1993 by the authors. This report is a revised draft of a report published as New York University Courant Institute of Mathematical Sciences Computer Science Technical Report (No. 589 and Robotics Report No. 256), October, 1991. Last revised February 15, 1993.*