[15] J. Webb, L. Hamey and I-C Wu (1989) "An architecture independent programming language for low-level vision" Computer Vision, Graphics and Image Processing, Volume 48, 1989, p. 246-264.

[16] Vision Applications, Inc. (1991) *The Lunchbox Active Vision System*, technical specification.

[17] B. Bederson, P. Ong, R. Wallace and E. Schwartz (1992), *Space-variant active vision*, forthcoming.

[18] M. Swain, ed. *Final Report of the Active Vision Workshop*, University of Chicago, August, 1991.

[19] E. Schwartz (1977) "Spatial mapping in primate sensory projection", Biological Cybernetics, vol. 25: pp. 181-194

[20] (1979) C. Weiman and G. Chaikin, "Logarithmic spiral grids for image processing and display." Computer Graphics and Image Processing, vol. 11, pp. 197-226.

[21] (1988) E. Schwartz, B. Merker, E. Wolfson, and A. Shaw. "Computational neuroscience: Applications of computer graphics and image processing to the two and three dimensional modeling of the functional architecture of visual cortex. IEEE Comp. Graphics and App. vol 8, pp.13-28.

[22] (1988) K. Gremban, C. Thorpe and T. Kanade. "Geometric camera calibration", Proceedings of IEEE Robotics and Automation conference, 1988.

[23] (1981) Martins, Burke and Kelley. "Camera models based on data from two calibration planes", *Computer Graphics and Image Processing*, 17. 1981.

References

[1] G. M. Chaikin and C. F. R. Weiman, *Image Processing System*, U.S. Patent No. 4,267,573.

[2] G. Sandini, F. Bosero, F. Bottino, and A. Ceccherini. "The use of an anthropomorphic visual sensor for motion estimation and object tracking", in Proceedings of *Image Understanding and Machine Vision*, June, 1989.

[3] G. Sandini and P. Dario, "Active vision based on space-variant sensing", International Symposium on Robotics Research, August, 1989.

[4] J. van der Spiegel, G. Kreider, C. Claeys, I. Debusschere, G. Sandini, P. Dario, F. Fantini, P. Belluti, and G. Soncini "A foveated retina-like sensor using CCD technology" in *Analog Implementations of Neural Networks* eds. C. Mead and M. Ismail, Kluwer Publishing, 1989.

[5] A. Montanvert, P. Meer and A. Rosenfeld (1991), "Hierarchical image analysis using irregular tesselations" *IEEE Transactions on Pattern Analysis and Machine Intelligence* vol. 13 No. 4, pp. 307-316.

[6] A. S. Rojer and E. L. Schwartz, (1990) Design considerations for a space-variant sensor with complex-logarithmic geometry, *10th International Conference on Pattern Recognition*, Atlantic City, NJ, pp. 278-85.

[7] A. S. Rojer (1989) *Space-variant computer vision with a complex-logarithmic sensor geometry*, Ph.D. Thesis, New York University.

[8] A. Rosenfeld and A. C. Kak (1982), *Digital Picture Processing*, 2nd ed. 2 vols. Academic Press.

[9] A. Rosenfeld (1970), "Connectivity in digital pictures" JACM vol. 17 No. 1, pp 146-160, January.

[10] D. H. Ballard and C. M. Brown (1982), *Computer Vision*, Prentice-Hall.

[11] B. K. P. Horn (1986) *Robot Vision*, MIT Press.

[12] N. Deo (1974) *Graph theory with applications to engineering and computer science*, Prentice-Hall.

[13] A. Shio (1989) "An automatic thresholding algorithm based on an illumination-independent contrast measure" International conference on computer vision and pattern recognition, pp. 632-637.

[14] J. Webb (1991) Adapt: Global image processing with the split and merge model, Carnegie Mellon Computer Science Technical Report CMU-CS-91-129.

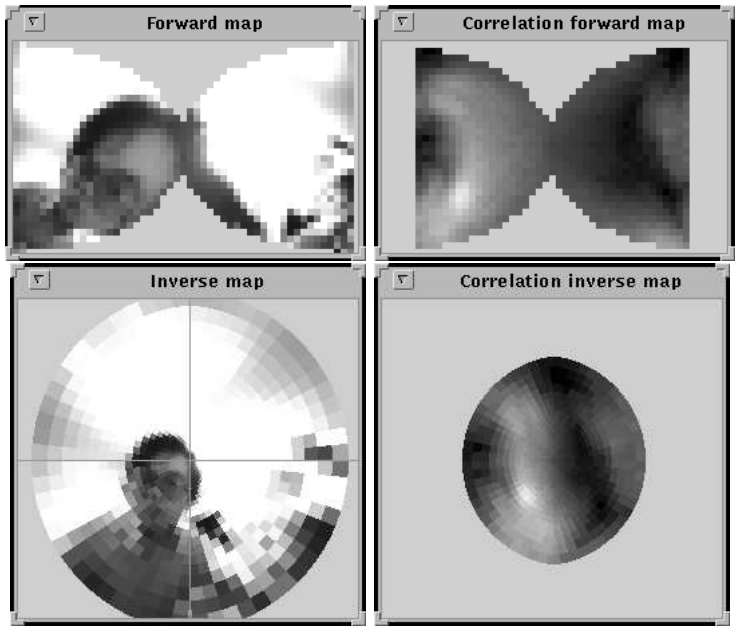

Figure 14: The left column shows the target image in the forward and inverse logmap coordinates. At the right is the result of template matching. The brightest pixel in the result map corresponds to the best matching position.

suggested an approximate algorithm that is more economical with respect to memory usage, and computes an approximate translation T in $O(n \log(T))$ 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 can be combined with the results of pantilt calibration to obtain a real-time target tracking system that requires processing only at the low data rate of the logmap sensor.

Image operations defined in the connectivity graph are independent of the sensor geometry. Therefore, we are developing a library of image processing routines that work on an endless variety of sensors. This library will continue to be very useful as we continue to experiment with new sensor geometries, and as we begin to use future VLSI implementations of these sensors. [2]

[2]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.

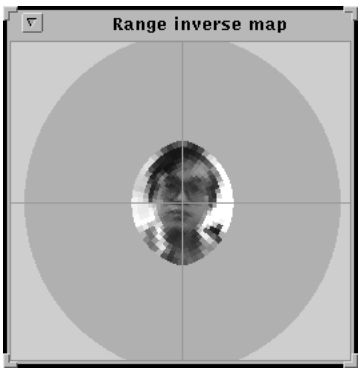

Figure 13: The template image.

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 structure allows us to define image processing operations for space-variant sensor data.

We have implemented a prototype miniaturized active vision system using a logmap image sensor. In order to demonstrate image processing in this system, we needed to use the connectivity graph to define image operations. Even with the stringent memory limitations (less that 128Kbytes) of our prototype system, we succeeded in implementing operations such as connected components analysis using the connectivity graph.

Local image operators such as edge detectors and relaxation operators can be defined easily in the connectivity graph. 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 connectivity graph 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 solve a sensor calibration problem relating the logmap sensor to a robotic pan-tilt actuator.

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 connectivity graph, 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 [5], we can also define pyramid structures over connectivity graph. 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. More work remains to be done in the area of pyramids over space-variant images, however.

