Perceptual Systems for Robots TR 83-24

PROFESSOR MICHAEL A. ARBIB, DR KENNETH J. OVERTON and DR DARYL T. LAWTON

Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts, USA

This review presents a sampling of recent research on the design of perceptual systems for robots, with special emphasis on pattern recognition based on an array of touch sensors, and optic flow techniques for depth extraction and navigation based on a sequence of visual images. It not only presents specific work in machine vision, machine touch and robotics, but also illuminates what we believe to be general principles for the design of perceptual systems for an animal, or human, as well as for a robot.

A hypothetical program which indicates the way one might integrate vision and touch in the control of movement is shown in Figure 1. This is not something yet implemented on our robot system, but it provides a simple example of the shape of things to come. It indicates the way in which the sensory systems described below might be used in a coordinated way. The task is simply to reach out and grasp an indicated target object. When human beings perform this task, they do not reach to the object and then start shaping the hand to grasp the object; rather, they use visual cues about the shape and orientation of the object to determine, as movement toward the object begins, the distance between the fingers and the thumb and the orientation of the hand. Thus, when the hand reaches the object, only minor shaping is required, under a delicate spatial pattern of tactile feedback, to complete the actual grasping that conforms the shape of the hand to the shape of the object.

INTEGRATING VISION AND TOUCH IN MOTOR CONTROL

A program for this would not be a serial computer program in which one movement is done at a time, but would rather be a coordinated control program in which a number of control systems are phased in and out in various patterns to complete the task. Figure 1 suggests one pattern whereby the various subsystems, the perceptual schemata of visual target location in the top of the figure and the motor schemata of reaching and grasping of target, below, could pass activation and data to each other. First there would have to be visual location of the target on the basis of some recognition criteria and the available visual input. Successful visual location of the target, using the dashed lines in Figure 1 to indicate activation, would then turn on perceptual schemata to make available the size and orientation of the target. When we activate the reaching, this sensory information, coded now in terms of salient parameters of the object, is available not to one, but to several motor schemata. We turn on the ballistic movement schema which throws the hand toward the object, but at the same time activates those schemata that rotate the hand and adjust the separation of the fingers.

Figure 1 shows hand rotation and finger adjustment as part of the grasping schema, the control of the hand's shape to grasp the object successfully. What is interesting is that once these subschemata complete their initial setting, they turn themselves off. The grasping schema does not wake up its other subschemata until the ballistic movement has reached the stage where visual or tactile feedback states that movement of the hand toward the object is complete. This message wakes up the actual grasping schema, not the ones that use visual cues to shape the hand in anticipation, but the one that uses a delicate spatial pattern of tactile feedback to conform the grasp to the hand.

This example indicates the robot control architecture we may expect in the future, in which multiple chips are used for differential processing, passing messages to each other, many being active simultaneously, in a style we call cooperative computation.

TACTILE PERCEPTION

In the first stage of our research on touch, reported here, we studied a simple robot hand with two fingers, one of which has a touch sensor. The important point is that the touch sensor is not a simple on/off contact switch, but comprises an array of force sensors, giving us the possibility of pattern recognition. In due course, our research will close the loop, using such pattern recognition to guide coordinated hand movement. Here we emphasize the use of touch to recognize what is being touched and to locate it so that if further

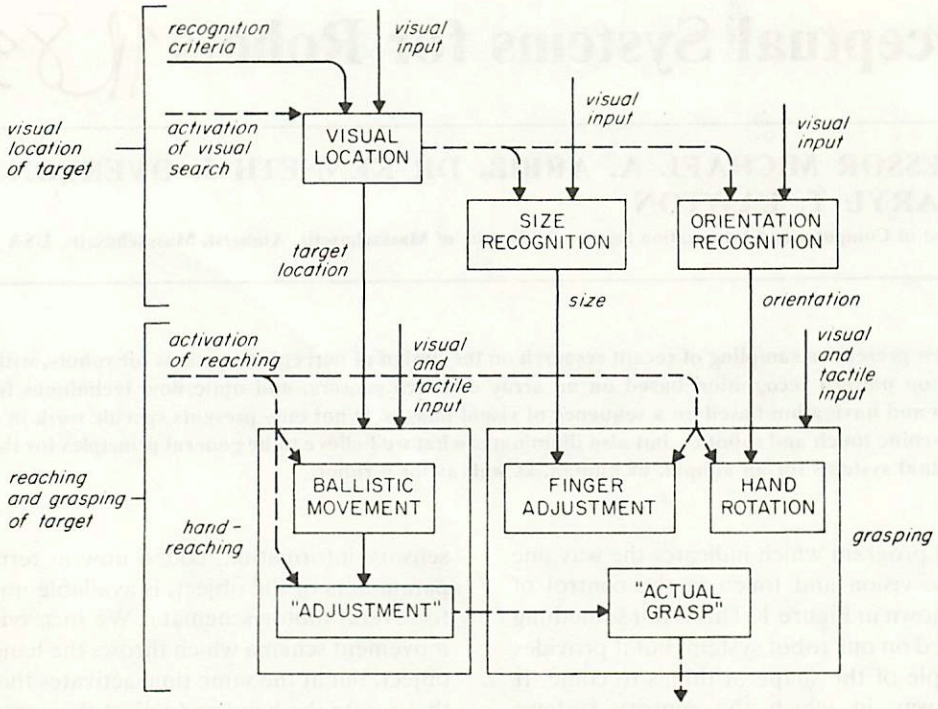


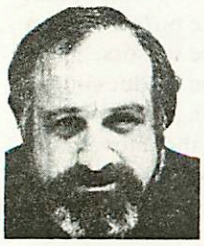
Figure 1. A coordinated control program for visually locating, and then reaching and grasping, an object. (From Arbib.[1])

manipulation is required – think of the robot as working on an assembly line – the relative position of hand and object will provide the necessary parameters.

Another area of current research, going beyond what we report here, concerns dynamic patterns of touch. For example, a particular static force pattern, maintaining a hold upon an object, will change during the use of that object during assembly. The dynamic pattern of force will provide crucial feedback to guide the fitting together of parts. Before moving on to an interesting example of tactile pattern recognition, we mention a trivial example (Figure 2(*a*)): holding an egg, and then printing out a classification such as *Jumbo Grade A*. This is trivial because the program did not recognize the egg as such – all it had to do was process the separation of the fingers to come up with the appropriate classification, using a table of the range of sizes to be classified. This is not intended as an example of the state of the art, but rather serves to emphasize the point that one wants to know, for any particular discrimination task, the minimal sensory information needed within the context of a given set of objects.

A more interesting pattern recognition task involved recognition of each type among several different components: in one study, the parts included three transformers which are of the same width because they are to be used interchangeably in the assembly of some computing equipment. Thus the simple strategy of using finger separation would not discriminate between them. Yet the correct manipulation for assembly may depend on the type of transformer, and so more subtle use of the pattern of the ends of the transformer is required to make the discrimination. Below, we give some insight into one simple algorithm for such discrimination, and also an indication of what more complex algorithms would look like.

In a restricted environment, such as that of a robot working on an assembly line, we do not have to solve very elaborate perceptual problems if we can assume that the system will be working with just one object from a small repertoire of objects. In this case, the recognition program has to (a) decide which is the most plausible hypothesis from a rather small set to identify the part; (b) determine if there is something



PROFESSOR MICHAEL A. ARBIB is Professor in the Department of Computer and Information Science, Adjunct Professor of Psychology, and Director of the Center for Systems Neuroscience, all at the University of Massachusetts at Amherst. Born in England, he studied mathematics at the University of Sydney and at MIT, and his research has embraced brain theory and artificial intelligence, as well as the mathematical theory of computation and control. After working at Stanford University, he moved to Amherst, where he was Foundation Chairman of the Department in which he now works. He wrote his first book, *Brains, Machines and Mathematics*, in 1964.
Address: Department of Computer and Information Science, the University of Massachusetts, Amherst, Massachusetts 01003, USA.

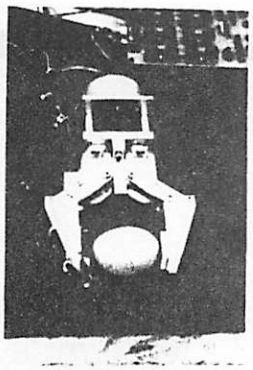


Figure 2(*a*). A URI gripper, with a tactile sensor array attached to one fingertip, holding an egg.

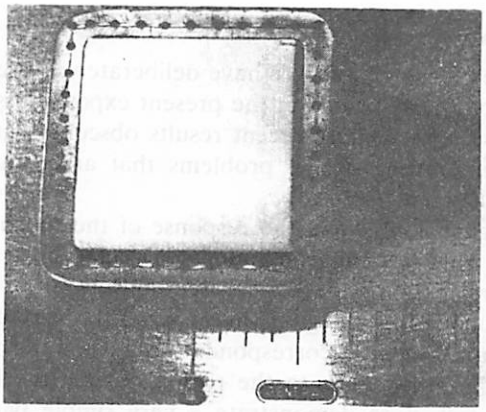


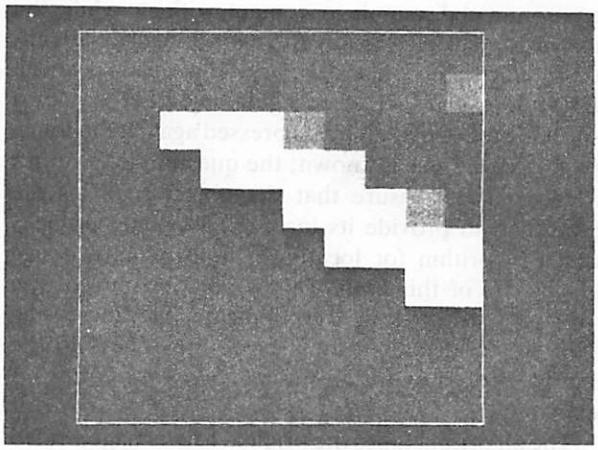
Figure 2(*b*). The tactile sensor array.

wrong, perhaps based on an error measure, so that if a part does not quite meet the usual specifications, this can be detected and some appropriate course of action taken, rather than proceeding with assembly; and then, if a part is successfully recognized, (c) report not only what it is but where it is so that the appropriate pattern of coordination of this hand with other objects and other hands can take place.

