A Parallel Noise-Robust Algorithm to Recover Depth Information from Radial Flow Fields

F. Wörgötter
A. Cozzi
Department of Neurophysiology, Ruhr-Universität, 44780 Bochum, Germany

V. Gerdes
Institut für Informatik III, Universität Bonn, Bonn, Germany

A parallel algorithm operating on the units ("neurons") of an artificial retina is proposed to recover depth information in a visual scene from radial flow fields induced by ego motion along a given axis. The system consists of up to 600 radii with fewer than 65 radially arranged neurons on each radius. Neurons are connected only to their nearest neighbors, and they are excited as soon as a sufficiently strong gray-level change occurs. The time difference of two subsequently activated neurons is then used by the last-excited neuron to compute the depth information. All algorithmic calculations remain strictly local, and information is exchanged only between adjacent active neurons (except for the final read-out). This, in principle, permits parallel implementation. Furthermore, it is demonstrated that the calculation of the object coordinates requires only a single multiplication with a constant, which is dependent on only the retinal position of the active neuron. The initial restriction to local operations makes the algorithm very noise sensitive. In order to solve this problem, a prediction mechanism is introduced. After an object coordinate has been determined, the active neuron computes the time when the next neuronal excitation should take place. This estimated time is transferred to the respective next neuron, which will wait for this excitation only within a certain time window. If the excitation fails to arrive within this window, the previously computed object coordinate is regarded as noisy and discarded. We will show that this predictive mechanism relies also on only a (second) single multiplication with another neuron-dependent constant. Thus, computational complexity remains low, and noisy depth coordinates are efficiently eliminated. Thus, the algorithm is very fast and operates in real time on 128×128 images even in a serial implementation on a relatively slow computer. The algorithm is tested on scenes of growing complexity, and a detailed error analysis is provided showing that the depth error remains very low in most cases. A comparison to standard flow-field analysis shows that our algorithm outperforms the

older method by far. The analysis of the algorithm also shows that it is generally applicable despite its restrictions, because it is fast and accurate enough such that a complete depth percept can be composed from radial flow field segments. Finally, we suggest how to generalize the algorithm, waiving the restriction of radial flow.

1 Introduction

During the projection of the three-dimensional environment onto the two-dimensional receptor surfaces of the eyes, depth information is lost. Several ways exist for recovering depth from these projection images. Many biological and technical systems rely on the analysis of stereo image pairs. In these systems, depth information is retrieved from the analysis of the local image differences between the left and the right image (called disparities), which result from the lateral displacement of the two eyes or cameras (e.g., correlation-based methods: Marr & Poggio, 1976; phase-based methods: Sanger, 1988; Fleet, Jepson, & Jenkin, 1991; for a review of the older work, see Poggio & Poggio, 1984; a recent review is by Qian, 1997). If the viewer or the objects are moving, the motion pattern can be analyzed instead in order to obtain depth information (Ullman, 1979; Prazdny, 1980; Longuet-Higgins & Prazdny, 1980; Lucas & Kanade, 1981; Fennema & Thompson, 1979; Heeger, 1988; Fleet & Jepson, 1990). Ego motion or object motion generates a so-called flow field on the receptor surfaces (see, e.g., Horn & Schunck, 1981; Koenderink, 1986; Barron, Beauchemin, & Fleet, 1994a; Barron, Fleet, & Beauchemin, 1994b). The projection of the displaced objects thereby consists of curves of various shapes. In the most general case (object plus ego motion) the curved flow-field patterns cannot be resolved for depth analysis without additional assumptions (rigidity and smoothness constraints; Poggio, Torre, & Koch, 1985; Hildreth & Koch, 1987; Yuille & Ullman, 1987). However, even if simplifying assumptions are made, the problem of structure from motion remains rather complex.

The goal of this study is to devise a neuronal algorithm that allows the analysis of radial (diverging) flow fields by the parallel operation of its individual photoreceptive sites (its "neurons"). We will show first that depth information is obtained by a single scalar multiplication with a neuron-dependent constant at each active neuron. Thus, the algorithm is very simple and so fast that it operates in real time even in our serial computer simulations. The structure of our network is such that all computations remain local, and neurons need "to talk" only to their nearest neighbors, which permits parallelization. The locality of all calculations, however, makes the algorithm initially very noise sensitive. Therefore we will show, second, that the algorithm can be extended by a local predictive mechanism that relies on the propagation of the excitation pattern one step into the network. Prediction of the future excitation pattern requires only one more scalar multiplication at each active neuron. Thus, the computational complexity remains low. As a result we will show that this local predictive mechanism almost completely eliminates noise and other errors in the analysis.

The restriction to radial flow finds its motivation in the behavior of animals. In different species, varying strategies are observed in order to reduce the optic flow as much as possible to a few or, if possible, a single component. For the housefly (*Musca domestica*), Wagner (1986, p. 546) stated: "Thus, the flight behavior and the coordination of head and body movements may be interpreted as an active reduction of the image flow to its translational components." Ideally this would mean that only forward motion exists and that optic flow is reduced to its radial component. In the fly, this actually leads to the tendency to fly along straight lines, making rather sharp turns when changing direction (Wagner, 1986). Flow-field reduction is pushed to an extreme in some birds while they are walking. The intriguing head bobbing of pigeons serves the purpose of eliminating all optic flow while the bird moves its body forward "under" the motionless head (Davies & Green, 1988; Erichsen, Hodos, Evinger, Bessette, & Phillips, 1989). Similarly it has been observed that pigeons and other birds keep their head stable during different flight maneuvers, such that the head pursues a smooth-motion trajectory while the body can make rather jerky movements (Green, Davies, & Thorpe, 1992; Davies & Green, 1990; Erichsen et al., 1989; Wallman & Letelier, 1993). In particular, when very high accuracy is required during landing, the compensatory head movements become very pronounced and accurate such that relatively undisturbed radial flow is obtained (analysis of high-speed video data of free-flying pigeons by J. Ostheim, personal communication). Given the complexity of insect or bird flight, the reduction of the optic flow must remain incomplete; the strategy to reduce the computational complexity of flow-field analysis, however, seems to be pursued widely, even in mammals (e.g., component specificity of MST cells, Duffy & Wurtz, 1991, 1995; Graziano, Anderson, & Snowden, 1994; see also Lappe, Bremmer, Pekel, Thiele, & Hoffmann, 1996; Wang, 1996, for theoretical approaches on medial superior temporal area cells).

Under the assumption that flow-field restriction is a biologically justified strategy, the central goal of our study is to arrive at an efficient and noise-robust algorithm that can operate in parallel on a restricted flow field, thereby making use of a task-dedicated artificial neural net architecture. Although the initial motivation comes from biology, it is obvious that the algorithmic transfer of the underlying concept into a more technical domain immediately imposes restrictions with respect to the biological realism of the network.

We will describe the algorithm and present results from the analysis of artificial and real image sequences, which demonstrate that depth information is retrieved with very high accuracy. A rather technical appendix provides a detailed error analysis that demonstrates that the algorithm is generally applicable. (This appendix is mainly of relevance for those who wish to implement this algorithm. It may be skipped otherwise.)

2 Description of the Algorithm

The core part of the algorithm consists of a parallel[1] operating network of neurons, called the "retina" (see Figure 1), with which we are mainly concerned. In order to describe the algorithm, we assume a moving robot driven by a stepper motor and a visual system consisting of one camera with its camera plane orthogonal to the axis of motion of the robot.

Two restrictions are introduced:

1. The robot is assumed to move only along the optical axis of the camera.
2. The environment is regarded as stationary (i.e., no moving objects).

The first restriction leads to a purely radial flow field on the camera plane, and this condition seems fatally strong, limiting the algorithm to a special case that exists only during short intervals of robot motion. In particular, during a curve, the focus of expansion is no longer aligned with the motion trajectory, rendering the algorithm useless. Initially the restriction to radial flow was biologically motivated. However, it will almost always be sufficient to approximate the complete "depth percept" by such linear motion segments provided the robot makes rather sharp turns, (similar to the flying pattern of a fly) during which it is "blind." The high camera frame rates and the speed of the algorithm ensure that a novel depth percept builds up rather fast after a turn, such that periods of "robot blindness" remain short. The second restriction can also be partly waived, as explained in the discussion.

2.1 The Retina. The retina consists of radially arranged neurons that are connected only to their nearest neighbors in both directions on the same radius.

In a radial flow field, the virtual speed of the projected objects on the retina increases with the distance from the optical axis, and the retina location of the projected image for the geometrical camera arrangement shown in Figure 3 is inversely proportional to the distance of the object from the nodal point (conventional hyperbolic projection geometry). The goal now is to design an arrangement that takes care of this projection geometry and allows for a uniform sampling of the scene along the locations on the radii during radial flow.

We can restrict ourselves to a single radius and define the neuronal density by the hyperbola

$$D(r_n) = \frac{1}{r_n - r_{n-1}} \qquad n \in [1, \ldots, N], \tag{2.1}$$

[1] "Parallel" means that this structure can *in principle* be implemented and operate in parallel. All computational results shown in this study, however, are based on regular workstations, such that all computations are performed serially.

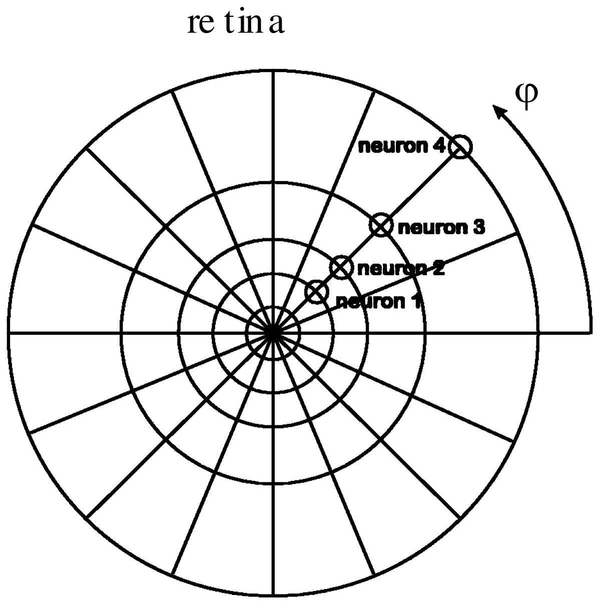

Figure 1: Layout of the retina. Neurons are arranged according to equation 2.4.

where r_n is the location of neuron n on a retinal radius and $N + 1$ the total neuron number on every radius.