Translation graphs are connectivity graphs that represent the effects of translation in space-variant images. logmap images in particular are known to have excellent 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 n pixels, then we can compute a translated image in $O(n)$ steps. We have also

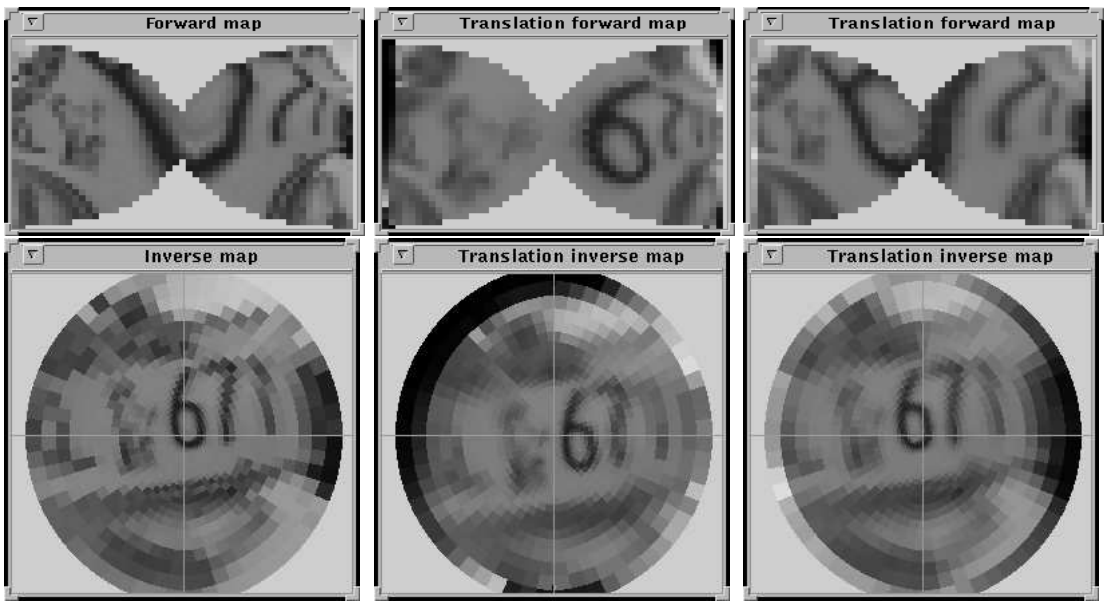

Figure 12: Sample results for translated logmap image. The first row shows the forward maps and the second row shows the corresponding inverse maps. The left one is the original grayscale image. The middle one shows the result of translating by (16,16) and right one shows the result of translating by (-9,1).

template G_M we define the "cover" of the translated template as those vertices V_C such that $q \in V_C$ iff $(p,q) \in E_T$ and $p \in G_M$. We compute the translation T for the pixels in V_C using an instance of formula (16):

$$L_C(q) = \frac{1}{\sum \kappa_T(p,q)} \sum \kappa_T(p,q) L_M(p) \mid (p,q) \in E_M. \tag{17}$$

The matching value between a translated template image and a source image $L(p)$ is

$$\sum_{p \in V_C} (L(p) - L_C(p))^2$$

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

Figure 13 shows a template image in the inverse map. Figure 14 shows the test image on which we do the template matching and the result of the matching.

8 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 the pixels along the vertical meridian are

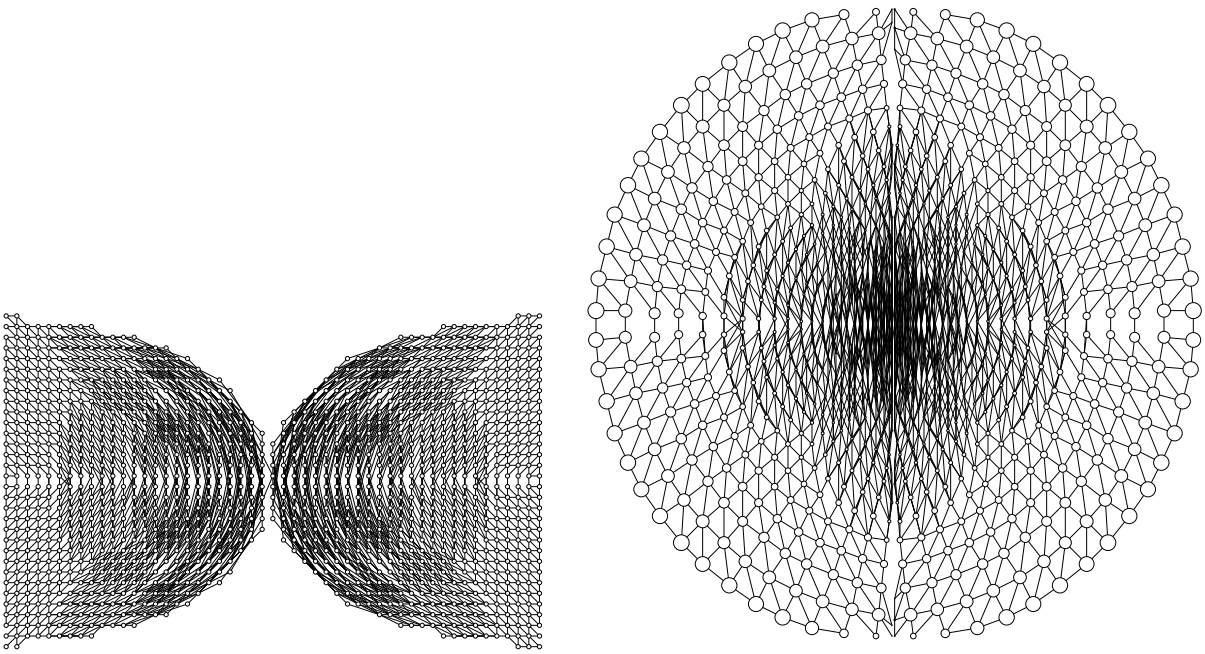

Figure 11: Translation graph for the translation vector $T = (0, 8)$.

steps, if $d_i \cdot d_j \neq 0$. If $d_i \cdot d_j = 0$, we only need to translate the image by

$$\log_2(\max\{|d_i|, |d_j|\})$$

steps. If the number of the edges in the translation graph is $O(n)$, the computation of one translation is order $O(n)$. Therefore, the total time complexity for computing a translation by vector (d_i, d_j) is $O(n \log(|d_i| + |d_j|))$. The space complexity is $O(n \log(c + d))$. There are two disadvantages to this approximation. First, the time needed to compute the translation is not constant. It depends on the translation vector. Second, the incremental translation blurs the image due to aliasing. In Figure 12, we show two translation examples.

7 Template Matching

Using the translation method discussed in the previous section, we can define template matching for a space-variant sensor. Define a *template* G_M to be a subgraph of a translation graph $G_T = (V_T, E_T)$. The subgraph $G_M = (V_M, E_M)$ such that $V_M \subset V_T$ and $E_M = E_T$. Given a

every edge $(p,q) \in E_T$ is a value $\kappa_T(p,q)$ indicating the number of TV pixels in $M(p)$ shifted by T from $M(p)$ to $M(q)$.

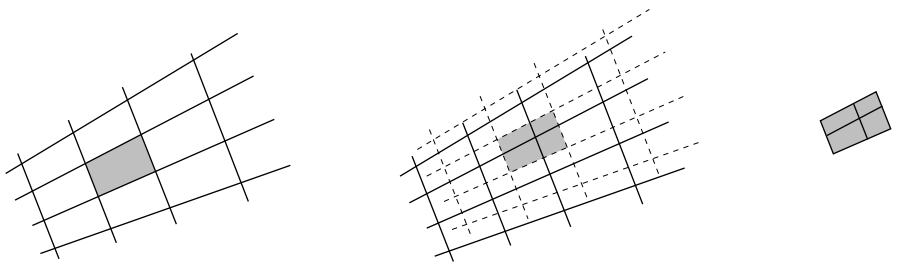

Figure 10: (a) a region $M(p)$ correponding to a logmap pixel p (b) the result of translation (c) the four pixels, q_0, q_1, q_2, q_3, whose gray value will be affected by this translation.