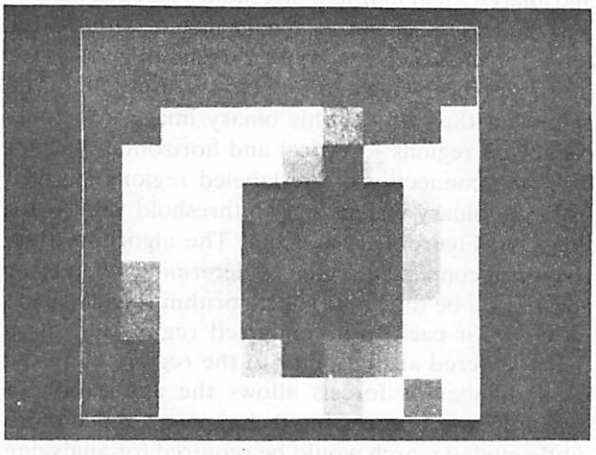
Figure 2(*b*) shows a prototype of the touch array we have developed[2] mounted on the finger of a simple gripper. It comprises a rubber pad in which is embedded a crisscross matrix of wires, so that when you press on the pad, the resistance at each junction changes. With proper circuitry we can obtain an array of numbers proportional to those resistances. The readout is not a linear function of force, but is usually a monotone function of the force, and thus adequate for pattern recognition. Just as people in picture processing talk about picture elements and abbreviate them to *pixels*, so we talk about force elements and abbreviate them to *forcels*.

Scanning circuitry has been developed allowing a particular forcel to be selected. The resistance of the forcel forms part of a voltage divider, with the voltage seen across a reference resistor going through an analog–digital converter to a PDP 11/23 which can then build up, in perhaps 0.02 second, a complete force image. We can display these images using computer graphics. In this review we use a display in which we have a brightness array, with the brighter the pixel, the larger the value of the forcel it represents. Figure 3(*a*) shows the diagonal pattern of intensity which corresponds to pressing a 1/16-inch shaft into a 1-inch-square pad containing a 10×10 array of forcels.

Note that there is one forcel in the top right corner which is black. That proved to be a dead forcel – there were no responses at that point of the array, no matter what stimuli were applied. In developing touch sensors, we have to improve quality control to ensure that there will be no dead forcels as there were in the prototype. Then, before using the array for pattern recognition, we carry out systematic analysis of how the activity in a particular forcel varies with stimulation, and we use that to calibrate the forcels to compensate for different response characteristics. In the case of Figure 3(*a*), no calibration was done, and the top right forcel is of no use in



(*a*)



(*b*)

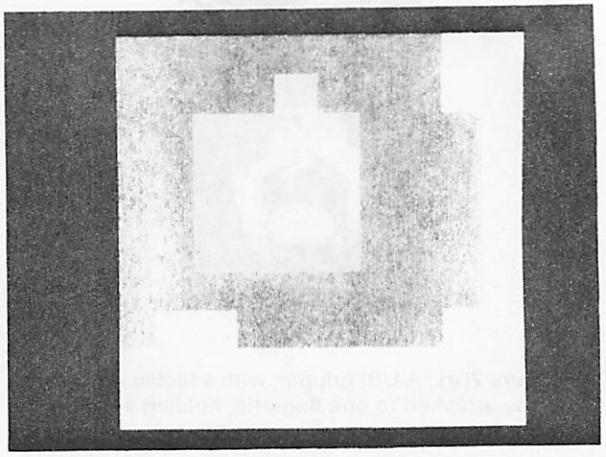
Figure 3. These two graphic displays translate the resistance read from each forcel on the tactile sensor array into brightness. The greater the force, the brighter the image, though the relation is not linear. (*a*) The force image for a thin cylinder pressed against the array. (*b*) The image for a square plate with a hole in the center.

pattern recognition. We have deliberately chosen to use our early results in the present exposition since the smoother, more recent results obscure some of the interesting design problems that arose during development.

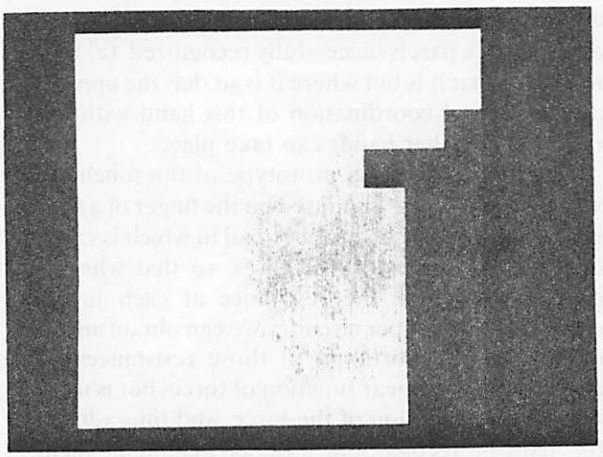
Figure 3(*b*) shows the response of the touch pad to a square plate containing a circular hole. The central region is dark, representing low force; it is surrounded by a region which, using the very crude quantization here, corresponds to the plate: the outer region corresponds to the region outside the hole. Our task is to demonstrate a very simple pattern recognition problem: to locate the hole. In a restricted robotics application, a tactile recognition program need not be able to handle exotic objects, or objects of variable shape and size, or even, as in vision (see the final section, on visual systems), the problem of compensating for distortions of the image due to distance and perspective. The tactile array sensor merely provides an image corresponding to the environmental stimuli in contact with it, while the image available in vision is a two-dimensional representation of an inherently three-dimensional world. If the job is in fact to recognize whether or not a specific type of plate is being pressed against the touch pad, then the size is known; the question is to return a confidence measure that the object really is the plate and to provide its location. We can outline a crude algorithm for locating a hole of known size given data of this kind. This is not the state of the art, but rather serves to indicate the increasing subtlety of such algorithms as we move beyond the simple size monitoring of our *Jumbo Grade A* example.

The algorithm takes the original force image, scaled such that all forcel values are within a specified range, and analyzes it at multiple thresholds. At each threshold, the image becomes a binary image, with a 1 for those forcels which are above threshold, and a 0 for those forcels which are below threshold. The algorithm then breaks this binary image into four-connected regions – vertical and horizontal, but not diagonal, connections. The labeled regions, derived from the binary images at two threshold values, are shown in Figure 4(*a*) and (*b*). The algorithm then tests each connected region to determine whether or not it could be the hole. The algorithm calculates the centroid for each four-connected region and fits a circle centered at that point to the region. Here the small number of forcels allows the application of straightforward processing techniques; a much more subtle guided search would be required for analyzing for example a 512×512 visual array. Issues of computational efficiency will become far more important than they are at this prototype stage as the spatial resolution increases, and the number of sensors employed grows.

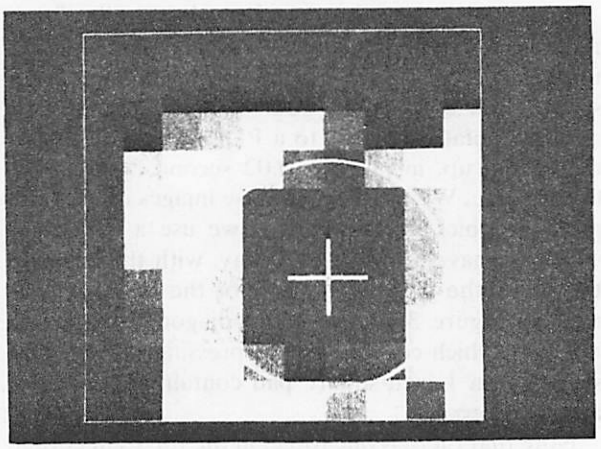
To test the hypothesis that a point is at the center of the circle, the algorithm measures how much of the region of the circle is covered by the posited threshold region, and how much of the threshold



(*a*)



(*b*)



(*c*)

Figure 4. In (*a*) and (*b*) we see, for two different thresholds, the result of thresholding the image to yield a binary image (1 = above threshold; 0 = below) then forming connected regions with the same binary value. (*c*) The current result of estimating circle position by performing a best fit to the regions in the threshold series. The cross indicates the position of the center of the circle.