As a consequence of the hyperbolic projection geometry, we find that the neuronal density D at a given retinal location x should be directly proportional to $1/x$. Since we deal with a discrete neuronal placement problem, this reads:

$$D(r_n) \sim \frac{1}{n}. \tag{2.2}$$

This requirement is fulfilled for the following definition of r_n:

$$r_n = \frac{1}{2}kn^2 + \frac{1}{2}kn + r_0, \tag{2.3}$$

because in this case we get

$$D(r_n) = \frac{1}{r_n - r_{n-1}} = \frac{1}{kn}.$$

Let ρ be the radius of the retina. Then we place one neuron in the center of the retina (i.e., $r_0 = 0$) and one on its border ($r_N = \rho$) and get:

$$r_n = \frac{\rho}{N(N+1)} \cdot n(n+1) = h \cdot n(n+1). \tag{2.4}$$

A VLSI design that basically emulates this structure already exists, but without the required connections between the neurons (Pardo, 1994).

2.2 Flow Diagram of the Algorithm. The individual neurons are designed to perform only very simple operations: reading, comparing, and storing gray-level values; reading the stepper motor count of the robot, computing (two) scalar multiplications; raising or lowering a counter; and transferring the output as specified below. Thus, they contain a "memory" and a simple processing capability.

For now, we assume an ideal situation, which consists of noise-free gray-scale images. We say that a neuron is excited as soon as the luminance at this neuron changes significantly. In order to explain the algorithm, let us further restrict the situation to a robot that moves in an environment with only a single black-dot object somewhere in the distance.

Before the robot starts to move, all neurons will be reset and their memory deleted. At time-step t_1 the black dot will excite neuron 1 (see Figure 2A). Since its memory does not contain any information, the neuron will transfer only the gray-level value ("black") to the next outer neuron (neuron 2). After some time, the projection of the black dot will have traveled to neuron 2 (see Figure 2B). This neuron compares the newly read gray-level value with the one stored in its memory and finds that they are similar within a reasonable range. It will then read the stepper motor count ($\Delta Z = 8$; see Figure 2B) and compute the cylinder coordinates of the object $(R_1, Z_1)_\varphi$. In addition it will assign a label—say $L = \alpha$—to this object (see Figure 2B). From the coordinates and the known motion pattern of the robot, neuron 2 can then also compute the stepper motor count, which will be expected at the moment when neuron 3 will be excited ($\Delta Z_p = 6$; see Figure 2B). Neuron 2 will transfer the predicted value (ΔZ_p), the gray-scale value ("black"), and the label (α) to neuron 3. The coordinates $[(R_1, Z_1)_\varphi]$ of the object, as well as the label (α), will be read by the common read-out to generate the depth map. As soon as neuron 3 is excited (see Figure 2C), it compares the gray-level values; after having found a match, it also compares the predicted and the actual stepper motor count. If they match within a certain tolerance (e.g., $\Delta Z = \Delta Z_p \pm 1$), the object is regarded as confirmed and the confirmed counter (C; see Figure 2C) is increased. In this way, object positions become more reliable, the detection error is reduced, and false object positions are soon rejected. The object coordinates will be recomputed $[(R_2, Z_2)_\varphi]$, and the object position with label α in the depth map will be updated. In the case that the reconfirm failed (e.g., $\mid \Delta Z - \Delta Z_p \mid > 1$) the object is considered new, and a new label is assigned to it. In this case, the old object with label

α is regarded as unreliable and removed from the depth map. If an object has already been confirmed several times, it will not be directly eliminated from the depth map, but the confirmed variable will be lowered gradually until it reaches zero ("slow death").[2]

2.3 Equations. Figure 3 shows the geometrical situation for which the equations are defined. Most equations are defined in cylinder coordinates $[(R, Z)_\varphi]$, and only at the end will we give the final result in Euclidean coordinates (X, Y, Z). We will first describe how to obtain the object coordinates (R_n, Z_n) from the excitations of the neurons and then compute the prediction value ΔZ_p for the next expected excitation occurring at (R_{n+1}, Z_{n+1}). In addition, at first we will use vector notation ($\overrightarrow{s}$), which does not impose any restrictions on the geometry, and only later include the already described retinal neuron arrangement.

To get the object position, we have to solve the following equation by eliminating k and l:

$$k \cdot \overrightarrow{s_n} + \overrightarrow{\Delta Z} = l \cdot \overrightarrow{s_{n-1}}. \tag{2.5}$$

Since we assume that the robot motion contains only a Z-component, we get:

$$k \cdot \begin{pmatrix} s_{n,r} \\ s_{n,z} \end{pmatrix}_\varphi + \begin{pmatrix} 0 \\ \Delta Z \end{pmatrix}_\varphi = l \cdot \begin{pmatrix} s_{n-1,r} \\ s_{n-1,z} \end{pmatrix}_\varphi . \tag{2.6}$$

Note that the angular component φ of the cylinder coordinates is defined by the angle of the neuron chain on the retina. Thus, for each neuron chain, it is constant. For the radial and the Z-component, this reads:

$$k \cdot s_{n,r} = l \cdot s_{n-1,r}, \qquad \text{and} \qquad k \cdot s_{n,z} + \Delta Z = l \cdot s_{n-1,z}. \tag{2.7}$$

From this we get:

$$k = \frac{\Delta Z}{\frac{s_{n,r}}{s_{n-1,r}} s_{n-1,z} - s_{n,z}}. \tag{2.8}$$

The actual object position can now be computed by:

$$\begin{pmatrix} R \\ Z \end{pmatrix}_\varphi = k \cdot \begin{pmatrix} s_{n,r} \\ s_{n,z} \end{pmatrix}_\varphi = \Delta Z \cdot \overrightarrow{(P_n)_\varphi}. \tag{2.9}$$

[2] In the current implementation of the algorithm, all information exchange remains local and thus restricted to subsequent neurons. In this case, slow death leads only to the prolonged persistence of highly confirmed objects. In a more elaborate version, information transfer could be implemented over more than two neurons, such that Z_p is computed for them. This could better compensate for single misses.

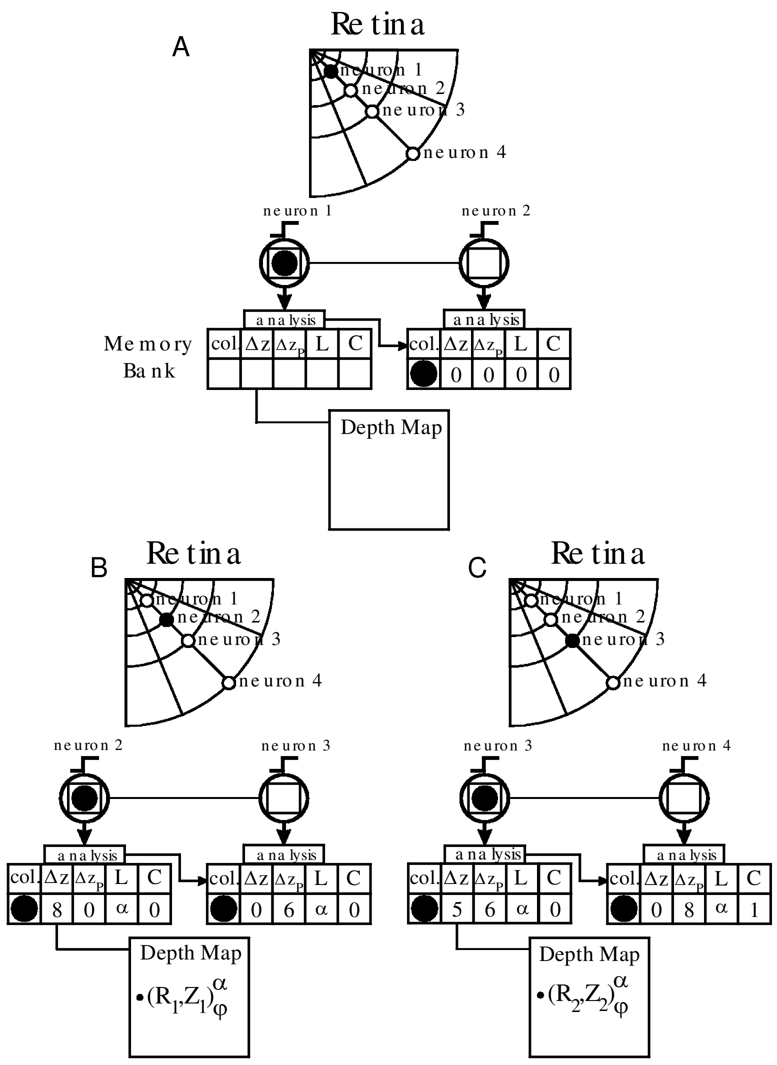

Figure 2: Flow diagram of the algorithm.

So far the equations do not impose any restrictions on the neuron positioning and also leave other geometrical constants open. However, it makes sense to assume that the Z-components of the vectors $\vec{s}$ are identical to the focal length (i.e., $s_{n-1,z} = s_{n,z} = f$). Furthermore from equation 2.4, we get:

$$\frac{s_{n,r}}{s_{n-1,r}} = \frac{r_n}{r_{n-1}} = \frac{n+1}{n-1}. \tag{2.10}$$

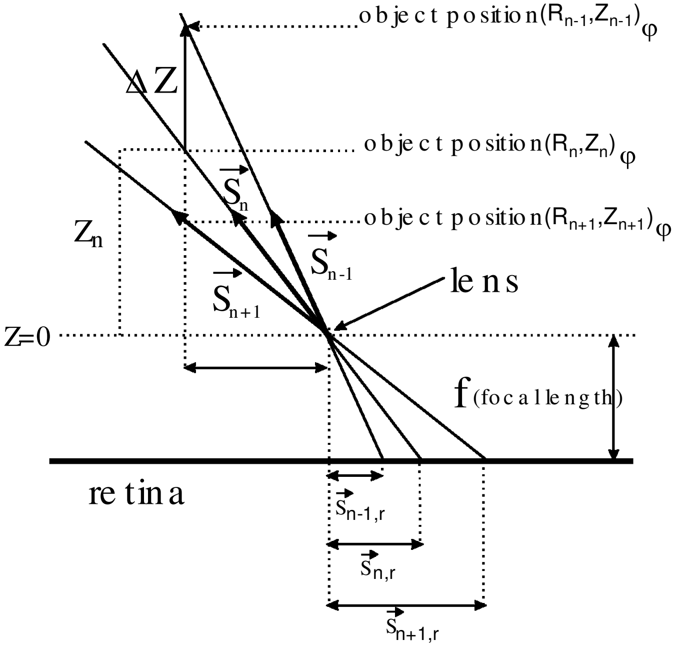
object position(R_{n-1},Z_{n-1})_φ
ΔZ
object position(R_n,Z_n)_φ
S_n
object position(R_{n+1},Z_{n+1})_φ
Z_n
S_{n-1}
lens
S_{n+1}
Z=0
f (focal length)
retina
S_{n-1,r}
S_{n,r}
S_{n+1,r}

Figure 3: Geometry of the projection of a black-dot image onto one radius of the retina. This geometry defines the equations used to compute the object coordinates.