Using the translation graph, we can compute the translated image by the following expression:

$$L_T(q) = \frac{1}{\sum \kappa_T(p,q)} \sum \kappa_T(p,q) L(p) \mid (p,q) \in V_T. \tag{16}$$

Any given vertex p in G_T is associated with a set of edges $E_T(p) = \{(p,q) | (p,q) \in E_T\}$. Assuming that $E_T(p)$ is bounded above by a constant, then computing a translation by (16) takes $O(n)$ steps. The drawback to the translation graph 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 cd graphs of size $O(n)$. Allowing translation only to the n centroids $\boldsymbol{\mu}(p), p \in V$ requires us to store n graphs, giving a total storage requirement of $O(n^2)$ locations.

In Figure 11, we show a plot of graph G_T for the vector $T = (0,8)$.

6.2 Efficient Approximation Method

This section describes an appoximate translation algorithm that requires less memory than the $O(n^2)$ locations needed by the set of all translation graphs.

Suppose we have the translation graph for translating image by

$$(2^b,0),(2^b,2^b),(0,2^b),(-2^b,2^b),(-2^b,0),(-2^b,-2^b),(0,-2^b),(2^b,-2^b)$$

for all $b \in \{0,\ldots,\log_2(\max\{c,d\})\}$ and c,d are the dimensions of the TV image.

By consecutively applying these translations, we can translate the image to any location. For example, if we want to translate the image to location (9,3), we can apply the translation sequence (2,2), (1,1), (4,0), (2,0) to achieve the desired translation. By using this incremental translation method, we can translate the image to location (d_i, d_j) by

$$\log_2(\min\{|d_i|,|d_j|\}) + \log_2(\max\{|d_i|,|d_j|\} - \min\{|d_i|,|d_j|\})$$

a constant gray value to the remaining graph vertices at level 0, then apply the algorithm (11) to obtain individual threshold values for every pixel in the graph. Using these thresholds, we binarize the image.

The image in Figure 9 is a licence plate under uneven lighting conditions. We show the results of applying global and local thresholding to it.

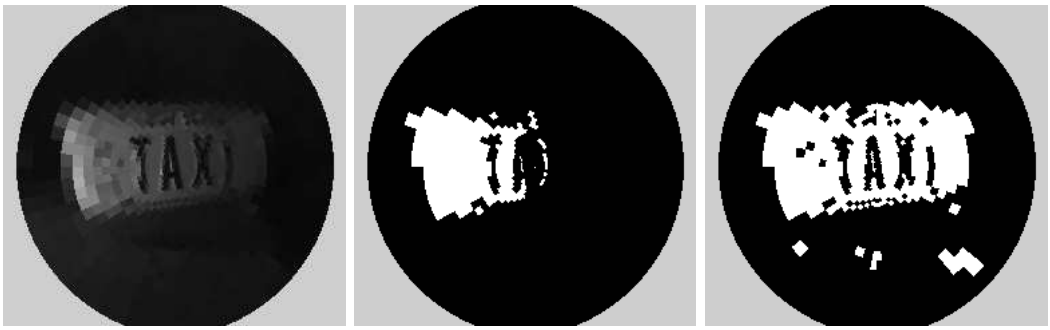

Figure 9: Example of binarization using a 2-level pyramid. On the left is the original gray scale scene: a license plate under uneven illumination. In the middle is the result of applying global threshoding. The right image shows the result of applying our local binarization.

6 Translation graphs

Log polar images have the well-known properties that image scaling is equivalent to radial shifting, and image rotation is equivalent to annular shifting [20][19]. But image translation has until now remained a cumbersome problem for space-variant sensors. One solution is to remap the image back to TV coordinates, using the mapping descibed in section 2, then translate the TV image and map back to sensor coordinates. The drawback to this approach 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 one to four orders of magnitude smaller [6].

6.1 Analysis of Translation

For every translation $T = (d_i, d_j)$ 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 an offset $T = (d_i, d_j)$ we need to compute the redistribution of gray levels in the translated image. For any given pixel $p = \phi(u, v)$ there is a region $M(p)$ in the inverse image whose pixels map to (u, v):

$$\forall (i, j) \in M(p), R(i, j) = u \text{ and } S(i, j) = v.$$

If we translate all the TV pixels in $M(p)$ by T, those pixels will be mapped to k other sensor pixels $\{q_0, \ldots, q_{k-1}\}$. The edge set E_T contains the directed edges (p, q_i). Associated with

5 Pyramid operations

Following the method of Montanvert et al. [5], we can recursively define a pyramid structure, each layer of which is a general connectivity graph. This structure gives us a natural way to define partitions for a space variant sensor.

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 $\ell+1$. First obtain $V(\ell+1) \subset V(\ell)$. Then,

$$\begin{aligned} e = (p,q) \in E(\ell+1) \text{ provided} \quad & p \in V(\ell+1) \text{ and } q \in V(\ell+1) \text{ and} \\ & \exists\ p_0, q_0 \text{ such that} \\ & \mathit{parent}(p_0) = p \text{ and } \mathit{parent}(q_0) = q \text{ and} \\ & (p_0, q_0) \in E(\ell). \end{aligned} \tag{15}$$

Similar to nodes at level 0, the neighbors of a node p at level ℓ are $\mathcal{N}(p) = \{q |\ (p,q) \in E(\ell)\}$.

A 2-level pyramid defines a partition $P_1, \ldots, P_{|V(1)|}$ for the sensor, where P_i is the set $\{p \mid \mathit{parent}(p) = p_i\}$. We call the region formed by any of these partitions P_i the *local region*. We fix one pixel $p_{ic} \in P_i$ to be the *seed pixel* for the region P_i.

A variety of methods can be used to partition the sensor data, but it is desireable to partition it into connected regions having approximately the same number of pixels. Our method recursively splits the sensor data into smaller sets, until a minimum size is reached.

One application of the pyramid is local binarization. To binarize a real image, we compute a local threshold value for each local region. Shio [13] used an illumination-independent contrast measure to verify the reliability of a local threshold values. His reliability index $r(P_i)$ for a given local region P_i is defined as

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

where $\sigma(P_i)$ and $\mu(P_i)$ 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 [13], we find the value for β is very close to 0 for our equipment.

The larger the value of $r(P_i)$, the more reliable the local threshold value is. So we call a threshold reliable when $r(P_i)$ is large. 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 can choose either the mean or the median as the threshold value. In our experiments, both mean and median performed well. The following algorithm computes threshold values for every pixel.

Make the seed points those points p_i at level 1 having reliable thresholds. Set the gray values $L_0(p_i)$ equal to a threshold such as the mean $\mu(P_i)$. Then, apply the iterative interpolation algorithm given by expression (11) for t iterations until convergence. We propagate the interpolated values to the seed points at level 0, i.e. $L(p_{ic}) = L(p)$ if $\mathit{parent}(p_{ci}) = p$. We assign

vision system [16]. In that case the motor commands take the form of relative currents. In the description of calibration that follows, however, we simplify the description by stating only the position-control case.