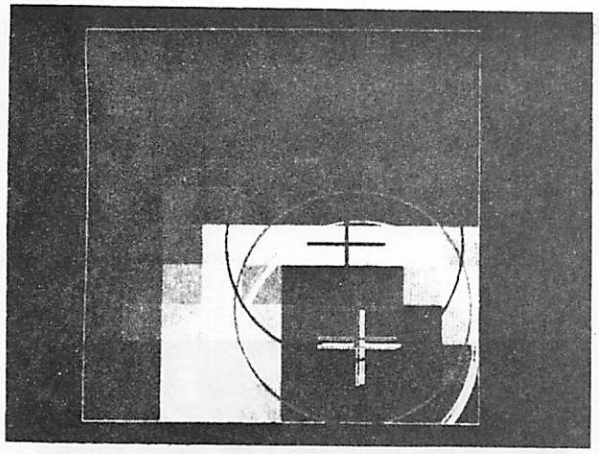


Figure 5. If the 'holey-plate' of Figure 4(*b*) is moved across the sensor array, we can estimate the hole position for each successive snapshot. Here we superimpose the resultant series of estimates.

region lies outside the circle. The sum of the two discrepant areas provides the error measure. For each threshold level, the region with the minimum error measure is saved. Centers with low error measure get a high confidence measure, and vice versa. The centroids of the saved regions are combined in a weighted average based upon the error measure to produce a final estimate. Figure 4(*c*) shows the final weighted combination of these estimates. Note that this image is somewhat biased by the fact that there was a small vote all the time for a hole comprised solely of the bad forcel. As mentioned above, we now have techniques to ignore that forcel automatically and thus not bias the estimates.

Figure 5 shows the result of moving the plate across the pad. We see a sequence of estimates of where the hole is, with a general progression downwards and to the right, but it is not a smooth progression. In our work on sequences of visual images, we have already developed prediction algorithms,[3] but have not yet applied these to our tactile processing. These yield more accurate and efficient performance, for example, by weighting search to the neighborhood of the previous estimate and by extrapolating the last few estimates. The result is a weighted sum of an extrapolation and a neighborhood search which gives an increasingly reliable estimate of hole position as time progresses, as long as the hole follows a reasonably smooth trajectory.

OPTIC FLOW, NAVIGATION AND DEPTH PERCEPTION

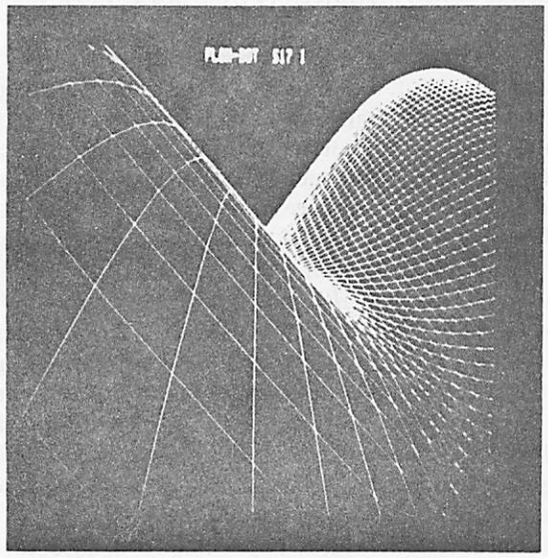
The attention of the psychological community was first drawn to optic flow fields by J. J. Gibson[4] during World War II, when he was studying the visual performance of pilots landing airplanes. He noted that, quite apart from the recognition of landmarks, there was the diffuse information of flow of patterns across the retina; this could be used by the pilot in deciding when to switch his trajectory from descent to leveling out, prior to landing.

Figure 6 shows three successive views as seen by a simulated robot, called the *flowbot*, as it flies through a landscape of Gaussian mountains. Its motion is unrestricted but the orientation of the sensor is fixed relative to the environment. Thus, the robot's motion can be described as successive translations along an arbitrary trajectory. What sort of visual information will enable it to avoid a simulated collision with the hillside? Figure 6(*d*) shows an optic flow field obtained by drawing arrows, each going from a point in Frame 1, 6(*a*), to a corresponding point in Frame 2, 6(*b*). This results in a vector field of displacements of the retinal image from one moment to the next, called the *optic flow field* or *optic displacement field.* Figure 6(*e*) shows the flow from Frame 2, 6(*b*), to Frame 3, 6(*c*).

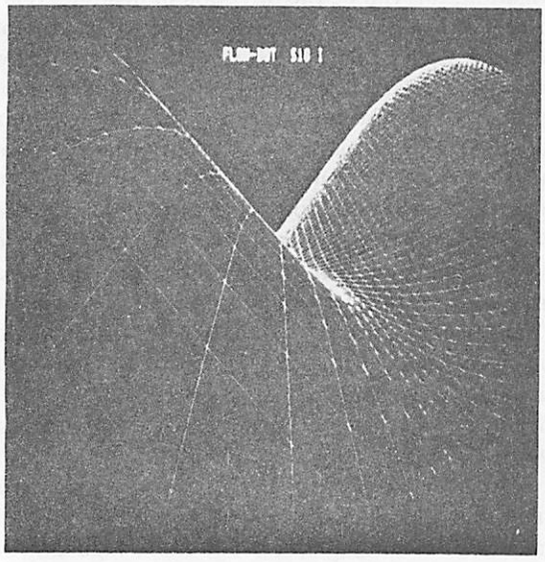
From the flow field, it is possible to deduce very important information for purposes of visually guided navigation; this can be the direction of heading, relative to the environment, and the relative depths of environmental points.[5] Note that the flow radiates from a particular image point termed the Focus of Expansion, the FOE. This point corresponds to the intersection of the image plane and the translational axis of observer motion. Thus, the FOE specifies the direction in which the flowbot is heading. To relate that to everyday experience, recall that when you are driving down a straight road and the image of the world is flowing across your retina, there is a point on the horizon from which the whole visual world appears to stream – the FOE – and it is towards that point that your course is taking you.

In Figure 6(*d*) we see that the flowbot is heading towards the hillside. A navigation system would take the pattern of optic flow and use it to compute a change of course, rather than using a single control parameter, like an angle of orientation. Figure 6(*c*) shows the flowbot's view after such a correction, and Figure 6(*e*) shows that the system has changed its course so that the focus of expansion is moving off the hillside. The relative depth of environmental points is recovered from its position and from its rate of motion along a flow path, radiating from the FOE. As we will see, environmental depth is recovered from a translational flow field in units of *time-until-adjacency*, or how long it will be until an environmental point is beside the flowbot. For motion along arbitrary trajectories, the position of the FOE and the time-until-adjacency values will change.

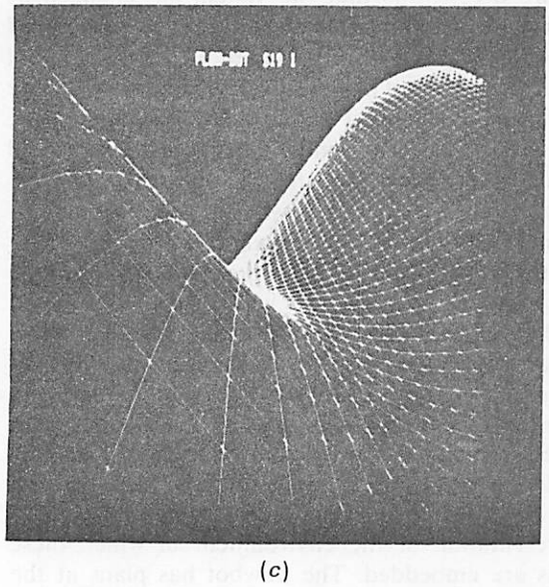
The control of a flying robot is extremely complex. In particular, it involves taking into account and representing a whole array of spatial information to determine time-critical decisions. Additionally, it is necessary that the control be done in an expectancy-guided or model-driven mode. That is, there is a plan specifying behavior over different time scales and a representation of the environment in which these plans are embedded. The flowbot has plans at the



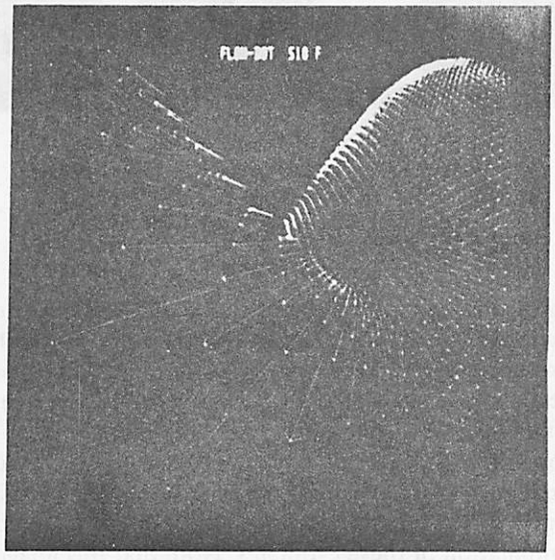
(*a*)



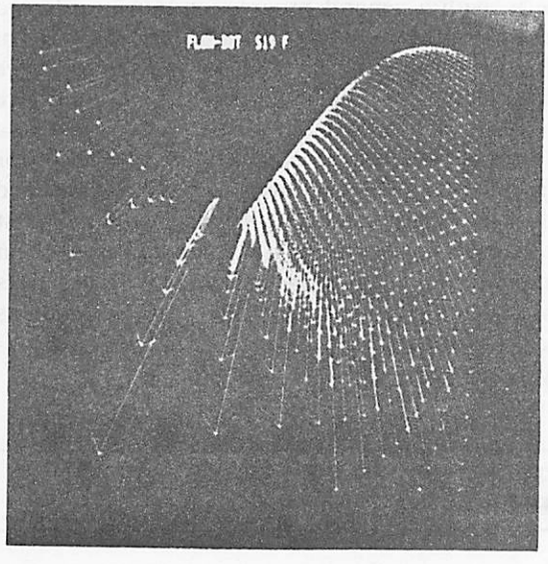
(*b*)



(*c*)



(*d*)



(*e*)

Figure 6. (*a*), (*b*), and (*c*) show three successive views of a Gaussian hillside, while (*d*) and (*e*) show the optic flow fields – vectors indicate displacement of corresponding points in two successive images – for (*a*)→(*b*) and (*b*)→(*c*), respectively.