Both assumptions now lead to (for the definition of h see equation 2.4):

$$\overrightarrow{(P_n)_\varphi} = \frac{1}{f\left(\frac{n+1}{n-1}-1\right)} \begin{pmatrix} h\cdot n(n+1) \\ f \end{pmatrix}_\varphi = \begin{pmatrix} \frac{h\cdot n(n-1)(n+1)}{2f} \\ \frac{n-1}{2} \end{pmatrix}_\varphi = \begin{pmatrix} \frac{h\cdot(n^3-1)}{2f} \\ \frac{n-1}{2} \end{pmatrix}_\varphi . \quad (2.11)$$

The general form reads in Euclidean coordinates:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = k\cdot \begin{pmatrix} s_{n,r}\cdot\cos\varphi \\ s_{n,r}\cdot\sin\varphi \\ s_{n,z} \end{pmatrix} = \Delta Z\cdot \overrightarrow{P_n}. \quad (2.12)$$

Again imposing the geometrical restrictions, we get:

$$\overrightarrow{P_n} = \begin{pmatrix} \frac{h\cdot(n^3-1)}{2f}\cdot\cos\varphi \\ \frac{h\cdot(n^3-1)}{2f}\cdot\sin\varphi \\ \frac{n-1}{2} \end{pmatrix}. \quad (2.13)$$

Thus, the object position is obtained from a single multiplication for each component of the stepper motor count ΔZ with $\overrightarrow{P_n}$. It should be noted that $\overrightarrow{P_n}$ [or $\overrightarrow{(P_n)_\varphi}$ in equation 2.11) is constant but different for each neuron. Thus, the multiplication is effectively reduced to a scaling operation with a different scalar factor at each neuron, which is the central feature on this algorithm, making it exceedingly simple.

In the second step, the prediction value ΔZ_p will be computed using the radial component R from the first computation. In the general case we get:

$$\frac{s_{n,z}}{s_{n,r}} = \frac{Z_n}{R} \quad \text{and} \quad \frac{s_{n+1,z}}{s_{n+1,r}} = \frac{Z_{n+1}}{R}. \tag{2.14}$$

Since $\Delta Z_p = Z_n - Z_{n+1}$ we get:

$$\Delta Z_p = R\left(\frac{s_{n,z}}{s_{n,r}} - \frac{s_{n+1,z}}{s_{n+1,r}}\right). \tag{2.15}$$

With the same geometrical restrictions as before, this is:

$$\Delta Z_p = R \cdot \left(\frac{2f}{h \cdot n(n+1)(n+2)}\right). \tag{2.16}$$

The radial component R of the cylinder coordinates has been stored from the calculation of the object position. Therefore, this equation amounts to only a simple scaling operation because everything except R is constant.

2.4 Results.

2.4 Results. We will first show the results obtained with artificial images of increasing realism and then how the algorithm works in a real environment. In the appendix, we present a detailed analysis of the inherent error sources of the algorithm. All results were obtained on a SUN SPARC 10 workstation.

2.4.1 Results on Artificial Images. In a more realistic environment, objects can no longer be described as single dots. Therefore, for the following scenes, we used the color transition that occurs at an edge as the excitation criteria for the neurons. Consequently, all depth maps are defined only at the edges of the objects. We share this characteristic with all depth analysis algorithms that do not introduce additional regularization schemes. Figure 4 shows the results from the analysis of an artificial environment without (A) and with noise (C), which consists of three flat objects of different geometry (triangle, vertical bar, square) located at different distances (4.0 m, 5.5 m, 7.0 m, B) in front of a background 10.5 m away. Quantitative diagrams to supplement the results of Figure 4 are shown in Figure 5. Parameters for this test are given in Table 1.[3] The parameter $\epsilon_{Position}$ indicates the maximal position error (here,

[3] In this section, we will focus on the description of the basic findings; therefore, we refer readers to the appendix for an explanation of some of the non-self-explanatory

Table 1: Parameters for the Simulation Shown in Figure 4.

Parameter	Value	Parameter	Value
Sequence	800 images	Neuron chains	600
Resolution	160 × 150 pixel	Neurons per chain N	≤ 50
Step	5 mm/image	Retina radius ρ	≤ 105 pixel
$\epsilon_{\text{Position}}$	10 mm	$\epsilon_{\text{Displacement}}$	0.05 pixel

1.0 cm) allowed between two subsequent depth estimates arising from two adjacent neurons. If this parameter is exceeded, the second depth estimate is rejected, and the point is not included in the depth map.

Figure 4 shows the changing retinal projection on the left side (column D). The other panels demonstrate how the depth map (columns E, H, I) and the confirmation map (columns F, G) for this scene evolve over the 800 steps of simulated robot movement, equivalent to 4 m of traveled distance. The confirmation maps show the gray-scale-coded value of the confirmed counter (light gray=0, darker gray=1, etc.), whereas the depth maps show all coordinates that had a confirmed value as indicated in the figure. The left side was computed for the noise-free scene; for the right side, 25% of random noise was added to each individual frame.[4] After about 50 cm, the first data points become confirmed more than once, and after 1 m, the outline of all obstacles is clearly visible in the case of no noise. With noise, only a few data points are confirmed twice, but after 1.5 m (not shown) enough data points are reliable to perform for example obstacle avoidance.

The bottom part of the figure (J–Q) shows side-view maps of the obtained depth estimates for different confirmed values after the complete run. The horizontal lines pointing left from the start of the robot motion indicate the distance traveled by the robot. The diagrams show that the depth estimates are very accurate. Pixels overlay each other such that the total number of depth estimates cannot be deduced from the side-view maps (but see the histograms in Figure 5).

In such an artificial situation, even a confirmed value of zero leads to good results if no noise is present. Increasing the confirmed value leads to more rejections of data points and a reduced density of the depth map. With noise, a confirmed value of 2 is a good compromise between the accuracy of the depth estimates and the density of the map.

parameters (e.g., $\epsilon_{\text{Displacement}}$) in Table 1.

[4] The maximally allowed noise amplitude was 25% of the maximal gray-scale difference between the darkest and the brightest pixel in all frames. For every pixel, a random number was drawn (flat distribution) between −12.5% and +12.5%, and this value was added to the pixel value, clipping at 0 and 255 if necessary.

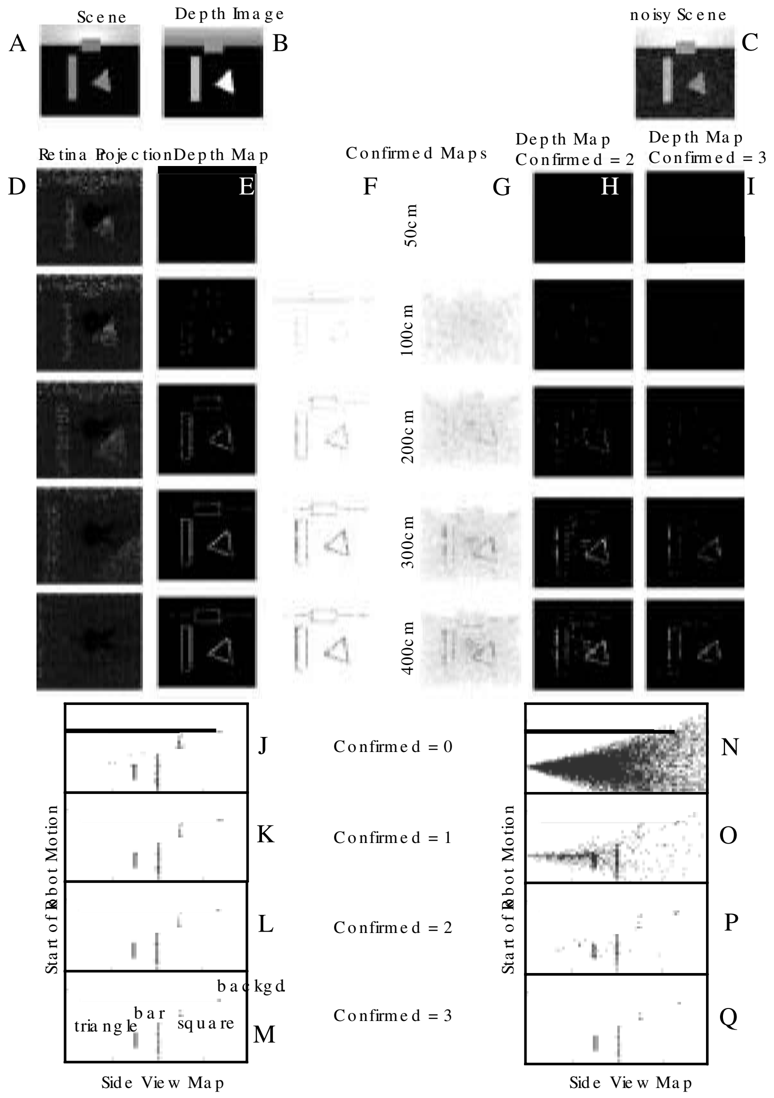

Figure 4: Different stages of the retrieval of depth information from an artificial noise-free (left) and noisy scene (right). Side-view maps are given at the bottom.