A TV camera can be calibrated reliably by imaging a target array and measuring the pixel position of each element [22] [23]. With this method, the precise position of each target element must be known. But because our camera is mounted on an actuator, we need to image just a single dot on a blank background. The dot can be a dark blob on a white background, or a light, or any easily identifiable symbol or mark. The only contraint for reliability of the procedure is that the dot can be identified uniquely anywhere in the sensor image. The space-variant geometry of the sensor means that a small dot may cover many pixels when it is near the center, but must be large enough to cover at least one pixel in the periphery.

One reliable way to find the calibration dot uses the connectivity graph. If the position of the dot is estimated to be the pixel p, then our program searches from p recursively, first checking the pixel p itself, then its neighbors $\mathcal{N}_1(p)$, then the pixels in $\mathcal{N}_2(p)$ and so on, until it finds the dot. We take the first dark blob found in the search to be the dot. The position of the dot is defined as the centroid of the blob.

The calibration begins with an estimate command table $\boldsymbol{\tau}_{est}(p)$. Let p_c be the pixel closest to the focal axis. The estimate can be obtained in a variety of ways, but we use a fully automatic prodedure. First, center the dot at p_c. Set a variable $\boldsymbol{\tau}_a$ to zero. Then, execute a series of small motor moves until the dot disappears from the field of view, accumulating the motor commands in $\boldsymbol{\tau}_a$. Just before the dot disappears, it crosses a pixel p_e whose centroid is $\boldsymbol{\mu}(p_e)$. If a command $\boldsymbol{\tau}(p)$ moves a point imaged at p to the center p_c, then the inverse motor command vector $-\boldsymbol{\tau}(p)$ moves the point back from p_c to p. So we initialize the motor command for p_e to $\boldsymbol{\tau}(p_e) = -\boldsymbol{\tau}_a$.

The table is initialized by computing the distance of each pixel p from p_c and setting the estimated motor command to

$$\boldsymbol{\tau}_{est}(p) = \frac{|\boldsymbol{\mu}(p) - \boldsymbol{\mu}(p_c)|}{|\boldsymbol{\mu}(p_e) - \boldsymbol{\mu}(p_c)|}\boldsymbol{\tau}(p_e). \tag{14}$$

The main loop of the calibration procedure repeats the following steps: For each sensor pixel p, (1) center the dot, (2) move actuator by $-\boldsymbol{\tau}_{est}(p)$, (3) using p as the estimated new dot position, find the actual position p_a of the dot, and (4) set $\boldsymbol{\tau}(p_a) = \boldsymbol{\tau}_{est}(p)$. Since it not necessarily true that $p_a = p$, not all the vertices in the graph are filled by the main loop.

The final step is to fill the remaining "holes" in the calibration table by applying the simple relaxation operator of equation (11). The seed points are those points where the calibration table is filled by the main procedure, all other points are interpolated by the soap bubble procedure. If necessary, these interpolated values can be fed back as a set of new estimates, and the main loop repeated. A pyramid technique that improves the interpolation will be presented in the next section.

A simple relaxation procedure, the soap bubble algorithm, is also defined naturally on the connectivity graph:

If p is a seed point, then $L_{t+1}(p) = L(p)$, otherwise

$$L_{t+1}(p) = \frac{1}{1 + |\mathcal{N}(p)|}(L(p) + \sum_{q \in \mathcal{N}(p)} L_t(q)). \tag{11}$$

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 operation helpful in definining such operations is the $\mathcal{N}_2$ function.

$$\mathcal{N}_2(p) = \{r \mid r \in \mathcal{N}(q) \textbf{ and } q \in \mathcal{N}(p) \textbf{ and } r \notin \mathcal{N}(p)\}. \tag{12}$$

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. similary.

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

$$s(p) = \frac{1}{|\mathcal{N}_1(p)| + |\mathcal{N}_2(p)| + 1}(L(p) + \sum_{q \in \mathcal{N}_1(p) \cup \mathcal{N}_2(p)} L(q).) \tag{13}$$

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 connectivity graph and apply any of the local operations on any of the sensors.

4.2 Pantilt calibration

The calibration of a space-variant sensor to a robotic pantilt mechanism provides a simple example of applying the connectivity graph to a real problem. If the actuator rotates exactly around the optic center of the camera, then the calibration problem is to find a motor command vector $\boldsymbol{\tau}(p)$ that tells, for each pixel p, how much to move each motor axis so that the object seen at pixel p becomes centered.

In general a pantilt platform will not necessarily rotate the sensor around its optic center. In this case there is a translational component to the camera motion, so the calibration depends on the distance δ of an object as well as the pixel index p. The calibration procedure described here is the simpler case, however, which is approximately true for our sensors and actuators. We assume that $\boldsymbol{\tau}(p)$ is independent of δ.

We have developed this calibration procedure for two pantilt devices. The first actuator, manufactured by Klinger Scientific, is a 2-axis pan-tilt device based on a conventional motor-on-a-motor design. Its controller is connected to a host computer through a serial interface. The host directs the actuator with high-level commands, such as "move through X degrees of rotation". Lower-level motor control is hidden in the Klinger controller, therefore the calibration problem is simply to find the motor commands $\boldsymbol{\tau}(p)$ in the form of degrees. The second actuator is the Spherical Pointing Motor attached to our prototype miniature active

Our connectivity graph approach generalizes local image processing operations to arbitrary sensor geometries. The two together lead to a programming abstraction for local image processing which is independent of both sensor geometry and processing architecture.

4.1 Definitions

In the connectivity graph $G = (V, E)$, a pixel is represented by a graph vertex $p \in V$. Let the set of neighbors of p be denoted by

$$\mathcal{N}(p) = \{q \mid (p, q) \in E\}, \tag{7}$$

We can define a simple edge detector as

$$e(p) = \frac{1}{|\mathcal{N}(p)|} \sum_{q \in \mathcal{N}(p)} (L(p) - L(q))^2. \tag{8}$$

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.

Another simple example is the Laplacian $\ell(p)$ given by

$$\ell(p) = |\mathcal{N}(p)| \; L(p) \; - \sum_{q \in \mathcal{N}(p)} L(q) \tag{9}$$

These definitions can be generalized to take into account the relative differences in pixel size $a(p)$, or the pixel mean $\boldsymbol{\mu}(p)$. A slightly more complicated definition is illustrated by a plane-fitting edge operator. Given $m = |\mathcal{N}(p)| + 1$ points

$$\{r_0, \ldots r_{m-1}\} = \{p\} \cup \mathcal{N}(p) \tag{10}$$

and their gray values $L(r_k)$, we 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 [11], we can find a plane by solving for

$$\mathbf{A} = \mathbf{aB} \quad \text{where } \mathbf{A} = (L(r_0), \ldots, L(r_{m-1})), \; \mathbf{B} = \begin{pmatrix} \boldsymbol{\mu}(r_0) & \cdots & \boldsymbol{\mu}(r_{m-1}) \\ 1 & \cdots & 1 \end{pmatrix}$$

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 $\mathbf{A}\mathbf{M}(p)$ for every graph node p.