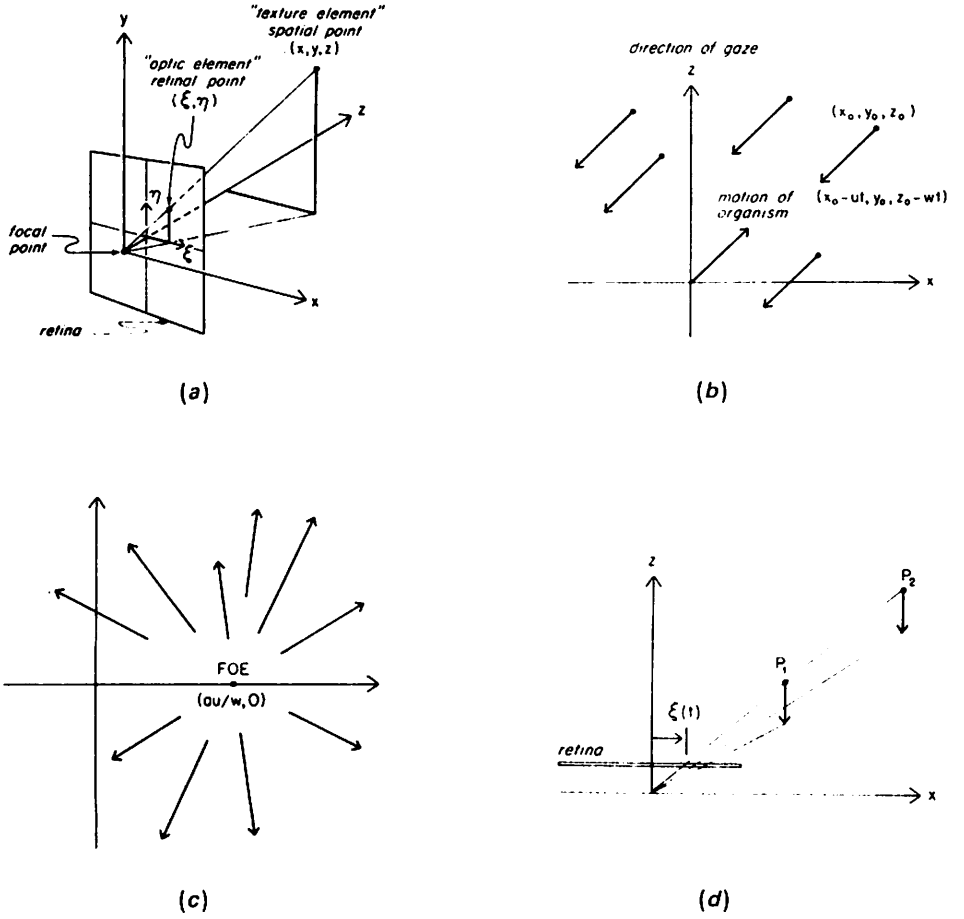


Figure 7. (*a*) To simplify visualization of correspondence between retinal and spatial coordinates, the focal point has been placed behind the retina, but this is, of course, equivalent to placing the retina behind the focal point and then inverting coordinates. We thus have $\xi = ax/z$ and $\eta = by/z$, where a and b are positive scale constants; ξ, horizontal coordinate, and η, vertical coordinate, of a point on planar retina; x, y, and z, horizontal and vertical body-centered coordinates and coordinate of distance in organism's line of gaze. (*b*) Relative motion of organism and environment; (x_0, y_0, z_0) is initial position of point in organism's frame of reference; $(x_0 - ut, y_0, z_0 - wt)$ is its position at time t, where $(u, 0, w)$ provides the (x, y, z) components of the organism's forward velocity relative to the environment. (*c*) Optic flow radiates from common FOE when motion of organism relative to environment is constant and forward; a, u, and w are as given in (*a*) and (*b*). (*d*) For a given retinal x-coordinate $\xi(t)$ at time t, the closer the corresponding texture element P_1 or P_2 is to the organism, the larger is the velocity $|\dot{\xi}(t)|$ with which the optic element at $\xi(t)$ moves across the retina.

level of 'get around the next potential obstacle; then find the next potential obstacle and repeat.' This is based upon monitoring the relations between the position of the FOE relative to extremal boundaries with the time-until-adjacency values that correspond to particular Gaussian mountains. The flowbot represents the environment in terms of position of FOE, time-until-adjacency maps, and extremal boundary maps. It does not even have particular knowledge about mountains. It attempts to capture the control of navigation in an egocentric space, based upon an environmentally stabilized body. This is assumed to be interfaced with more long-term spatial knowledge as modeled by Kuipers[6,7] and in Lieblich and Arbib.[8]

We next turn to two observations concerning the computation of the flow field and the FOE. The first is that in the real world hillsides are rarely marked with coordinate systems. Therefore, the process which was easy in a simulated system, like that of Figure 6 of matching points in Frame 1 with points in Frame 2, actually involves resolving considerable ambiguity in processing a sequence of real images. Which small part of texture on a real hillside in Frame 1 is to be matched with which small part of texture on the hillside in Frame 2? This is what we call the *stimulus-matching problem*: Given a pair of views at the same time in stereopsis, or a sequence of views at different times in optic flow, how do we match up corresponding features from frame to frame? We shall discuss algorithms for this below.

The second point is that, even if the stimulus-matching problem has been solved to determine the optic flow, where in fact is the focus of expansion? It is easy for a human to point to a small area in Figure 6(*d*) and agree that it contains the focus of expansion. But how do we pick the exact point, and how is that determination to be turned into an algorithm? For navigational purposes, some small

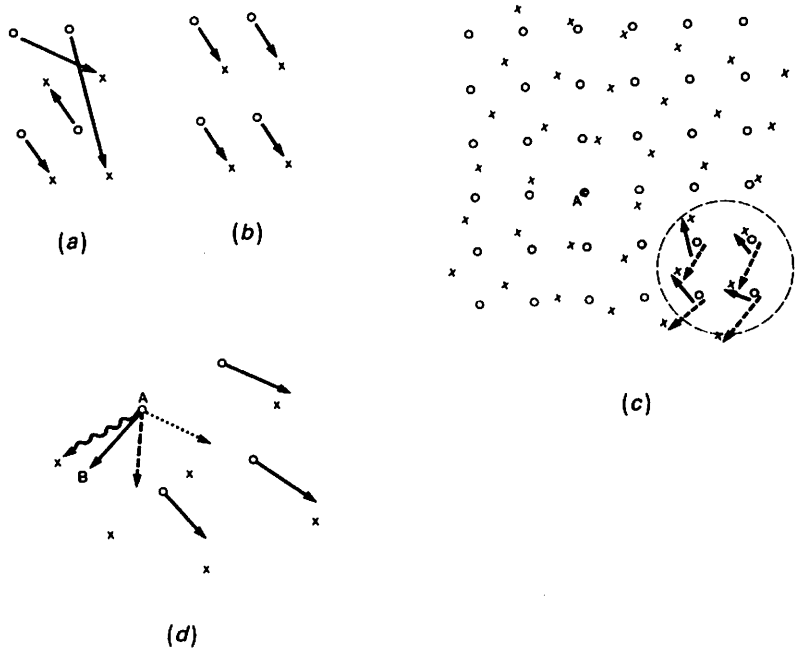


Figure 8. In a world made up of surfaces, nearby features are likely to have similar optic flow. Thus the flow of (*b*) is far more likely to be correct than that of (*a*). (*c*) Frame 1 comprises the dots indicated by circles: Frame 2 is obtained by rotating the array about the pivot at *A* to place the dots in the positions indicated by crosses. The dashed circle at lower right is the receptive field of a local processor. The solid arrows indicate the best local estimate of the optic flow, the dashed arrows show the actual pairing of features under rotation about *A*. (*d*) The circles indicate features in Frame 1, the crosses features in Frame 2, and the solid arrows the current estimate of the optic flow – the head of the arrow shows the posited position in Frame 2 of the feature corresponding to the Frame 1 feature at the tail of the arrow. *Feature matching* alone would adjust *A*'s optic flow to the wavy arrow pointing to the Frame 2 feature nearest to *B* (the current estimate of *A*'s Frame 2 position), local smoothness would yield the dotted arrow, the average of the optic flow of the neighbors, whereas our relaxation algorithm yields the dashed arrow as a weighted combination of these two estimates.

region of confidence would be sufficient, but considerable problems still remain in finding an algorithm which will take an optic flow field and infer the focus of expansion. Below, we shall discuss how to interdigitate a variety of processes for solving stimulus-matching and for finding the focus of expansion.

Such interaction between algorithms returns us to a constant theme of this paper, that of cooperative computation: when we design complex action/perception systems, very rarely do we design one system that completes its task, switches off, and lets another system take over. Rather, as in Figure 1, we have a whole set of simultaneously active subsystems passing data and activation messages to each other.