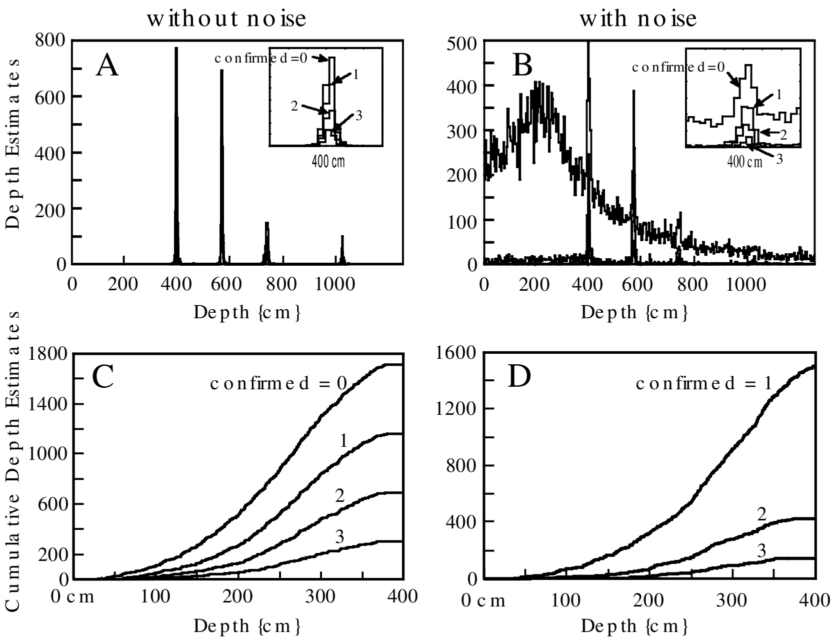

Figure 5: (A, B) Depth histograms for the scene shown in Figure 4. Note four curves overlay each other in A and B, corresponding to the values of *confirmed* = 0, 1, 2, 3. In the insets, the four curves can be discerned. Each peak corresponds to one object. (C, D) Cumulative diagram of the number of depth estimates obtained from the triangle in Figure 4 along the robot motion trajectory. Different diagrams for *confirmed* = 0, 1, 2, 3 are shown in A–D.

To get a better estimate of the quality of the results, we have plotted the detected depth values for the different confirmed values as histograms in Figure 5. The very narrow histograms confirm the high accuracy of the depth map, as already suggested by the side-view maps. Opposite to those, however, the histograms quantify the total number of depth estimates, which decreases with higher confirmed values. To be able to discern this effect, magnifications of the peaks from the "triangle" located at distance 4.0 m are shown in the insets.

A second aspect of interest in this context is how fast a reliable outline of an object is obtained. Parts C and D of Figure 5 demonstrate for different values of the confirmed variable how the depth estimates accumulate along the robot's path toward an object. Even in the noisy scene (D), more than 300 depth estimates are obtained for a confirmed value of 2 until the robot is only 1 m away from the object. This amount of estimates should suffice for most applications.

Table 2: Parameters for the Simulation Shown in Figure 6.

Parameter	Value	**Parameter**	Value	**Parameter**	Value
Sequence	600 images	Neuron chains	400	*Confirmed*	1
Resolution	128 ×120 pixel	Neurons per chain N	≤ 64	$\epsilon_{\text{Position}}$	0.5 cm
Step	0.5 cm/image	Retina radius ρ	≤ 80 pixel	$\epsilon_{\text{Displacement}}$	0.25 pixel

2.4.2 Results in a Seminatural Environment. In the next step we used a scene generated by a ray tracer that resembles the operating environment of an office robot, simulating a hallway with a few obstacles and several light sources (see Figure 6). While this scene (A) is more realistic than the benchmarking scene used before, it still contains no real error sources (like jitters from the robot motion). Parameters for this simulation are given in Table 2.

Figure 6 shows the retinal projection of the first frame (B), where each neuron stores that particular gray level of the image at the respective location with which it is confronted. In part C, the ground-truth depth map is given. The other parts show two snapshots: one after 300 frames and the other at the end of the simulation (frame 600) of the depth map (global depth map for *confirmed* ≥ 1, right). In addition, we show the current depth map, which reflects the depth values computed from those neurons that are excited within a small time window of ± 20 frames around the current frame. Side-view maps shown beneath the depth maps clearly demonstrate that the density of the depth map increases during the run and also show that the depth estimates are rather accurate. As expected, depth errors increase with distance (compare, e.g., the ball and background). Note as before that pixels overlay each other, apparently reducing the density of the side-view map. Some of the pixels lie on the floor and reflect detected shadows. Only a few more are floating in the air, representing erroneous depth estimates, which, however, are still located rather close to the real objects.

2.5 Results for a Real Scene. Figures 7 and 8 show the results we obtained by analyzing a real scene that was recorded in 540 frames over 72 seconds using an NTSC zoom CCD-camera (1/2 inch CCD chip) with autofocus. The viewing angle of the camera was 55.2×44.1 degrees, and the focal length was approximately 8 mm (slightly changing during the run because of the autofocus). We used a DataCube as frame grabber with an initial resolution of 512×480 pixels and a frame rate of 7.5 Hz. The images were then subsampled by a factor of two, leading to a final resolution of 256×240, and then immediately transferred to a SUN SPARC 10 computer for analysis. The camera was mounted on a small vehicle. No special procedure was adopted to adjust the camera axis. Adjustment to the motion trajectory was performed only by hand while viewing the image on a regular monitor.

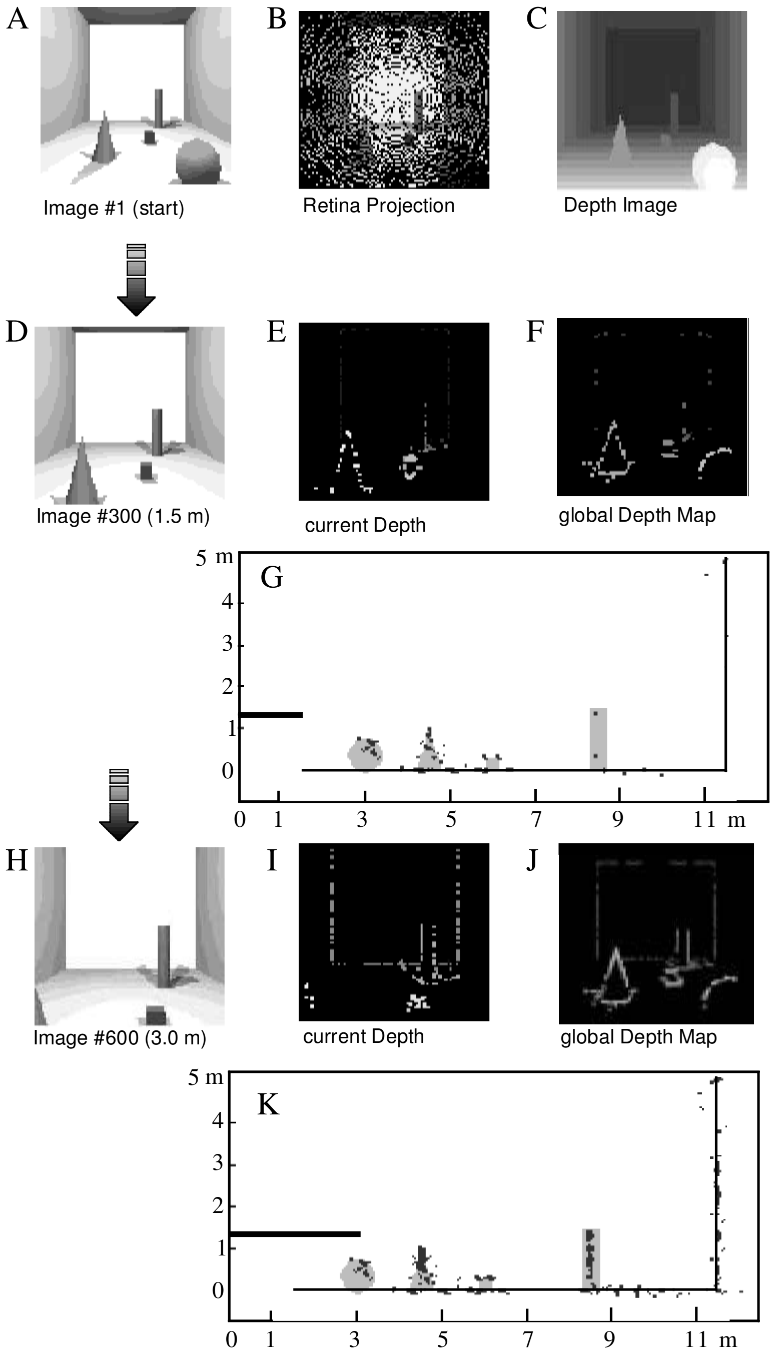

Figure 6: Results from the Ray-Traced Hallway Scene.