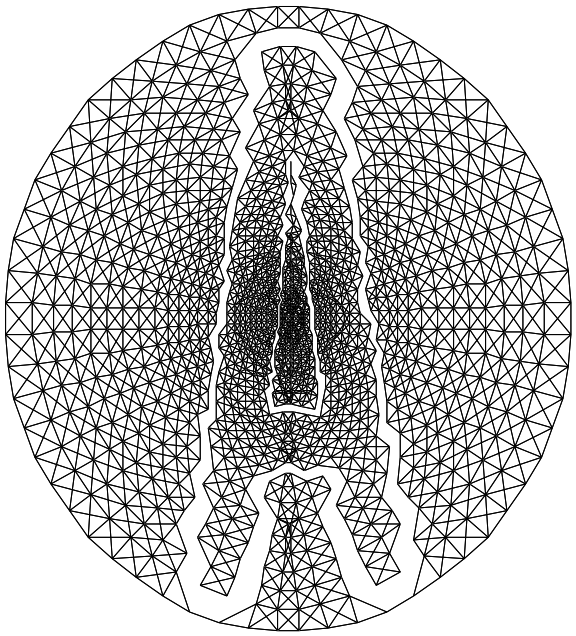

Figure 8: The connectivity graph reduces the image connected components problem to its graph theory conterpart. Here we show the foreground-background segmentation of an image containing the letter "A".

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 [10][11][8]. In fact we use the familiar graph theoretic algorithm [12]. First we break the graph 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.

4 Local neighborhood operations

Webb et al. [15][14] have developed a language, called *Apply*, to write architecture-independent specifications for local and global operations on rectangular images [15]. Compilers facilitate transfer of image processing software between different machines and architectures, particularly parallel processors. Existing image processing compilers assume a rectangular image. The code for a local operator consists of a function $f(\mathcal{S})$ from a small rectangular window $\mathcal{S}$ to a quantity $y = f(\mathcal{S})$. Care must be taken to specify how the function changes at near the image boundary, where the window $\mathcal{S}$ may extend beyond the border of the image. Using the connectivity graph, we can define local neighborhood operations independently of the number of neighbors, and therefore without arbitrary boundary effects.

The *Apply* language generalizes local image processing for arbitrary processor archictures.

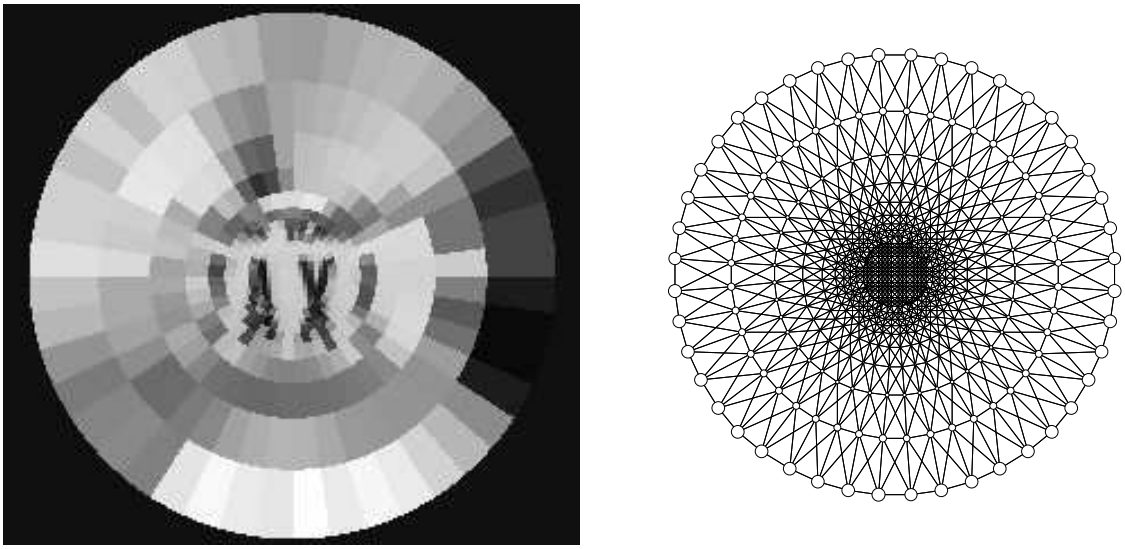

Figure 6: The connectivity graph (right) for a CMOS logmap sensor made by Synaptics, Inc. The center pixels form a rectangular grid, and these are surrounded by annular rings of increasing size. The image (left) shows a simulation of this sensor.

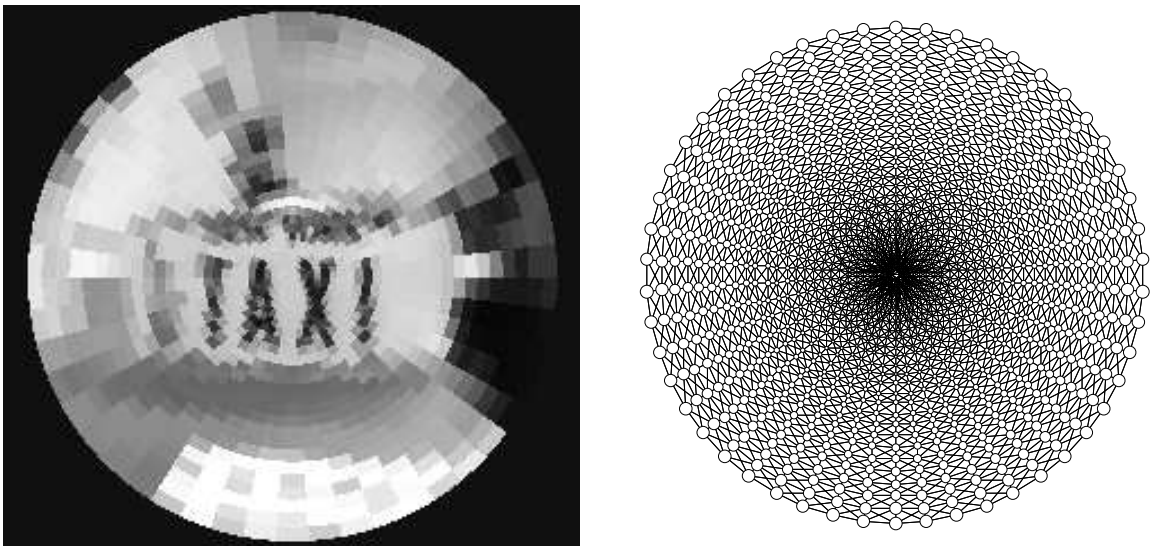

Figure 7: The connectivity graph of a "pure" log polar sensor given by the mapping $\mathbf{w} = \log(\mathbf{z})$, where the origin lies out of the mapping.

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 $R(i,j)$ and $S(i,j)$ with a 3×3 operator. The following procedure finds the edges E_I for the $4-$neighbors. Initially E_I is empty.

for each TV pixel (i,j)
  begin
  $u = R(i,j)$
  $v = S(i,j)$
  for each $(m,n) \in \{(i+1,j),(i,j+1),(i-1,j),(i,j-1)\}$
    begin
      $w = r(m,n)$
      $z = s(m,n)$
      if $(u \neq w$ **or** $v \neq z)$ **and** $e = (\phi(u,v),\phi(w,z)) \notin E_I$
      then $E_I = E_I \cup \{e\}$.
    end
  end

The algorithm can be changed in a trivial way to pick up $8-$neighbors. Another detail is that some points (i,j) may be outside the domain of the map, and $S(i,j)$ and $R(i,j)$ are undefined at those points. Also, this simplified algorithm does not take into account the fact that not all points (m,n) lie inside the bounds of the TV image.