With this background, let us look in more detail at the geometry of optic flow. Figure 7 shows the case of a planar retina and uniform translation – that is, a constant relative velocity of all objects in the environment relative to the observer. In Figure 7(*a*) – with the retina on the wrong side of the focal point just to make the external and retinal coordinates the same way round – we see how a given ray projects to a single point on the retina. In Figure 7(*b*), we see that as the organism moves a certain distance forward into the environment (looking down on the landscape), a distance indicated by the arrow radiating out from the origin, it is as if each point in the environment has an equal and opposite displacement, shown by the other arrows, relative to a coordinate system centered on the robot. (Note that in this review we discuss only uniform translation, but we have also developed algorithms for nonuniform trajectories.)

If, using the geometry of Figure 7(*a*), we project the coordinates for the motion of Figure 7(*b*) onto the retina, we obtain the display of Figure 7(*c*): with a planar retina, the uniform translation of each point in the environment relative to the observer is converted into a vector on the retina, with all vectors radiating from a common point, the FOE (Figure 7(*d*)) (Recall the intuition driving down the road: 'Optic flow flees the FOE.')

A RELAXATION ALGORITHM FOR STIMULUS-MATCHING

We now return to the stimulus-matching problem. As mentioned above, unlike the situation in Figure 6, points of the real world are rarely tagged with their coordinates. One way to solve the stimulus-matching problem is to do expensive processing on single

images, for example to recognize actual objects, so that distinctive features of objects are then readily matched. But here we are concerned with relatively cheap ways of stimulus-matching which do not rest on a prior perceptual analysis. An analogy: when walking down the street you have recognized certain objects, but if somebody suddenly bears down on you from the side, you will jump out of the way without even recognizing what is there. Optic flow is akin to that process which we suggest comes before detailed classification of the input, yet conveys information useful for navigation.

In Figure 8 we pose the stimulus-matching problem for features which are indistinguishable. In Figure 8(a) and (b), Frame 1 has four features shown by circles while Frame 2 has four features shown by crosses. How do we match them up? In general, we look at a world made up of surfaces so that nearby points will on the average be on the same surface and thus, although they will not move with exactly the same velocity (consider a rotating surface), they will have similar velocity. Thus a good hypothesis is that the optic flow is locally smooth, and for that reason the hypothesis shown in 8(b), which meets the *local smoothness criterion*, is a better bet than that of 8(a), given just the data available there.

Now consider the large circle in the lower right-hand corner of Figure 8(c), and try to match up the small circles within it with the crosses, while respecting the local smoothness condition. The solid arrows are our best hypothesis using the local data – they constitute a locally smooth flow, and the flow vectors are relatively short. However, if we look at Figure 8(c) globally, we see that Frame 2 is obtained from Frame 1 by rotation about the point A, and as a result the correct flow for that pattern is obtained by the dashed arrows shown for the features in the large circle.

The point is this: we expect the architecture of complex visual systems, whether implemented for a robot or occurring naturally in an animal or human brain, to involve local processes communicating with each other. What we show in Figure 8(c) is that even with constraints like nearest-neighbor match or local smoothness of the flow field, a local view need not be correct. The question then is: how do we set up a communication process which will get each local view to influence its neighbors in such a way that eventually there will be a global pattern of consistency?

The general name for such a process is a *relaxation procedure*, derived from Southwell's study of how to compute the conformation of a beam to various loads: Start from an initial hypothesis as to what the vertical displacement at different points of the beam would be, then use the stress–strain equations to adjust hypotheses on nearby displacements, and iterate the adjustment until the pattern of displacements along the entire beam relaxes into a pattern in equilibrium with the external forces.

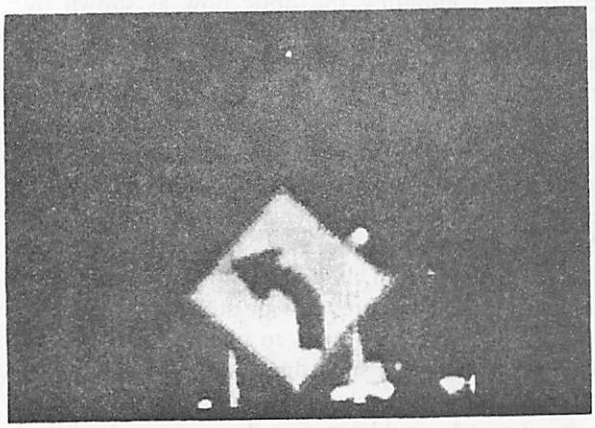
Figure 8(d) is a diagram of one relaxation procedure for computing the optic flow as shown by Prager[9] and by Prager and Arbib.[3] This procedure has two time scales: one is a time scale on which Frame 1 comes in, then Frame 2, and so on. But we now turn to a finer time scale of successive iterations used to estimate the optic flow based on displacement from Frame 1 to Frame 2. At each iteration we update our optic flow estimate for each feature in Frame 1. The solid arrows in Figure 8(d) represent the result of one such iteration, indicating for each Frame 1 feature (a circle) where it is posited to move into Frame 2. Consider two ways to update the estimate for Feature A. Local smoothness would replace the current vector by the average of its neighbors; neighborhood match would examine the position B of the current Frame 2 estimate and replace it by the Frame 2 feature that is nearest with respect to a metric which includes both similarity of feature and displacement on the image. In fact, our algorithm takes a convex combination of the smoothness hypothesis and the nearest-neighbor hypothesis. Each iteration applies the updating rule to every feature in the image.

Computation experience with simple images has shown that about 20 iterations yield a stable estimate of the optic flow. If we have a sequence of images, we can do much better by using the computed displacement for a feature from Frame n to Frame $n+1$ to initialize our hypothesis for its displacement from Frame $n+1$ to Frame $n+2$. This is of course only a first approximation – consider a rotating flow – but we find that with this *informed initialization* we not only get a more reliable estimate of the optic flow field, but we require only five or six iterations, rather than 20, to come to a state where there are no significant changes with further iterations.

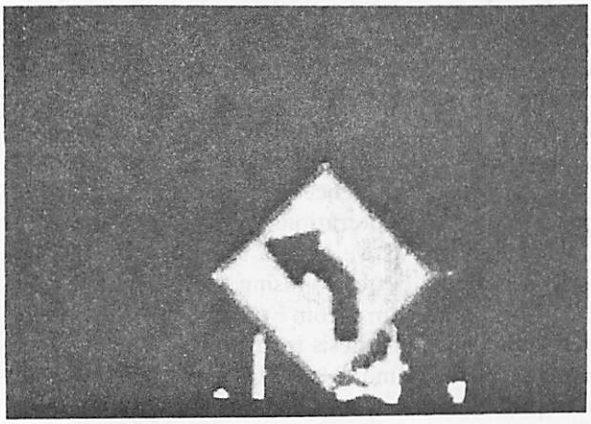
DEPTH ESTIMATION

Just as stereopsis uses the disparity of view between two eyes separated in space to provide depth cues, so the optic flow between two views separated in time provides depth cues. In stereopsis, the depth cues are scaled by the spatial separation of the eyes; in optic flow, the depth cues are scaled by the temporal separation of the frames. Depth is thus in terms of time-until-adjacency rather than centimeters, a very useful form for the control of behavior. Figure 7(d) demonstrates how the length of the optic flow vector provides a depth cue – for a given relative velocity, the farther away the object, the smaller the optic flow vector.

With this background, we turn to the integrated computation of optic flow and focus of expansion, with consequent determination of a depth map.[10] Figure 9 shows successive images taken from a car moving down a country road in Massachusetts. The camera was stabilized with a gyroscope so that the motion was basically a translation with respect to a stationary background. Processing involved



(*a*)



(*b*)

Figure 9. (*a*) and (*b*) show two successive views taken from a moving car.

determining three things: first, the diretion of sensor translation; second, the displacements of image features over the successive images; and thirdly, the extraction of the relative environmental depths of the corresponding image points. In the processing described here, all three factors are determined simultaneously, using the strong constraints that each type of information supplies. For example, image displacements are constrained to lie along image paths which radiate from a single point, the FOE. In addition, the determined displacements should yield consistent environmental depth inferences that break the image into pieces, or segments, that have related depths.

The particular processing steps are detailed in Figure 10. Figure 10(*a*) shows contours which have been extracted from the initial image of the sequence. These particular contours were found by using *zero-crossing* extraction: convolving the image with a Gaussian–Laplacian mask and then thresholding the resulting image at zero. It should be noted that the technique used here is independent of the type of contour extraction used, and, in fact, the resulting flow and depth inferences can be used to evaluate particular and simultaneous segmentations formed by any of several means.

The image points along the contours are then processed to extract interesting points along them. An interesting point is basically one which is different or distinguished with respect to the image positions surrounding it. This distinctiveness is important for motion analysis because it limits the potential displacements of an image feature over successive images. The interest operator applied here basically finds points of high curvature along the extracted contours, and this is shown in Figures 10(*b*) and (*c*). This processing involves finding interesting points along the contour and then filtering, based upon curvature estimates along the contours. Other processes we have developed combine the steps shown here into a single procedure which is applied locally over an image.

The determination of the translational axis and the image displacements of the extracted image features is performed by a simple optimization process. For a particular translational axis there is a corresponding FOE in the image plane. It constrains the image displacements of the extracted image points to lie along the paths which radiate from the FOE. We then express the value of a particular translational axis as the sum of the best match values, which have been normalized, that each feature can find along the image path determined by the corresponding translational axis. This optimization measure has been found to be very well behaved in all image sequences we have investigated: it is smooth, with a single maximum in a very large neighborhood surrounding the correct translational axis. Because of this, the optimization can be quite simple and rapid, especially when expressed hierarchically.