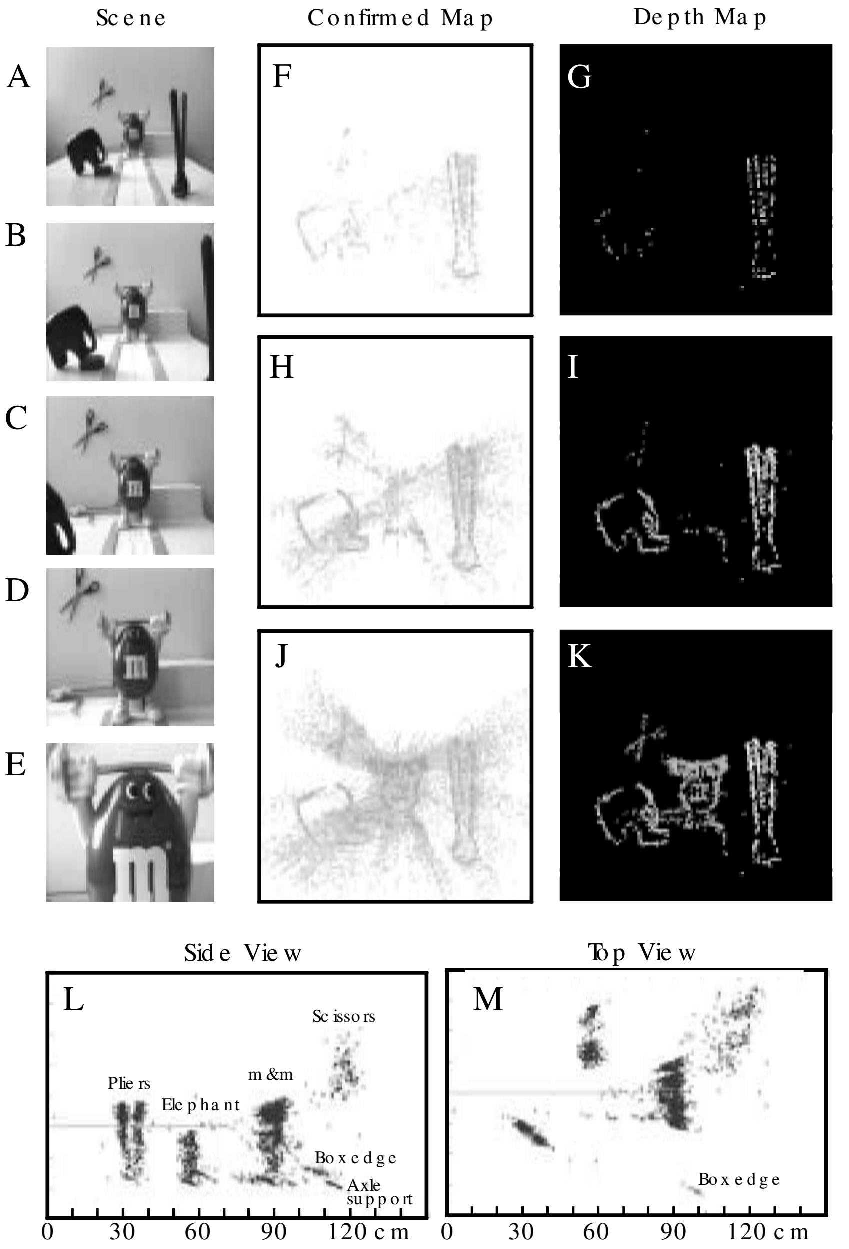

Figure 7: Results for a real scene. (A–E) Frames 100, 200, 300, 400, and 540. (F, H, J) Confirmation maps for frames 100, 300, and 540. (G, I, K) Depth maps for $C \geq 1$ for frames 100, 300, and 540. Different gray shades encode the relative depth of the objects. (L, M) Side-view and top-view map showing all data points in the depth map of frame 540.

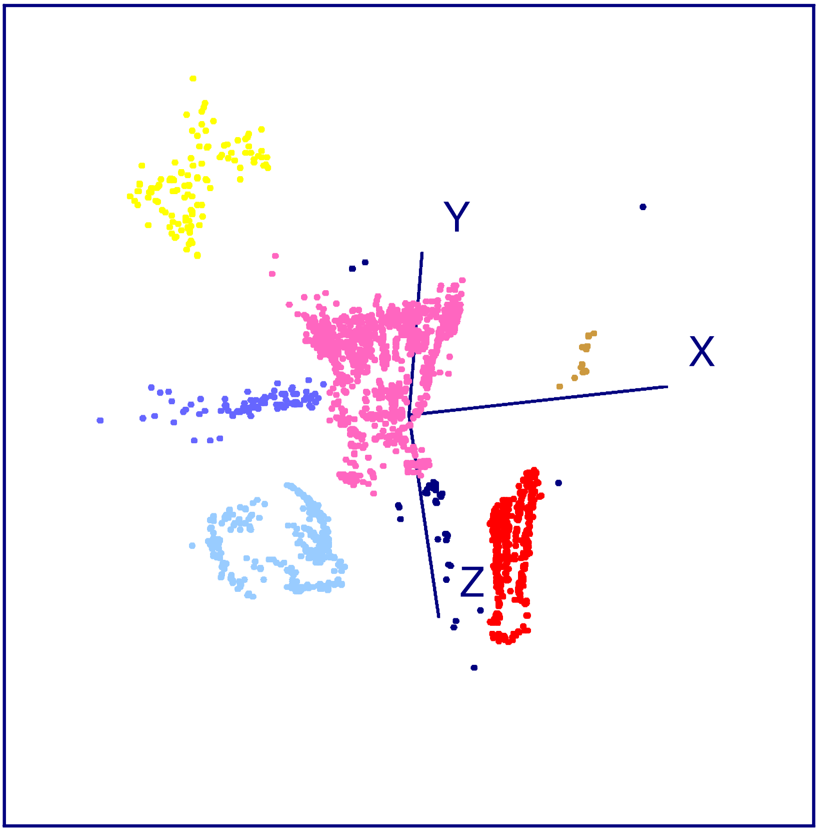

Figure 8: Aerial view onto the data clusters obtained from the real scene. Different shades show the different objects. The black outliers along the *z*-axis belong to the right rail, which was otherwise thresholded.

The vehicle was pulled across a table guided by two lateral rails. Pulling was achieved by means of a thin thread, which was continuously wound onto the extended axle of an electrical motor (visible in Figure 7D, left). The total traveled distance was 1.2 m at a velocity of 16.6 mm/s, which amounts to 2.22 mm per camera frame. The scene contained a pliers with their center of gravity at a distance of 0.35 m from the starting point, an elephant on a post (0.55 m), the M&M mascot (0.90 m), a white box that hides the motor, the axle of the motor and the axle support (all at about 1.10 m), and a scissors on the wall (1.20 m). The white box, the aluminum rails, and the (barely visible) thread were blanked out before analysis by thresholding all low-contrast objects. Apart from this thresholding, no other preprocessing of the image data was performed, and the algorithm operated at the unprocessed noisy

Table 3: Parameters for the Analysis of the Real Scene Shown in Figure 7.

Parameter	Value	**Parameter**	Value	**Parameter**	Value
Sequence	540 images	Neuron chains	600	*Confirmed*	1
Resolution	256×240 pixel	Neurons per chain N	≤ 64	$\epsilon_{\text{Position}}$	1.2 mm
Step	2.22 mm/frame	Radius ρ	≤ 150 pixel	$\epsilon_{\text{Displacement}}$	0.10 pixel

gray values as they were recorded. Note that this scene—like all other real-world scenes—is contaminated by reflections and shadows, which could in principle influence the final results. Table 3 lists the parameters of the algorithm used to analyze the scene.

Parts F, H, and J show the confirmation map for frames 100, 300, and 540, respectively. The confirmation maps show the gray-scale-coded value of the confirmed counter (light gray $= 0$, darker gray $= 1$, etc.). Thus, the confirmation map contains all data points so far encountered up to that particular frame. Most of these data points are encountered only once, which leads to a value of the confirmed counter of $C = 0$ (lightest shading). Some of them occur more often ($C \geq 1$). Panels G, I, and K represent the accumulated depth maps for the same frames showing only those data points confirmed at least once ($C \geq 1$). The outlines of the different objects are clearly visible and in good focus. The different gray shading indicates the distance of the objects from the starting point in absolute coordinates. After 100 frames (compare A), the closest objects, still at a safe distance, can be discerned, which would allow for steering maneuvers if desired. After the complete run, even finer details become visible, like the "M" of the M&M man. Intriguingly, a small part of the back of the elephant is left out. Probably this edge fell exactly between two adjacent radii of the retina and therefore remained invisible. Due to the slightly changing focal length as the consequence of the autofocus, an increasing radial spread of the data points is observed in the confirmation maps (F, H, J). The changing focal length, together with the hyperbolic projection geometry, leads to an enhanced radial displacement of the data points the closer the vehicle gets to a certain object. The actual depth maps (G, I, K) nicely demonstrate the efficacy of the confirmation mechanism, which is part of the algorithm. All (with the exception of very few) of these wrongly detected data points are eliminated because they are observed only once and never confirmed. This also applies to other erroneous data points (e.g., those induced by wandering reflections).

The top- and side-view maps give an estimate of the accuracy of the depth values. The different objects are clearly discernible, and even subparts like the two handles of the pliers can be seen. Only the axle support and the scissors close to the wall are confused in the top-view map. The side view, however, shows that these elements are also clearly separated. We determined the center-of-gravity Z-coordinates from histograms similar to those shown in Figure 5 (not shown here) for the four major objects in the scene as: pliers, 0.341 m; elephant, 0.558 m; M&M man, 0.892 m; and scissors, 1.191 m. None of these values deviates more then 10 mm from the true position. For the closest object (the pliers), the average relative error is maximal but still less than 2.6%. Top- and side-view maps also demonstrate that the depth-extend (thickness) of the individual objects is correctly retrieved.

These results show that the algorithm is applicable under real-world conditions, and the accuracy of the results should almost always suffice. The noise reduction due to the confirmation mechanisms is one major component that ensures this accuracy and robustness. The scene (A–E) and the motion parameters were arranged such that every parameter could be upscaled by a factor of 10 in order to represent the situation encountered (e.g., by big robot in an office or industrial environment). Due to the simplicity of the algorithm, data analysis could still be performed in real time at the given frame rate of 7.5 Hz using a rather slow SUN SPARC 10 workstation. Furthermore, it should be noted that no image preprocessing was performed. We would expect that the quality of the results could be further improved, for example, by applying edge-enhancement algorithms prior to depth analysis. With a more powerful computer, we would also estimate that the frame rate could be at least tree times higher. Given that the first 10 to 100 reliable depth estimates occur within 50 frames, the traveled distance at the higher frame rate would be only 37 mm. Thus, "robot blindness" would be restricted to a very short distance after a turn, even when using a regular serially operating processor. Any parallel implementation would be even faster.

Figure 9 shows how the direct analysis of the flow field (Lucas & Kanade, 1981; Barron et al., 1994b) performs on the same real-world example. The top-view map is shown as in Figure 7. Parameters of the algorithm were adjusted such that about the same density of depth estimates was achieved.

Only vague outlines of the objects are discernible (circled) but without prior knowledge of the scene (e.g., through an image segmentation algorithm; Opara & Wörgötter, 1998), no matching between the objects and their depth coordinates is possible. In addition, the accuracy of the depth estimates is very low. The reason for the poor performance is the noise and the small systematic distortion due to the zooming in the images. Direct flow-field analysis is not very robust against these effects as compared to our method, which includes the confirmation mechanism.