Each graph node p is represented by a constant size data structure in memory. The graph edges $\{(p,q)\}$ are represented by one field containing a list of pointers. For a given graph node p, then the structure for p may also contain fields such as: the pixel array coordinates $(u(p), v(p))$, the pixel centroid $\boldsymbol{\mu}(p)$, the pixel area $a(p) = a(u(p), v(p))$, and the number of neighbors $|\mathcal{N}(p)|$ (defined in section 4.1). The pixel centroid $\boldsymbol{\mu}(p)$ is given by

$$\boldsymbol{\mu}(p) = \frac{1}{a(p)} \sum_{i,j} (i,j)^T \mid R(i,j) = u(p) \text{ and } S(i,j) = v(p). \tag{5}$$

A variety of space-variant sensors have been designed and fabricated [7] [6] [3] [3] [4] [1] [2]. The connectivity graph can be defined for any of these architectures. Using it, we will show how to develop a library of image processing routines that can run on data from any of these sensors, and therefore we provide a generic approach to image processing on an unconstrained sensor geometry.

3.1 Connected components

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

$$F(p,q) = \begin{cases} \text{true} & L(p) = L(q) \\ \text{false} & \text{otherwise} \end{cases} \tag{6}$$

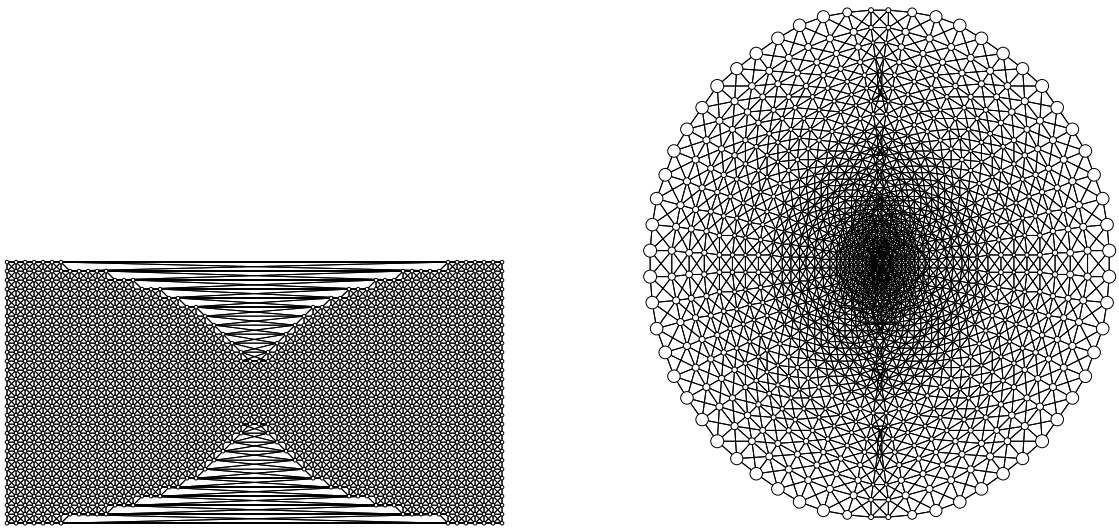

Figure 4: The connectivity graph for the sensor depicted in figures 2 and 3. Circles stand for graph vertices and lines stand for graph edges. The left plot shows the vertices in logmap array coordinates, and the right plot shows the same graph with the vertices in sensor coordinates. The graph explicitly represents the neighbor relations across the vertical meridian.

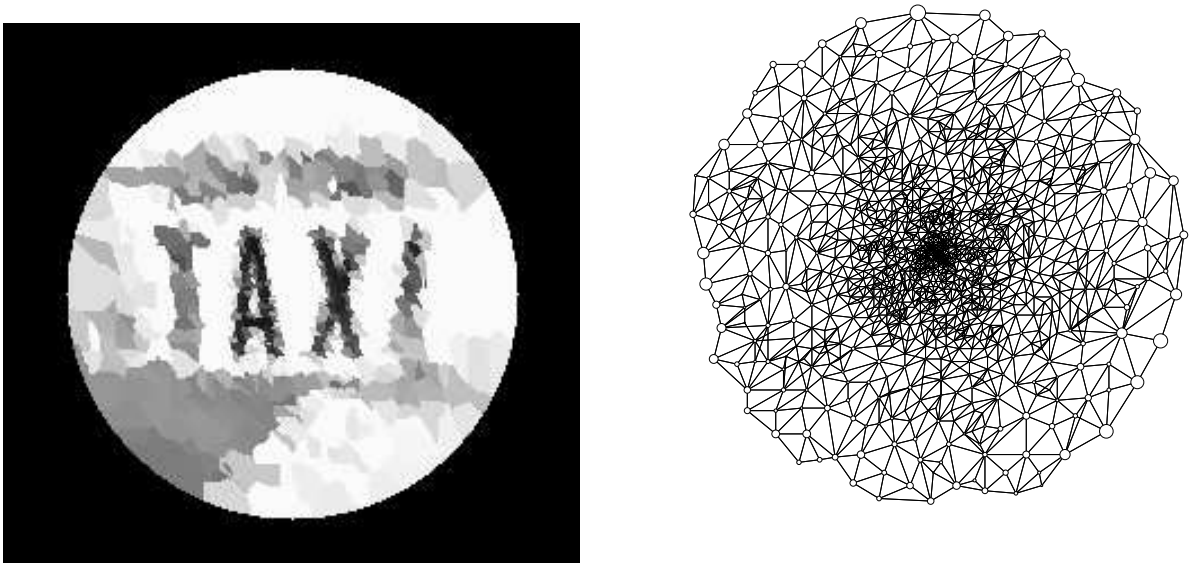

Figure 5: The image from a sensor having a random pixel geometry (left). The connectivity graph for this sensor (right).

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

The space variant sensor image is defined by

$$L(u,v) = \frac{1}{a(u,v)} \sum_{i,j} I(i,j) \mid R(i,j) = u \text{ and } S(i,j) = v. \tag{2}$$

The inverse map, illustrated in Figure 2, is

$$L^{-1}(i,j) = L(R(i,j), S(i,j)). \tag{3}$$

The relationship between the space-variant sensor and the lookup tables $R(i,j)$ and $S(i,j)$ is illustrated by comparing Figure 1 with Figure 2. The gray values in the lookup tables depict the row and column addresses of pixels in the space-variant image array. We observe that if n is the number pixels for which $a(u,v) > 0$, then $n \leq rs$.

3 The connectivity graph

We define the connectivity graph $G = (V, E)$ such that the vertices V stand for sensor pixels and the edges E represent the adjacency relations between pixels. Associated with a vertex p is a 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, $V = \{p_0, \ldots, p_{n-1}\}$, where $|V| = n$ is both the number of logmap sensor pixels and the number of vertices in V.