How can this procedure be extended to more general motions? One technique is based upon the fact that the translational processing procedure can be robustly applied to small image areas containing a few features. By applying this procedure to small image areas across an image surface, an intermediate description of the environmental motion results, which associates a direction of environmental motion with the small image areas. This is called the *local translational decomposition*, and it can be used to simplify significantly the processing of unrestricted, and potentially nonrigid, motions. In addition, the interpretation of local image motion as being produced by translations of the corresponding parts of the environment provides a powerful heuristic, related somewhat to the smoothness constraint discussed earlier, for consistently determining image displacements.

We are also considering hybrid sensor systems consisting of image processing and other sensor systems for determining the rotational parameters of sensor motion to deal with general motions. Conventionally, expensive gimbal-gyroscope systems are used to do this. However, there is the very exciting possibility of using optic fiber rotation sensors in the near future to do this.[12] These have several desirable qualities: they are small (currently less than 10 cubic centimeters), cheap, and robust. Some designs bear a striking resemblance, perhaps not coincidentally,

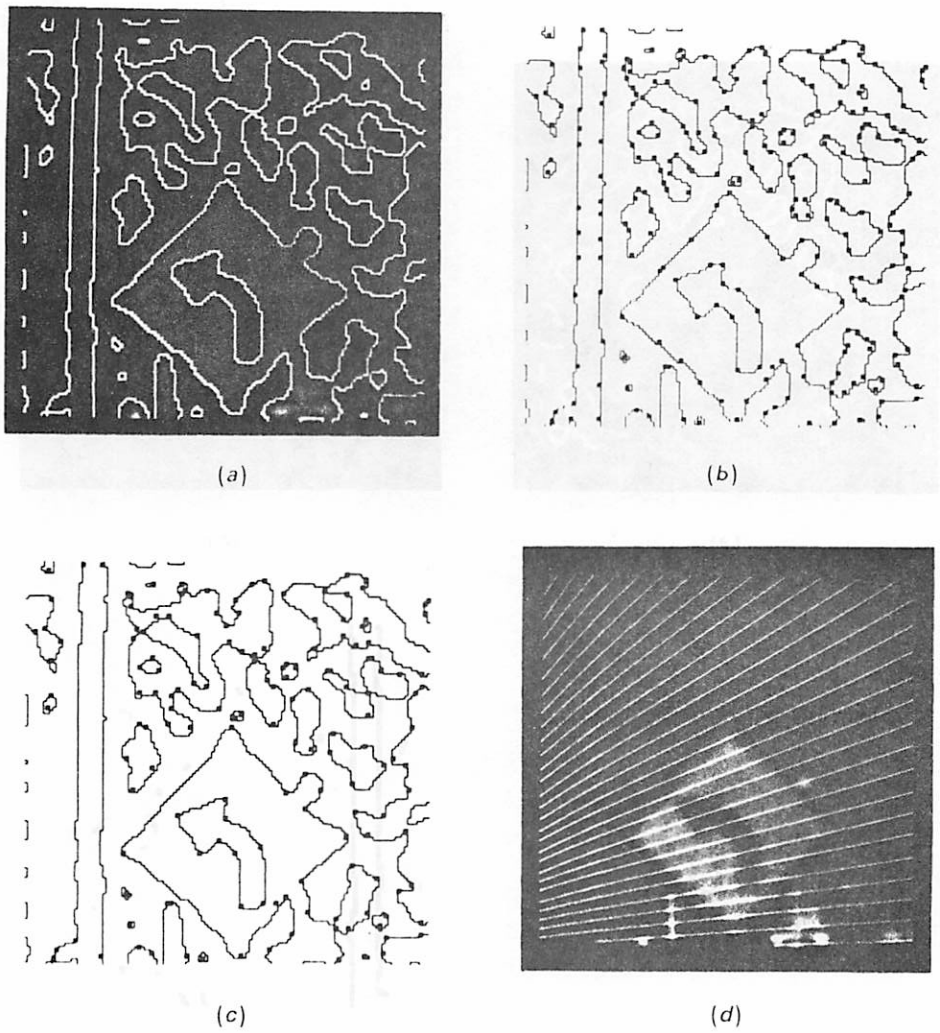


Figure 10. (*a*) shows contours extracted from the image of Figure 9(*a*) by using zero-crossings. (*b*) and (*c*) show the stages in extracting high-curvature points in the image, as targets for matching in forming the optic flow field. (*d*) shows the rays which contain the optic flow vectors for the estimate computed for the optic flow field.

to the semicircular canals of animal vestibular systems.

Returning to translational motions: Given the FOE, it is possible to attribute depth values to particular image points, such as those along the contours determined by some segmentation process. For the image points along the extracted contour in Figure 10(*a*), the determined depth values of Figure 11(*a*) are represented as a histogram in Figure 11(*b*). Note the three distinct peaks which correspond to the three distinct environmental objects in the scene: the sign, the pole, and – over a wider range of depth values – the trees. These peaks in the histogram can be recognized by some simple procedures and the corresponding clusters mapped back onto particular image points. This will produce a segmentation based upon environmental depth, and the image contours corresponding to the extracted peaks are shown in Figures 11(*c*)–(*e*).

Note that the contours are now broken where environmental depth discontinuities occur, and approximate, but do not completely match, the contours of the distinct objects in the scene. Further refinement requires algorithms which embody important new ideas about perceptual processing. For example, after the depth map has been used to divide the image into regions with a consistent estimate of depth, high-level knowledge might then be invoked to yield a clean segmentation of the image into separate objects. But rather than show such processes on the present data, we turn in the next section to results on outdoor scene recognition obtained as part of the VISIONS project developed by Hanson and Riseman.[13] First, however, we briefly discuss how FOE processing is used in the robotics domain.

Figures 12(*a*) and (*b*) show successive images of some gear parts and a transformer on a table as it is approached by a camera held in a robot manipulator. These images are superimposed in Figure 12(*c*). Processing was performed by thresholding the successive images at regular intervals to form a set of binary images. Points along the contours of the binary images formed from the initial image were matched into the points along the contours of the corresponding binary

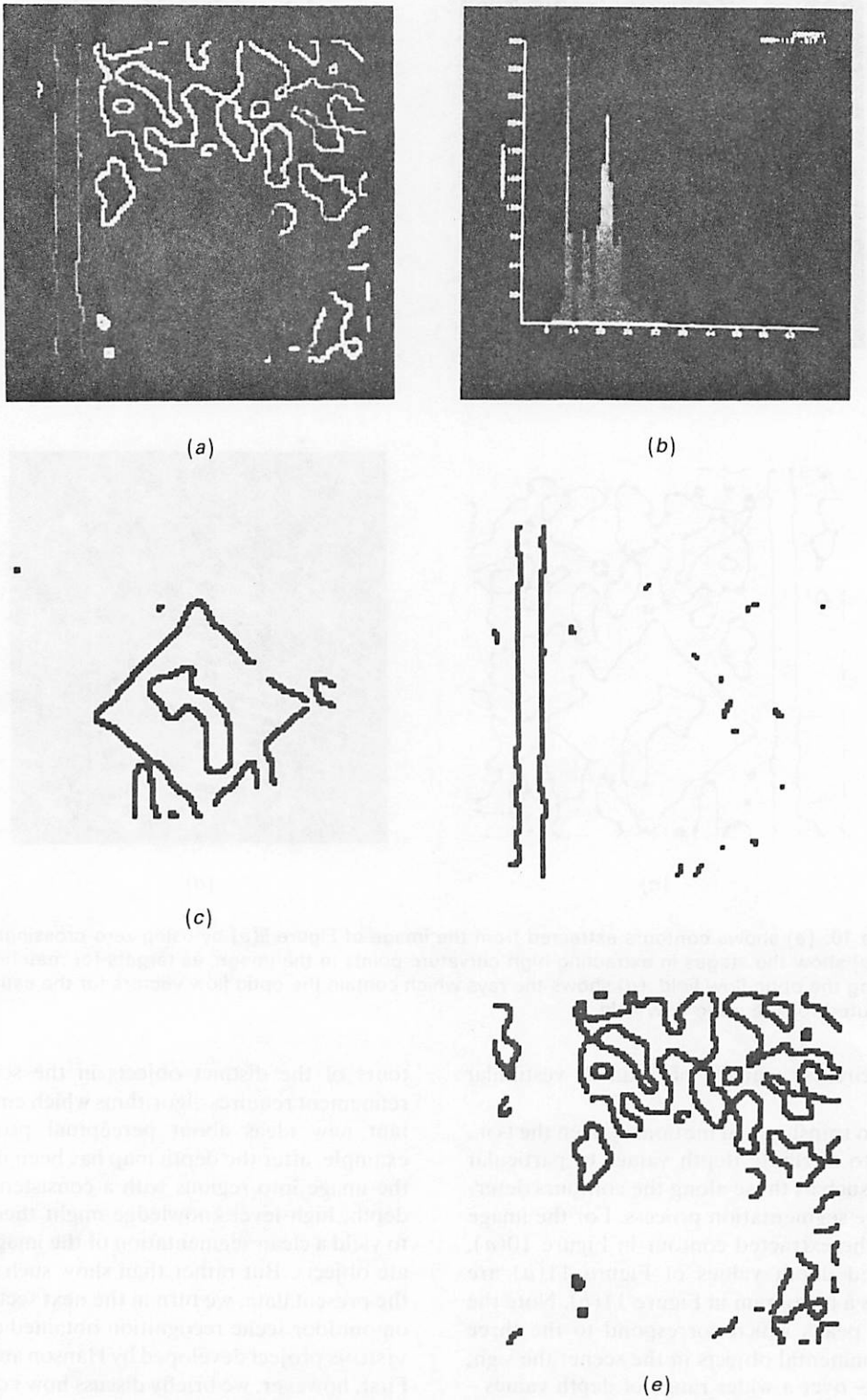


Figure 11. The depth map obtained from the optic flow field for the images of Figure 10(*a*) and (*b*) is shown in (*a*), with brighter points being farther away. These depths are histogrammed in (*b*), and we divide the depths into three clusters based on the peaks. We can then segment the images into three pieces (*c*), (*d*) and (*e*) on the basis of these labels.

image, formed at the same threshold, in the succeeding image. This matching was done by conventional correlation matching techniques and was quite rapid since processing involved binary images.