3 Discussion

3.1 Advantages and Limitations. The goal of this study was to design a fast parallel-implementable module that performs depth analysis in real

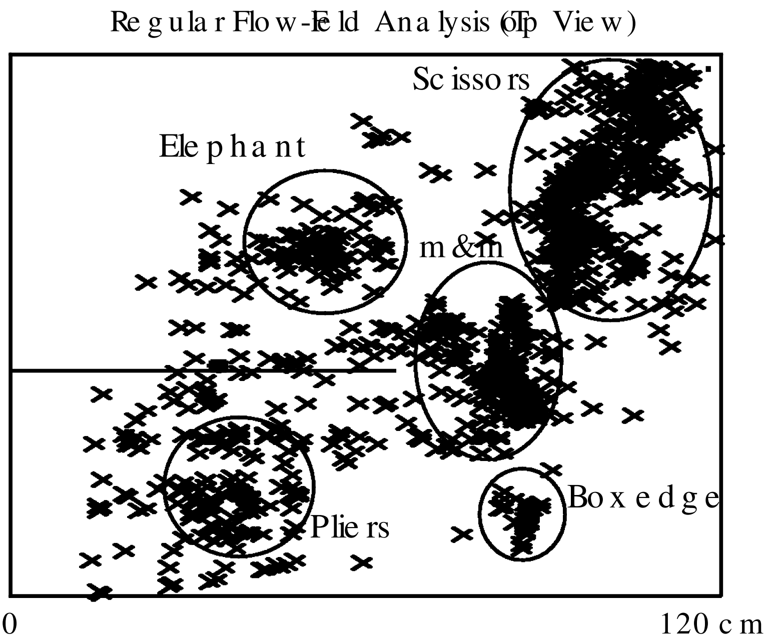

Figure 9: Results for the same real scene as in Figure 7, obtained by direct flow-field analysis.

time. We were able to show that:

1. All calculations remain local, and data transfer exists only between neighbors (with the exception of the final read-out). Thus, the algorithm can be implemented easily in parallel.
2. The computational complexity in our algorithm is reduced to two scaling operations.
3. The confirmation mechanism reduced noise and other error sources tremendously.
4. The error under different testing conditions remained below $\approx 2\%$ for reasonable parameter settings and remained mostly much smaller (see the appendix).

The simplicity of our equations results from the neuronal architecture in combination with the restriction to radial flow. Other algorithms are also substantially reduced in their complexity when considering only such restricted flow fields (see below), but the simplicity of needing only scalar multiplications can be obtained only when considering such a radial, parallel neuronal architecture (compare the "time-to-crash" detector; Ancona & Poggio, 1993). The virtue of a (possible) parallel implementation together with the simplicity of the approach is still not sufficient to render our approach useful, because local operations are exceedingly sensitive to noise. The additionally introduced novel confirmation mechanism solves this noise problem. This mechanism is also able to eliminate even significant systematic errors, like those introduced by optical distortions (e.g., due to using an autofocus zoom lens; see Figure 7). This argument, together with the outcome of the error analysis, gives a positive answer to the question of whether the algorithm would tolerate error sources in general like mismeasured step counts of the robot motion due to, say, a rugged terrain. The scaling properties of the curves shown in Figure 11 indicate a quite high tolerance of such error sources.[5] All this shows that the algorithm is indeed functioning very well under the constraint of a radial flow field. Thus, the crucial question that needs to be answered is if it would be applicable for different motion trajectories that also contain turns. The answer to this question lies in a combined speed and accuracy estimate.

The analysis of the systematic errors and of the aliasing behavior (see the appendix) showed that even in a serial implementation on a regular computer, systematic errors and aliasing problems are almost always negligible. Due to the simplicity of the algorithm, we can assume that a slightly faster machine will allow for frame rates of above 30 Hz. This now answers the critical question above: Even if the trajectories remain restricted to motion along only the camera axis, such high frame rates in a serial or parallel implementation allow for the analysis of short motion segments that within a short time window will produce a rather distinct map of the environment. The robot is allowed to turn rather abruptly, which would lead to only a brief reset of the algorithm, and the distance traveled before the novel map emerges after the reset remains very small. Thus, even without a parallel implementation, it should be possible to generate a sufficiently accurate complete depth map by piecing together linear motion segments provided that the robot changes its direction less often than once every 2 seconds or so,[6] rendering more than 60 consecutive frames.

The comparison of the performance of our restricted algorithmic version with standard flow-field analysis (Lucas & Kanade, 1981; Barron et al., 1994b) provides additional support to the sensibility of tailoring an algorithm to the restricted situation of radial flow fields.

Still, the question arises as to whether there is a way to generalize our algorithm to a less restricted situation. This is discussed in the last section.

[5] Indeed we observed a quite visible jerkiness of the camera motion when recording the visual scene due to slippage and a somewhat nonradial rotation of the motor axis. This error was also nicely eliminated by the confirmation mechanism, and the residual error remained so low that we found the final results to be rather accurate.

[6] A turn every 2 seconds still seems rather unrealistic. Almost all navigating robots in industrial environments turn much less frequently.

Complications can occur as the consequence of object motion that is not immediately detected by our algorithm. Slowly moving objects with a rather homogeneous structure will nevertheless be "seen," but their recognized shape is smaller than in reality. It is clear that this algorithm was not designed for such situations, which are also hard to resolve for most other algorithms. The algorithm will also fail if the terrain is too rugged. We have already noted that a certain robustness against jitter exists, but due to the limited detection range of the individual neurons, data points will be missed if the camera jitter becomes too strong.

3.2 Comparing the Algorithms. The problem of recovering the 3D structure of a scene from the optical flow has often been discussed in the literature (Longuet-Higgins & Prazdny, 1980; Prazdny, 1980; Heeger & Jepson, 1990; Little & Verri, 1989; Nelson & Aloimonos, 1989). Still, most of the literature deals with the general problem of estimating the self-motion, and the structure of the scene can be estimated from the optical flow. In the special case of pure forward motion with known constant speed that we take in account, the differences between the approaches reported in the literature disappear, and the depth recovery equations become extremely simple. Starting from the perspective projection equation expressed in polar coordinates $\rho = fR/Z$, the depth of the point is given by the simple relation $Z = -(\rho \dot{Z})/(v_\rho)$, where ρ is the radial coordinate of the projection, v_ρ is the radial component of the optical flow, and $\dot{Z}$ is the forward speed. Although these algorithms also become very simple, a reduction to single scalar multiplications can be achieved only by our parallel architecture.

In addition, there is a severe problem: Using this equation to recover the structure of the scene relies heavily on the precision of the measured optical flow. Small errors in the flow vectors are amplified by the factor $\dot{Z}/v_\rho$, in particular around the center of the image, where the flow vectors are very small (see Figure 9). This problem is unavoidable unless smoothness assumptions about the scene are made, for example, allowing neighboring measurements to merge or a sequence of estimations is integrated using data fusion techniques, like the Kalman filter (see Matthies, Kanade, & Szeliski, 1989). Such algorithmic extensions are far more complex than our simple confirmation mechanism, which solves the noise problem in a satisfactory way. There may also be other ways to account explicitly for the restricted situation of only radial flow, for example, by modifying the "classical" flow-field equations in order to make them numerically more stable around the focus of expansion, but we did not investigate this further.

3.3 Confidence Measurement by Long-Range Couplings. The last section also shows that regardless of which flow-field algorithm is used, it is of utmost importance to include a confidence measure in order to judge the accuracy of the depth estimates (Barron et al., 1994a). In our approach, confidence in the data points is gained by means of the confirmation mechanism.

The propagation of information along the radii by this predictive mechanism is equivalent to a long-range information exchange in a parallel network. In our case, this can be interpreted as if the detector range (receptive field) of each neuron was enlarged. Thus, currently only the confirmation mechanism ensures the necessary noise reduction, and the improvement of the algorithm is truly dramatic (see Figures 4 and 7).

3.4 Generalizing the Algorithm. A more generalized version of the algorithm could be obtained by allowing for a more extended neuronal coupling, which exceeds the nearest-neighbor interactions currently implemented in the algorithm. Single misses at any neuron would become insignificant in this way (see note 2). An additional extension, which is also interesting from a biological viewpoint, would be to introduce more complexity in the receptive fields of the detectors. Here a natural choice is to use center-surround receptive fields with a significant spatial overlap along and across the radii. In this case, a more elaborate version of the algorithm would be required, having to account for the now-existing lateral inhibition.

This leads us to the possibility of more generalized architectures. The central problem behind any generalization is the attempt to reach certain invariances, such as against scaling or rotation, which are common in flow fields. Indeed, there is a biologically motivated way to achieve a higher degree of invariance: Retinal coordinates are projected onto the visual cortex employing (roughly) a complex logarithmic transform (Schwartz, 1977). By means of this, rotational and scaling invariance is obtained because rotations or scaling operations translate into horizontal (medio-lateral) or vertical (anterior-posterior) shifts on the cortical grid, respectively. A combination of both leads to an oblique shift (Schwartz, 1980). In the context of our algorithm, one could now think of a rectangular grid with horizontal, vertical, and (several angles of) oblique connections replacing the design of our retina here. Then, given the results of the current study, it seems likely that a relatively simple set of local equations could be found that operate on such a grid after the input images have been transformed by the complex logarithm. The advantages of a complex logarithmic mapping in the context of classical (nonparallel) flow-field analysis have been demonstrated already (Tistarelli & Sandini, 1993). However, a parallel version does not exist yet. The ultimate version of a parallel algorithm for flow-field analysis would probably make use of such a wire mesh connection pattern and employ spatiotemporal receptive fields (e.g., spatiotemporal Gabor filters) in order to implement one of the well-consolidated phase-based or energy-based flow-field algorithms (Heeger, 1988; Fleet & Jepson, 1990). For this reason we think that the current study is the first step toward a class of parallel algorithms that could become of greater relevance in image analysis as soon as more sophisticated ways for producing parallel VLSI chips exist.