The definitions in the previous section do not constrain the mapping given by $R(i,j)$ and $S(i,j)$. Some sensor mappings, such as the logmap (Figure 2), are such that adjacent pixels in the TV image map to pixels that are also adjacent in sensor image space. In this case, the connectivity relations are straightforward. Let E_F be the set of edges representing connections between pixels in the forward mapping. The 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 (Figure 2) E_F does not contain the connections across the vertical meridian. To obtain all the connections in the sensor, we place the *inverse image* adjaceny relations in the edge set E_I:

$$\begin{aligned} (p,q) \in E_I \text{ provided } \exists\ i,j,k,l \text{ such that } \quad & R(i,j) = u(p), S(i,j) = v(p) \text{ and} \\ & R(k,l) = u(q), S(k,l) = v(q) \text{ and} \\ & (i,j) \text{ is an } 8- \ (4-)\text{neighbor of } (k,l), \end{aligned} \tag{4}$$

The connectivity graph edge set may be $E = E_I$ or $E = E_I \cup E_F$, depending on the map. Resulting from the discretization of the complex log function $\mathbf{w} = \log(\mathbf{z} + \alpha)$ in the tables $R(i,j)$ and $S(i,j)$, some of the $8-$adjacent neighbors in the forward map of Figure 2 correspond to nonadjacent pixels in the inverse image. In that case we chose $E = E_I \cup E_F$. For the random geometry sensor illustrated in Figure 5, the forward mapping contains only random connections, we use only $E = E_I$.

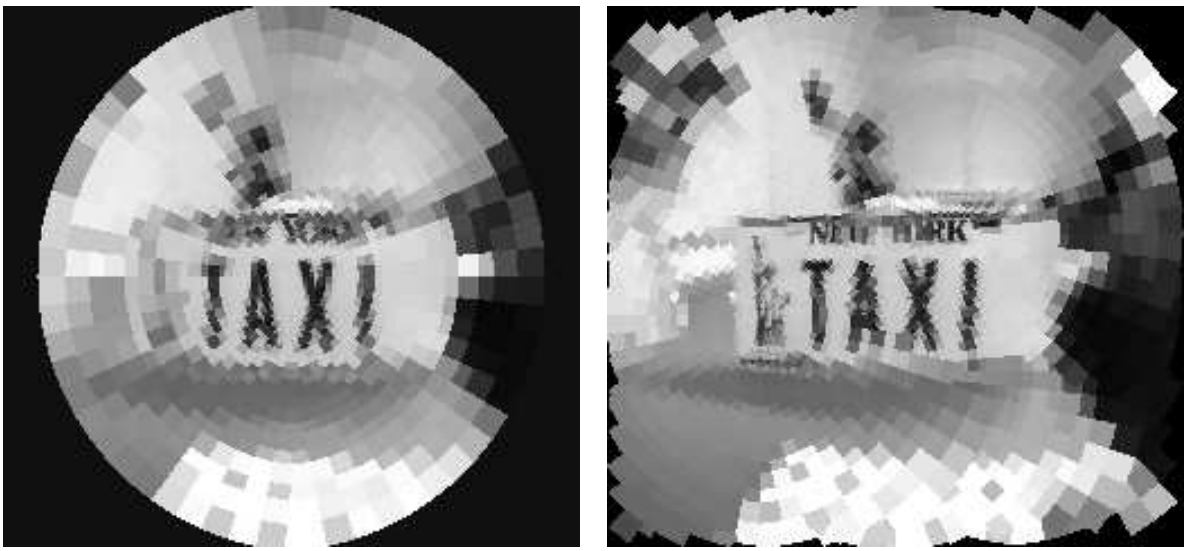

Figure 3: The inverse logmap image $L^{-1}(i,j)$ (left) and an example of progressive coding using blended logmap images (right). The right image is constructed by clipping logmap images having different origins in the original TV image. The clipping preserves the highest frequency information available at each TV pixel.

for space-variant images and apply it to a local binarization operation. Finally, we introduce a variant of the connectivity graph which we call a *translation graph*, and which represents the effects of translation in space-variant images, and demonstrate an implemention of template matching using the translation graph.

2 Space-variant image sensors

A space-variant image sensor is characterized by an irregular pixel geometry. We have experimented with several sensor designs, the common features of which are large scale changes between the smallest and largest pixels, wide field of view, and a pixel population far smaller than that of a conventional matrix image sensor. One such space-variant sensor is the *logmap* [7][6], in which the pixel pattern approximates the sensor geometry of the human eye.

We often want to display a logmap on a TV screen, or to compute one from a TV picture. Thus we can formally define the logmap as a mapping from a TV image $I(i,j)$ where $i \in \{0,\ldots,c-1\}$ and $j \in \{0,\ldots,d-1\}$. Let $L(u,v)$ be the logmap, with $u \in \{0,...,r-1\}$ and $v \in \{0,...,s-1\}$.

The forward mapping from TV image to logmap is specified by two lookup tables $R(i,j)$ and $S(i,j)$ where again $i \in \{0,\ldots,c-1\}$ and $j \in \{0,\ldots,d-1\}$. Let $a(u,v)$ be the area (in TV pixels) of a logmap pixel (u,v).

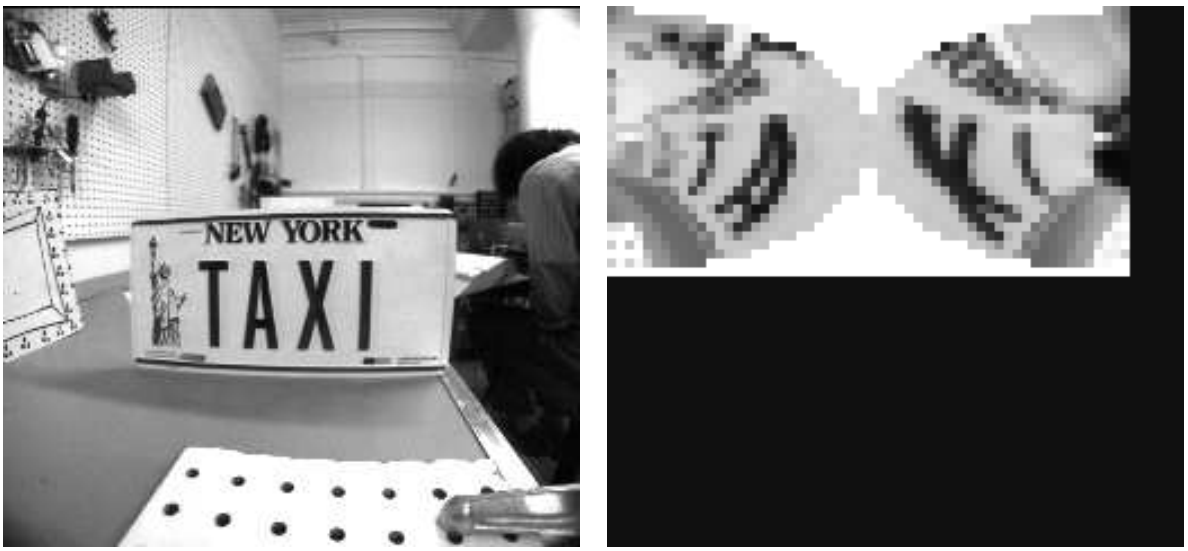

Figure 2: A video image (left) and its logmap image (right). The map is given by a complex function $\mathbf{w} = \log(\mathbf{z}+\alpha)$. The logmap image has the same field of view and maximum resolution as the TV image, but far fewer pixels. The logmap image is shown scaled so that its pixels are printed larger than the TV image pixels.

pattern, such as the one illustrated in Figure 5. 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.

Connectivity was first discussed in relation to image processing by Rosenfeld[9]. The standard image processing texts (e.g. [8] vol. 2 p. 208, [11] 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[7] 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. [10] p. 159ff). Recently, Montanvert et al. [5] have shown how to define an image pyramid recursively from an irregular tesselation.