There is an implicit assumption here that the successive images do not change greatly over small periods of time corresponding to the sampling period during motion. (Alternatively, the contours could be extracted by techniques which are less sensitive than thresholding to global lighting changes.) What is produced is a large number of matches which are cheap to compute. These matches are then filtered by using

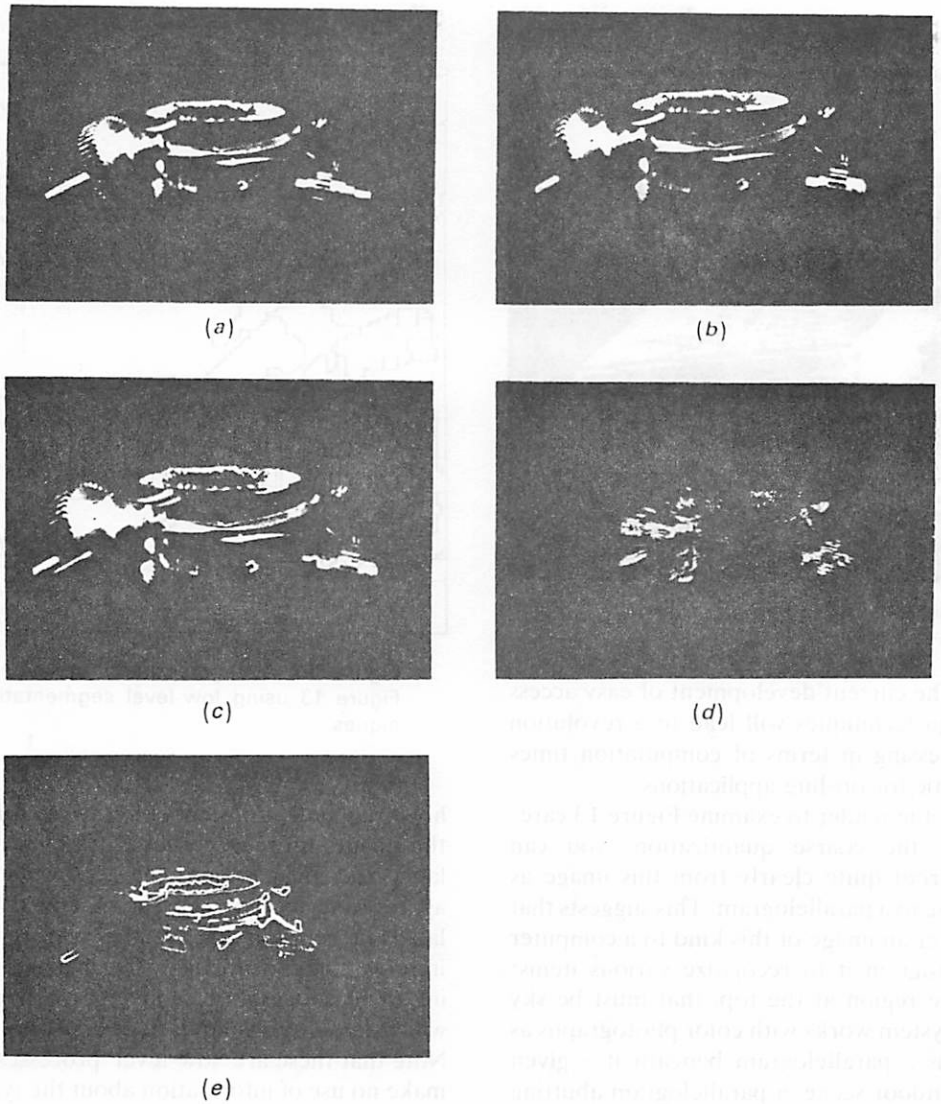


Figure 12. (*a*) and (*b*) show two successive views from a robot manipulator-borne camera moving relative to an assembly line. (*c*) shows the problem of stimulus-matching for these images. (*d*) exhibits the FOE and flow paths found for these images, while (*e*) exhibits the resultant depth map.

a smoothness constraint: displacements which are highly variant with the displacements computed in their surrounding neighborhood are suppressed as being unreliable. The remaining displacements are then used to determine a FOE, in this case by applying a least-squares extraction of a common intersection point. This, along with the determined FOE and corresponding flow paths, is shown in Figure 12(*d*), while the resultant depth map – the brighter, the farther away – is shown in Figure 12(*e*).

INTERACTION OF LOW-LEVEL PROCESSES AND HIGH-LEVEL KNOWLEDGE IN VISUAL SYSTEMS

This final section presents some general observations about cooperative computation by discussing the way in which multiple processes are orchestrated in successful visual perception.

Figure 13 shows a low-resolution image of a house. It is an interesting feature of the stage of our computer technology that it forces us to deal with poor images. The problem is this: a finer image is much easier for a human to recognize, but if we are processing the image on a serial computer, the difference between 128×128 and 1024×1024 can be decisive in terms of the computer time required. Current research is exploring how to take the algorithms that require several minutes of CPU time on a VAX computer and redesign them to be implemented in a fraction of a second on an array of parallel processors, built from VLSI chips.

Our vision problems are currently harder than they need be, because we cannot afford enough time to work serially with the high-resolution images that provide the extra information that aids human

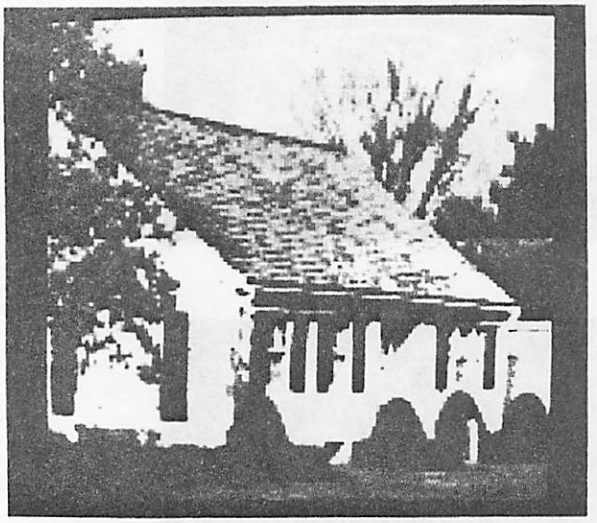


Figure 13. An image of a house.

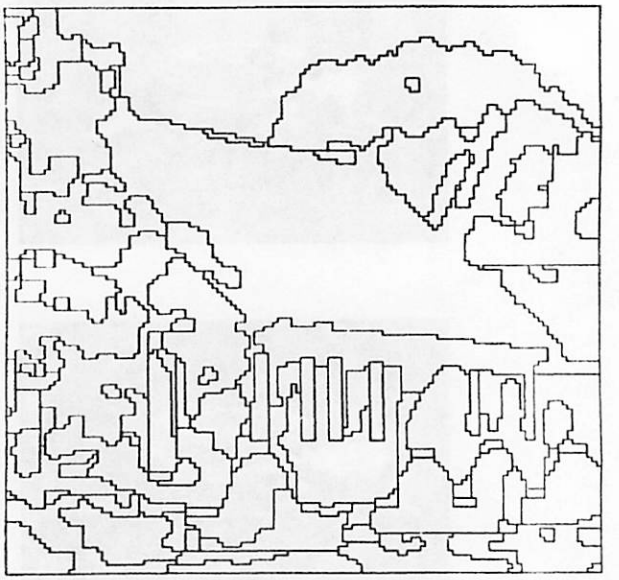


Figure 14. A segmentation of the image of Figure 13 using low-level segmentation techniques.

recognition. The current development of easy access to VLSI design techniques will lead to a revolution in vision processing in terms of computation times that are realistic for on-line applications.