3.5 A Possible Hardware Implementation. The central advantage that makes our algorithm so simple is that all computations at a given neuron remain restricted to scaling operations. This feature is not dependent on the actual distribution of neurons on the retina. Such scaling operations could in principle be performed even in analog hardware by amplifiers adjusted to the right gain. Memory transfer operations are also rather limited because transfer occurs only in one direction (radially) between pairs of neurons. In a parallel system, two related problems remain: (1) how to set the individual gain values for each neuron and (2) how to retrieve the depth map from the output of the different neurons. Douglas and Mahowald (1995; see also Mahowald & Douglas, 1991) have suggested a hardware for a multiplexing system that allows loading (and retrieving) values into (from) individual neurons in a parallel network. Such a system, or a similar one, could be used for this purpose. Loading would have to be performed only once, and for the retrieval it would have to operate at manageable frequencies of below 50 kHz even for very large networks at high frame rates (e.g., 100 Hz).

The layout of the retina should allow for a relatively easy hardware implementation (Pardo, 1994) as compared to other more elaborate parallel flow field algorithms (Bülthoff, Little, & Poggio, 1989). This could be achieved by a regular grid layout of the photoreceptive sites and an additional address decoder like the so-called Phytagoras processor (GEC Plessey, Semiconductors, PDSP16330/A/B), which converts 16-bit Cartesian coordinates into polar coordinates such that a radial arrangement and also the subsequent computations can be electronically simulated. A related approach, simulating the compound eye of flies, has already been undertaken by Franceschini and colleagues (Franceschini, Pichon & Blanes, 1992; Franceschini, 1996).

Appendix: Error Analysis

A.1 Systematic Errors. In the following section, we analyze two types of systematic errors that are inherent in the design of the algorithm. We distinguish between the depth error and the aliasing problem.

A.1.1 Depth Errors from Mismeasurements and Neuron Placement Inaccuracies. By depth error, we mean any error of the computed Z-component as compared to the true Z-component of an object. Figure 10 shows the actual situation most commonly encountered when an edge (vertical line) excites two neurons subsequently. Due to the radial flow, the edge travels along the ray indicated by the dashed line. The finite resolution of the pixel grid (usually integer resolution), however, restricts the placement of the neurons to the pixel centers, as drawn in the figure. Therefore, the left neuron will be excited too early by the edge and the right neuron too late.[7] The actual

[7] In other words, this means that as opposed to the optimal situation, both neurons are not excited by exactly the same location on the edge.

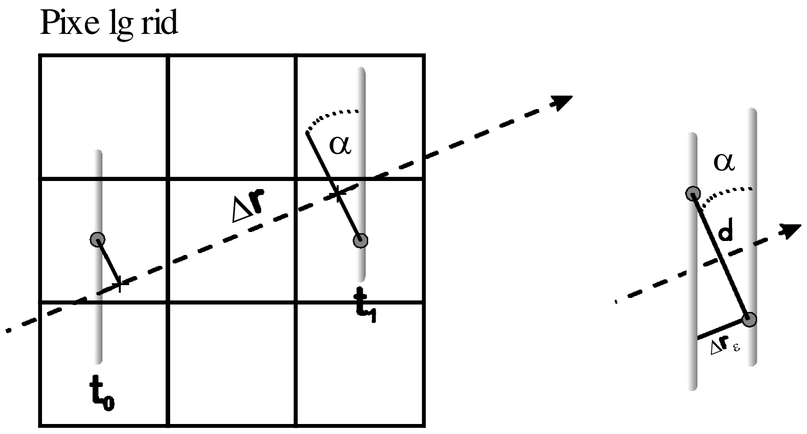

Figure 10: Geometry underlying the error estimation.

measurement error that results from this effect is given by Δr_ϵ. The radial distance between the neurons used to compute the depth estimate is Δr. Thus, the actually traveled distance on the ray is underestimated by Δr_ϵ and the true depth estimate would be obtained with $\Delta r_{true} = \Delta r + \Delta r_\epsilon$. This error depends on the sum of the distances between the ray and the neurons (d) and on the angle between ray and edge (α). It is immediately clear that the Δr_ϵ is zero for $\alpha = 90$ degrees, whereas it becomes infinitely large for $\alpha = 0$ degrees.

Under the assumption that: $s_{n-1,z} = s_{n,z} = f$, we get for the depth estimate Z:

$$Z = \frac{\Delta Z}{\frac{s_{n,r} - s_{n-1,r}}{s_{n-1,r}}} = \Delta Z \frac{s_{n-1,r}}{s_{n,r} - s_{n-1,r}}. \tag{A.1}$$

Let $s_{n,r} = s_{n-1,r} + \Delta r$. Then,

$$Z = \Delta Z \frac{s_{n-1,r}}{\Delta r} = \Delta Z \frac{r_{n-1}}{\Delta r}. \tag{A.2}$$

The second form of this equation relates to the labeling of the variables used in Figure 12A and will become of relevance later. In addition to the error introduced by the neuron placement (Δr_ϵ), we have to take into account that the measured value of ΔZ is probably erroneous, for example, due to inaccuracies following a wrong count of the stepper motor between two subsequently excited neurons. Thus, we should assume that the actually measured value is given by $\widetilde{\Delta Z} = \Delta Z + \Delta Z_\epsilon$, where ΔZ is the true value and ΔZ_ϵ the measuring error.

Thus, the erroneously estimated depth is given by

$$\widetilde{Z} = \widetilde{\Delta Z}\frac{r_{n-1}}{\Delta r} = (\Delta Z + \Delta Z_\epsilon)\frac{r_{n-1}}{\Delta r}. \tag{A.3}$$

On the other hand, the correct estimate would be:

$$Z = \Delta Z\frac{r_{n-1}}{\Delta r + \Delta r_\epsilon}. \tag{A.4}$$

The absolute error is then:

$$Z_\epsilon = \widetilde{Z} - Z = (\Delta Z + \Delta Z_\epsilon)\frac{r_{n-1}}{\Delta r} - \Delta Z\frac{r_{n-1}}{\Delta r + \Delta r_\epsilon}. \tag{A.5}$$

The relative error is

$$\epsilon = \frac{Z_\epsilon}{Z} = \frac{\Delta r_\epsilon}{\Delta r}\left(1 + \frac{\Delta Z_\epsilon}{\Delta Z}\right) + \frac{\Delta Z_\epsilon}{\Delta Z}. \tag{A.6}$$

The last equation shows that the relative error critically depends on the "relative placement error" $\frac{\Delta r_\epsilon}{\Delta r}$, which could in principle reach infinity. The examples from above (see Figures 4, 6, and 7), however, show that this is practically never the case. Nonetheless, in the course of this study, we observed that Δr_ϵ can have a tremendously destructive impact on the results as soon as d (see Figure 10) is too large. In a hardware implementation, the neuron grid can be made fine enough in order to reduce d sufficiently, such that this problem is negligible. In our computer implementation, however, we had to find a work-around. Therefore, we resorted to slightly modify the exact geometrical spacing of the neurons on the retina given by equation 2.2 and shifted them a small amount away from the computed locations in order to ensure that the displacement d of two adjacent neurons never exceeds the predefined threshold of $\epsilon_{\text{Displacement}}$ (see Tables 1 and 2). If the limit given by $\epsilon_{\text{Displacement}}$ could not be achieved by neuron shifting, the neuron was eliminated from the retina. In this way, the initial calculation of the lookup table for the neuron-dependent constants $\overrightarrow{P_n}$ became more complicated, but otherwise the accuracy of the results shown in Figures 4 and 6 would have deteriorated.

Figure 11 shows how the parameters $\epsilon_{\text{Position}}$ and $\epsilon_{\text{Displacement}}$ affect the average depth error and the average density of the depth map computed for two objects in the scene shown in Figure 4. For this diagram, the measurements of the triangle at 4.0 m and the vertical bar at 5.5 m in Figure 4 were evaluated for the run with and without noise. In general, the vertical bar (dotted lines) is less susceptible to error than the triangle (solid lines) because of the orientation of its edges. The orientation of an object edge relative to the radii of the retina thereby determines the error susceptibility. If an edge is parallel (orthogonal) to a radius, the error will be high

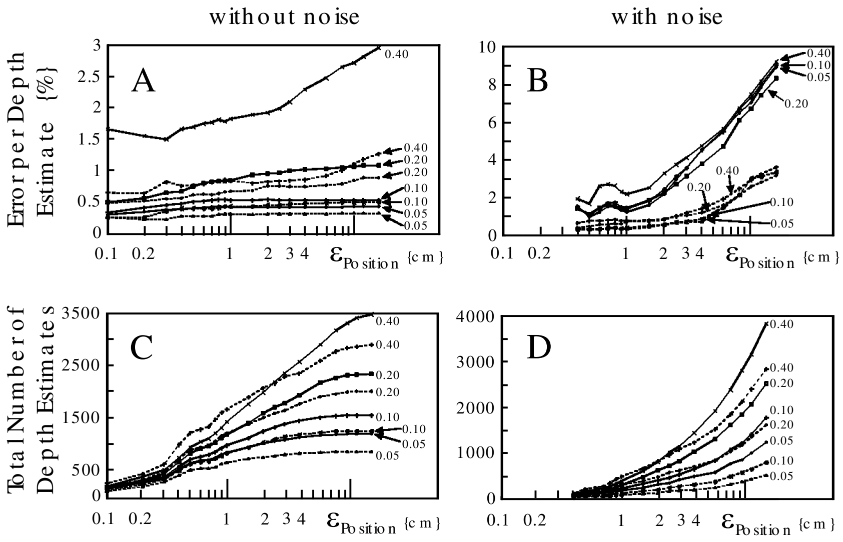

Figure 11: Depth error and density of the depth maps plotted for four different settings of $\epsilon_{\text{Displacement}} = 0.05, 0.1, 0.2, 0.4$ pixel (marked on the curves) against $\epsilon_{\text{Position}}$, which has been varied between 0.1 and 16.0 cm. Results from the triangle (solid lines) and the vertical bar (dotted lines) from Figure 4 are shown with and without noise after the complete simulated robot run. To make them comparable, error values are normalized with respect to the number of total depth estimates obtained. With noise, the algorithm becomes unstable for $\epsilon_{\text{Position}} < 0.3$; thus, these values have been excluded.