This paper develops the definition of a connectivity graph to represent connectedness relations in arbitrary space-variant images. Using the graph, we define a variety of image processing operations. The connected components problem for space-variant images then reduces to its graph theory conterpart. 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 [5], we defined a pyramid structure

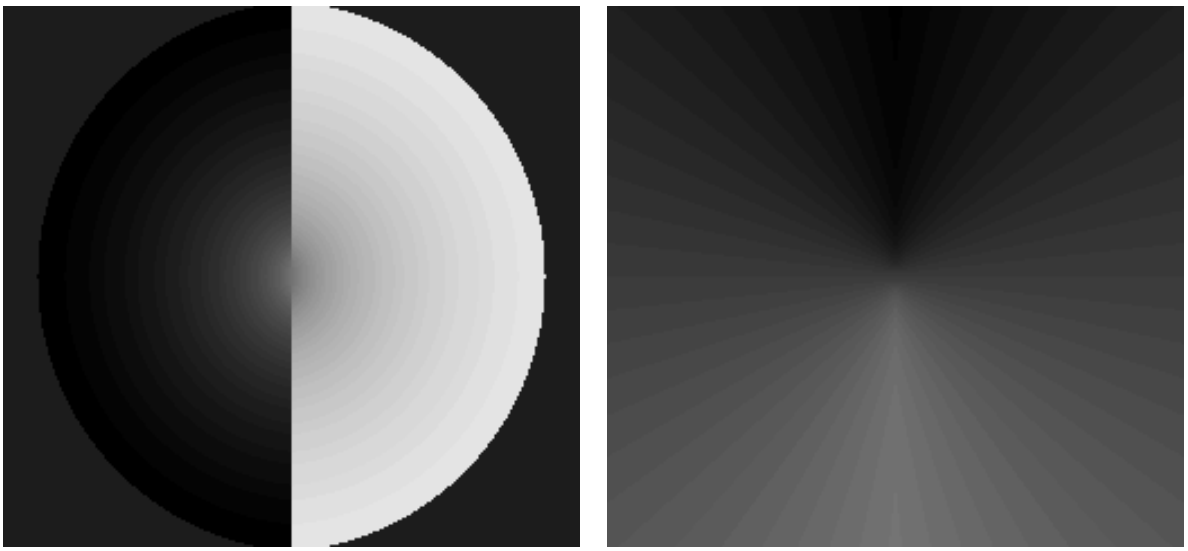

Figure 1: The lookup tables $R(i,j)$ (left) and $S(i,j)$ (right) defining the space-variant image.

built a miniature two degree of freedom pan-tilt device and video camera[17]. This system, together with the motors, control, camera and image processing is capable of performing connected components, motion tracking, and binary thresholding in near real-time (up to 10 frames per second). It weighs several pounds, and fits in a box $5.5'' \times 8.5'' \times 8.5''$. Constructed for a cost of only a few thousand dollars in parts, 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, and CPUs. For these reasons, space-variant sensors in general, and logmap sensors in particular, have begun to attract attention in the machine vision community [18] [7] [6] [3] [2] [4] [1] [20]. A more detailed description of this system is available [16].

The basis of our approach to space-variant image processing is the use of a *connectivity graph* to represent the neighbor relations between pixels in these space-variant images. The importance of the connectivity problem for a space-variant sensor is motivated by examination of Figures 2 and 3. The image on the right of Figure 2 is related to the one on the left by the complex logarithmic mapping $\mathbf{w} = \log(\mathbf{z} + \alpha)$ [7][6]. For more general mappings, we can think of the image on the left of Figure 3 as what the sensor sees and the one on the right in Figure 2 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 image processing libraries to data like that from this sensor, they produce undesireable artifacts along the vertical meridian.

More generally, suppose we have an image sensor in which the pixels have no regular

1 Introduction

We are accustomed to thinking of an image as a rectangular grid composed of square 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[19] as well as *space-variant* artificial image sensors, based on CCD [2][4], MOS [17] and firmware [16] implementations, there exists a pattern of photodetectors having changing size, shape and connectivity across the image array. Since part of the motivation for the use of space-variant sensing derives from its prominent role in the architecture of the human visual system, space-variant sensor arrays are called *foveated*, *retinal*, *logmap*, *log polar*, or *log spiral*. Their 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.

The space complexity aspect of logmap sensors is particularly attractive and has been analyzed in detail [6]. In this work, a spatial quality measure, Q_s, for sensors is defined to be the ratio of its work-space to maximal resolution. Logmap sensors can each achieve comparable Q_s to conventional uniform sensors that are one to four orders of magnitude larger. Image simulation of the human space-variant architecture suggests that this ratio may be as high as $10,000:1$ [21]. In current implementations of space-variant machine vision systems, logmap sensors with between 1000 and 2000 pixels have comporable Q_s to conventional sensors in the range of 256×256 to 512×512, a compression of between $60:1$ to $250:1$ Moreover, as shown in [6], Q_s grows exponentially with the number of pixels in the logmap sensor, thus providing a highly favorable route to upgrading sensor quality. The high compression ratios cited above for the human visual system derive from the fact that human acuity, which is roughly one-arc minute in the fovea, when extrapolated over a 120 degree visual field, yields a constant resolution sensor with roughly 0.2 gigapixels. The logmap sensor with comparable Q_s has roughly 10^4 to 10^5 pixels.[1]

In this discussion, compression refers to the ratio of the number of pixels in a conventional uniform sensor to that of a space-variant sensor with the same Q_s. The issue of image compression, in the conventional (e.g. JPEG) usage is an independent issue not addressed in this paper, but we point ou that progressive video coding, in which a video image is represented by a sequence of logmap images, is one application that has benefitted from the high compression ratio of the logmap transform [Figure 3]. If each logmap image is centered on a different point in the video image and clipped so that we preserve the highest frequency information available at each point in the video picture, we can progressively construct a video image of increasing detail. Rojer [7] reported some experiments with progressive logmap video coding, and discussed the problem of selecting the best sequence of gaze points.

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 constructing small and inexpensive active vision systems. Recently, we have

[1] A factor of two has been included for sampling. We have verified these estimates by computer simulation of images and verifiying the ability to resolve, for example, the image of an eye chart in the logmap image [21].

forms when defined on space-variant arrays. 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:

- 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.
- We show how to write local image operators in the connectivity graph that are independent of the sensor geometry, and solve a sensor calibration problem with these operations.
- We relate the connectivity graph to pyramids over irregular tessalations, and implement a local binarization operator in a 2-level pyramid.
- Finally, we expand the connectivity graph into a structure we call a translation graph, representing the effects of translation in space-variant image sensors. Using the translation graph, we define an efficient algorithm for translation in the logmap image 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 *

Richard S. Wallace †
Ping-Wen Ong †‡
Benjamin B. Bederson †‡
Eric L. Schwartz †‡§

†Vision Applications, Inc.
611 Broadway #714
New York, NY 10012

‡Courant Institute of Mathematical Sciences
New York University
251 Mercer Street
New York, NY 10012

§Computational Neuroscience Laboratory
Department of Psychiatry
New York University Medical Center
550 First Avenue
New York, NY 10016

October 8, 1991

Abstract

Space-variant sensors have nonuniform sampling across the image plane. Partially motivated by the observation that human vision is strongly space-variant, yielding an image compression for the human system that is estimated to be as much as four orders of magnitude, 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 fascinating 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

*supported by DARPA/ONR #N00014-90-C-0049 and AFOSR Life Sciences #88-0275. Please address all correspondence to the authors at Vision Applications, Inc., 611 Broadway #714, New York, NY 10012. This report is copyright (c) 1991 by Vision Applications, Inc.