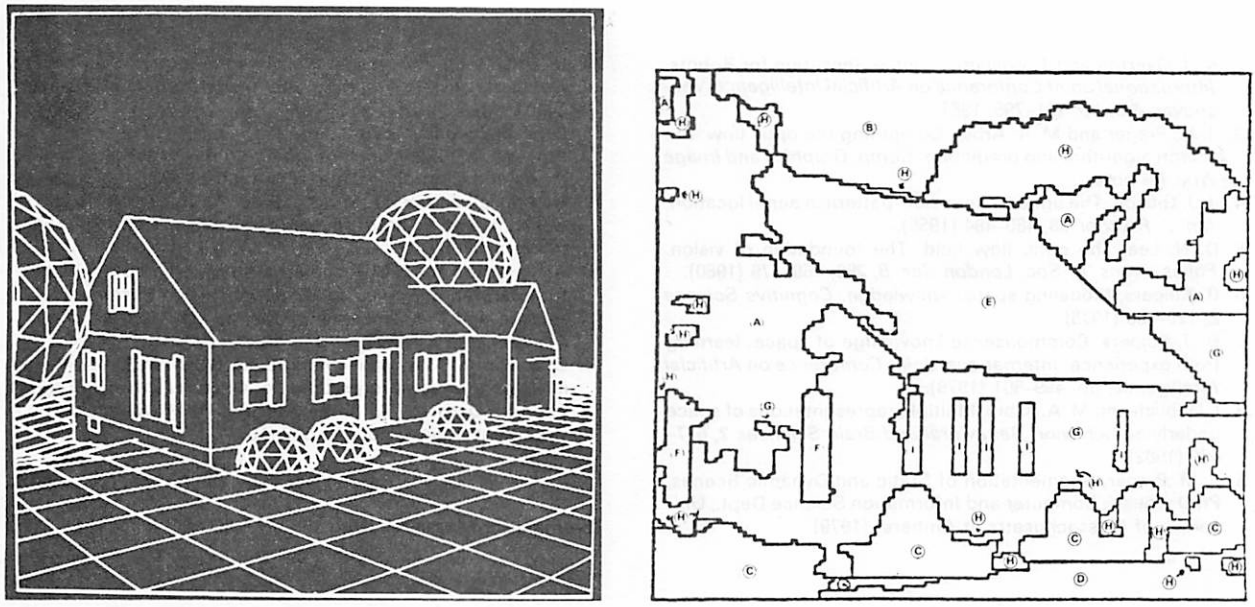
We now ask the reader to examine Figure 13 carefully. Despite the coarse quantization, you can delineate the roof quite clearly from this image as something close to a parallelogram. This suggests that we could deliver an image of this kind to a computer system and program it to recognize various items: 'Here is a blue region at the top, that must be sky (the VISIONS system works with color photographs as input). Here is a parallelogram beneath it – given that it is an outdoor scene, a parallelogram abutting the sky is probably a roof. Beneath the roof are some rectangles – such rectangles are either windows or shutters or doors. . .' and so the process could continue, calling upon high-level information, to classify all portions of the image.

However, the above account was designed to lull you into a false sense of security. To see this, try to find the parallelogram in Figure 13. We can design algorithms that will smooth over the quantization and report with some confidence the lines at the top and bottom of the roof. The left-hand side, however, will not be well defined, for an algorithm using some measure of gradient in color or texture would not return the roof-edge, but the contour where the foliage occludes the roof.

The problem is even worse for an algorithm tracing a gradient of color and texture along the right-hand side of the roof. Halfway up, the gradient wanders off into the sky, eventually coming back and going off again, to yield a bizarre shape that is nothing like a parallelogram because of occlusion and because of spectral problems which wash out boundaries in the image.

The VISIONS system does indeed include tools for image segmentation, in terms of both joining points of similar gradient to grow boundaries, and region-growing algorithms. These produce a histogram of how frequently different colors or textures appear in the image, forming clusters, using cluster labels to label the image, and then growing regions by aggregating locales well characterized by the same label, or constellation of labels. Moreover, we can improve the performance of such algorithms by having them cooperate to converge on a segmentation which integrates boundary and region information. Note that these are 'low-level' processes in that they make no use of information about the type of objects that may occur in the scene. Figure 14 shows the result of running such algorithms on the image of Figure 13. We do indeed see that the top and bottom of the roof are fairly well delineated, but that the left-hand edge is occluded while the right-hand edge bleeds into the sky and foliage. The figure also illustrates that highlighting or variation in texture may lead the algorithm to subdivide a natural region into several segments, as we can see for several of the shutters.

The problem, then, is to design algorithms that on the one hand, can take a region and split it into parts that are to be seen as giving us information about different objects, and on the other hand, will aggregate regions that together characterize some distinctive portion of the image. The process of *image interpretation* calls on high-level information about possible objects in the scene. For example, information about houses would, among other things, initiate a search for a parallelogram. However, the program would not fail if there were no parallelogram in the image, but might pursue more subtle possibilities, as for example: 'if you find two approximately parallel lines at top and bottom and portions of approximately parallel lines on the left and right, join up the lines, and explore the hypothesis that the resultant parallelogram is a roof.' Given a confident roof



(*a*) (*b*)

Figure 15. (*a*) A 2D projection of a 3D schema of a house scene. (*b*) By applying information in the 3D schema to the segmentation of Figure 14, the segmentation can be improved, and most of the regions labeled.

hypothesis, the system can hypothesize that below the roof the image will contain shutters or windows. Thus if regions there can be aggregated into a rectangle, the program can indeed follow the hypothesis that there is a rectangle.

How then is high-level information to be represented without reducing the problem to the simple problem of template-matching, as in the hole-finder described earlier? One approach used in preliminary studies is to provide a three-dimensional schema which gives confidence ranges for the location of different objects in space.[14] Figure 15(*a*) shows a two-dimensional view of such information. A sphere is not to be thought of as a veridical representation of a tree, but rather gives a 90% confidence interval for where tree foliage is located. Here we have gone from a specific tree in a specific position to a stochastic tree. Part of our research is to be able to look at more and more general characterizations. We want more general ways of representation which will allow us to recognize very different types of trees or houses. One question is to what extent one can recognize a house from one very general template, and to what extent it is one's experience of seeing many kinds of houses that allows one to use the appropriate form from occasion to occasion. It is beyond the scope of this review to detail the way the information is applied, but Figure 15(*b*) shows what the system finally produces for the segmentation of Figure 14. The roof is clearly delineated, shutter subregions are aggregated, and most of the image has been correctly characterized.

The overall computation, then involves the interleaving of multiple processes, a cooperative computation in which each process is invoked where appropriate, possibly many times, with hypotheses being generated and discarded until the system converges on as good an interpretation as it is able to give with the facilities available to it. We claim that this style characterizes the perceptual mechanisms of brains, and will provide the style for future developments integrating perceptual systems for robot control.

Acknowledgement

This review is an exposition of directions of research, as of March 1982, at the University of Massachusetts at Amherst, supported in part by NIH grant NS14971-04, NSF grant ECS-8108818, and DARPA grant N00014-82-K-0464. Versions of this review were presented by Michael Arbib during a lecture tour of Australia for the Australian Computer Society and the Australian Institute of Physics, 26 April to 7 May 1982, and at the Australian Academy of Sciences Symposium on 'Patterns of Perception,' Canberra, 30 April 1982. The review is, with the permission of the editors, being published both in *The Australian Computer Bulletin* as a record of the lecture tour and in *Interdisciplinary Science Reviews* as part of the record of the symposium.

LITERATURE CITED

1. M. A. Arbib, Perceptual structures and distributed motor control. In *Handbook of Physiology: The Nervous System, II. Motor Control* (V. B. Brooks, Ed.), Bethesda, Md: American Physiological Society, pp. 1449–1480 (1981).

2. K. J. Overton and T. Williams, Tactile Sensation for Robots. *International Joint Conference on Artificial Intelligence*, Vancouver, BC, pp. 791–795. 1981.
3. J. M. Prager and M. A. Arbib, Computing the optic flow: the MATCH algorithm and prediction. *Comp. Graphics and Image Proc.* (in press).
4. J. J. Gibson, The optical expansion-pattern in aerial location. *Am. J. Psychol.* **68**, 480–484 (1955).
5. D. N. Lee, The optic flow field: The foundation of vision. *Philos. Trans. R. Soc. London Ser. B*, **290**, 169–179 (1980).
6. B. Kuipers, Modeling spatial knowledge. *Cognitive Science* **2**, 129–153 (1978).
7. B. J. Kuipers, Commonsense knowledge of space: learning from experience. *International Joint Conference on Artificial Intelligence*, pp. 499–501 (1979).
8. I. Lieblich and M. A. Arbib, Multiple representations of space underlying behavior. *Behavioral and Brain Sciences* **2**, 627–659 (1982).
9. J. M. Prager, Segmentation of Static and Dynamic Scenes, Ph.D. Thesis, Computer and Information Science Dept., University of Massachusetts at Amherst (1979).
10. D. T. Lawton, Processing translational motion sequences. *Computer Vision, Graphics, and Image Processing* **22**, 116–144 (1983).
11. D. T. Lawton, Motion Analysis via Local Translational Processing. IEEE Workshop on Computer Vision: Representation and Control, Rindge, NH, August 1982.
12. S. Ezekiel and H. J. Arditty (Eds.), *Fiber-Optic Rotation Sensors and Related Technologies*. New York: Springer-Verlag (1982).
13. A. R. Hanson and E. M. Riseman, VISIONS: a computer system for interpreting scenes. In *Computer Vision Systems* (A. R. Hanson and E. M. Riseman, Eds.), pp. 129–163, New York: Academic Press (1978).
14. C. C. Parma, A. R. Hanson and E. M. Riseman, Experiments in schema-driven interpretation of a natural scene. In *Digital Image Processing* (J. C. Simon and R. M. Haralick, Eds.) Dordrecht: Reidel, pp. 449–509 (1981).

The manuscript was received 15 July 1983