(low). Without noise (see Figure 11A) the error increases in small steps with increasing $\epsilon_{\text{Displacement}}$ but remains almost the same for different $\epsilon_{\text{Position}}$ values. As soon as noise is introduced (B) the situation reverts, and $\epsilon_{\text{Position}}$ is the more sensitive parameter. For large values of $\epsilon_{\text{Position}}$, the curves saturate at the maximal number of obtainable depth estimates in the case of no noise (C). Such a saturation is not observed if noise is present; instead, more and more wrong pixels are included in the depth map if $\epsilon_{\text{Position}}$ is increased (D). In summary, these diagrams show that if very little noise is expected from the robot's camera system, large values of $\epsilon_{\text{Position}}$ should be used, while $\epsilon_{\text{Displacement}}$ is uncritical. On the other hand, if the noise level is high, one should increase the value for $\epsilon_{\text{Displacement}}$ in order to get more depth estimates but keep the value of $\epsilon_{\text{Position}}$ low in order to reduce the error.

In our simulations, we found that a reasonable range that limits the number of totally misplaced points is given by $0.1 < \epsilon_{\text{Displacement}} < 0.3$ and $0.5 < \epsilon_{\text{Position}} < 2.0$ for interframe distances of 1 cm. Both parameters scale with the step size.

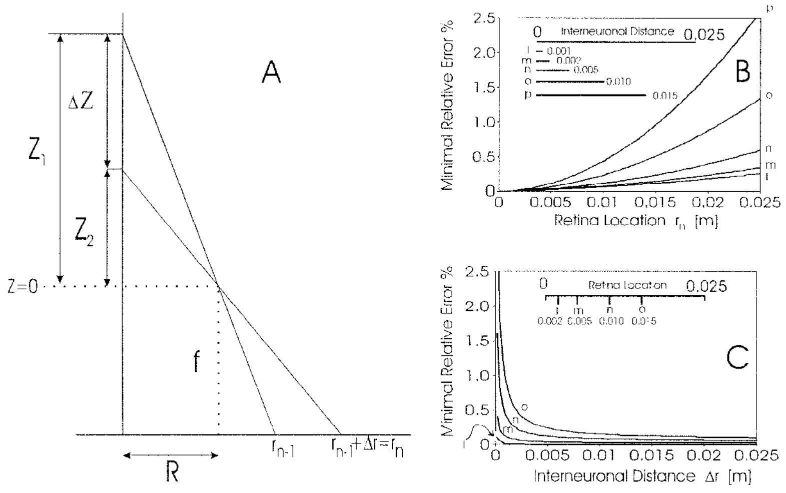

Figure 12: (A) Geometrical definitions used for the calculation of the systematic errors and the aliasing. (B, C) Systematic errors. Equation A.10 is plotted using (B) the interneuronal distance and (C) the retina location as parameter. Scale bars indicate the values of the fixed parameters.

It should be emphasized that the error introduced by the neuronal placement is a typical grid-aliasing problem and becomes irrelevant by means of the described anti-aliasing procedure or as soon as the grid is fine enough (e.g., in a hardware implementation). For this reason, we will restrict all further analysis to the unavoidable "relative measuring error" $\Delta Z_\epsilon / \Delta Z$ introduced by false measurements.

Setting $\Delta r_\epsilon = 0$, the equation for the relative error (see equation A.6) is reduced to:

$$\epsilon = \frac{\Delta Z_\epsilon}{\Delta Z}. \tag{A.7}$$

We assume a geometry as shown in Figure 12A, which is essentially identical to the one used for deriving the basic equations, with the exception of relabeling a few variables for convenience.

At the stepper motor counter, the minimal nonzero measuring error is one; that is, to get the minimal relative error, we set $\Delta Z_\epsilon = 1$. For any error bigger than one, Figures 12B and 12C would have to be scaled.[8] Then the

[8] All following diagrams are metrically scaled. To get this, we have defined a motion constant of 1 mm/Step of the stepper motor, which will not be explicitly mentioned in the following equations.

relative error is simply:

$$\epsilon_{\min} = \frac{1}{\Delta Z}. \tag{A.8}$$

This is intuitively clear. Since we assume a constant (minimal) measuring error of 1, this mismeasurement will contribute a lot to the relative error if the total measured interval ΔZ is small. The question arises, How will the parameters of the retina design will affect the relative error? Using equation A.2, we get:

$$\epsilon_{\min} = \frac{r_{n-1}\,(r_{n-1} + \Delta r)}{\Delta r \cdot f \cdot R}, \tag{A.9}$$

where R is the radial component of the cylinder coordinates of the object in Figure 12A. We set $R = 1$ m, which means we estimate the error for all objects at that particular lateral distance and get:

$$\epsilon_{\min} = \frac{r_{n-1}^2 + r_{n-1}\,\Delta r}{\Delta r \cdot f} = \frac{1}{f}\left(\frac{r_{n-1}^2}{\Delta r} + r_{n-1}\right) \tag{A.10}$$

Figures 12B and 12C show the behavior of equation A.10 for a retina with radius $\rho = 0.025$ m and a focal length of $f = 0.025$ m. For the sake of completeness, the curves in Figures 12B and 12C extend into meaningless regions (e.g., interneuronal distance of 0.015 m at a total radius of only 0.025 m, etc.). These cases were included to show the shape of the total curves better.

The figure and the corresponding equation show that the relative error is strongly affected by the retina position r_{n-1}, and Z-coordinates computed by neurons in the far periphery of the retina are highly sensitive to measuring errors (see Figure 12B). These curves have been obtained for a minimal measuring error of one step, and the curves need to be multiplicatively upscaled if the error would be larger. From the curves, it can be seen that even for the worst case, the (minimal) relative error is very low. Thus, the system survives significant error upscaling, which also explains the rather high accuracy of the results obtained from the simulations. In addition, the relative error is inversely proportional to the focal length f (held constant in the figure) and to the distance between two neurons Δr. The sensitivity to this parameter (see Figure 12C), however, is much smaller than that to the retinal position (B). At first glance, this inverse relation is quite intriguing, because it means that if the distance between the two adjacent neurons is lowered, then the error increases. In other words, if the neuronal density is increased by raising N, then Δr gets smaller but the error at a given retinal location[9]

[9] For this error estimation we had to fix the retinal location at r_{n-1}. Thus, it is not

also increases. The reason for this counterintuitive observation lies in the fact that for decreasing Δr, the measured interval ΔZ also decreases, and this increases the relative error. Average neuronal distances of 1 or 2 mm, however, still lead to rather small errors such that total neuron numbers between 50 and 100, as used in the simulations, are very well applicable.

A.1.2 Aliasing Problem. As an additional effect, which is of practical relevance, one needs to consider the time between two camera frames. If this time is too long, excitations will skip one or more neurons as soon as these are too densely packed. This problem is of central relevance for a system that implements the algorithm with conventional hardware (i.e., as a serial program on a computer), because the computational effort will limit the number of frames per second significantly. The following discussion is specifically dedicated to such a computer implementation. Thus, this section is rather irrelevant for a parallel processing system with high frame rate where no such aliasing occurs over a huge parameter range.

We will compute the shape of the maximal region in the environment that the robot will "see" without aliasing. We will consider the limit case where two subsequent excitations of an edge will fall exactly onto two subsequent neurons, r_{n-1} and r_n.

Considering the same geometry as before (see Figure 12A), we have:

$$\tfrac{Z_1}{R} = \tfrac{f}{r_{n-1}} \qquad \text{and} \qquad \tfrac{Z_2}{R} = \tfrac{f}{r_n}. \tag{A.11}$$

From this and equation 2.4, we get:

$$Z_2 = \frac{r_{n-1}}{r_n} Z_1 = \frac{n-1}{n+1} Z_1. \tag{A.12}$$

Let v be the velocity and μ the camera frame rate. Then:

$$Z_1 = Z_2 + \frac{v}{\mu} \tag{A.13}$$

and

$$Z_2 = \frac{n-1}{n+1}\left(Z_2 + \frac{v}{\mu}\right). \tag{A.14}$$

Solving this for the neuron number, we get:

$$n = \frac{2\mu}{v} Z_2 + 1. \tag{A.15}$$

possible to introduce equation 2.4 here, because in this equation the neuronal positions shift with changing N or ρ.

If this equation holds for an object distance Z_2, then n is the number of the outermost neuron r_n for which no aliasing occurs at a given frame rate and velocity. The neuron number n and the neuron location r_n are directly related by equation 2.4. Due to the simple geometry, it is also possible to project the neuron location back into the environment along the ray that connects neuron r_n with the nodal point.

From equation 2.4, we get:

$$n_{1,2} = -\frac{1}{2} \pm \sqrt{\frac{1}{4} + \frac{r_n}{h}}\,. \tag{A.16}$$

This equation enters in equation A.15. In addition, we substitute r_n using equation A.11 and after some arithmetic get:

$$R = \frac{h}{f}\left(\frac{4\mu^2 Z_2^3}{v^2} + \frac{6\mu Z_2^2}{v} + 2Z_2\right). \tag{A.17}$$

Given an object with depth Z_2, equation A.17 describes the radial distance R from the optical axis, which is maximally allowed such that no aliasing occurs. In other words, as soon as this object has a radial distance larger than R, its detection is subject to aliasing.

The actual depth of the object is the most crucial parameter and enters with a power of three. Objects very nearby are therefore almost always detected with aliasing. From the controllable parameters, frame rate, velocity, and total neuron number (contained in h) are most sensitive; focal length and retina radius (also contained in h) contribute less strongly.

Figure 13A shows the regions for which aliasing occurs at different velocities. To obtain these curves, we have assumed a total number of $N = 50$ neurons, a frame rate of $\mu = 5$ images per second, a retina radius of $\rho = 0.025$ m, and a focal length of $f = 0.025$ m. The solid lines reflect a reasonable working range lying inside the "bowl" enclosed by the two solid lines. For this curve, a radial displacement of maximally $|R| \approx 0.8$ m is allowed for objects that are 1 m away ($Z_2 = 1$ m) from the robot (crossing points with horizontal line).

In the following (see Figures 13B and 13C), we keep $\rho = 0.025$ m and $f = 0.025$ m. If we assume that a detection range of $R = \pm 0.5$ m is desired for objects at a distance of $Z_2 = 1$ m, then we can solve equation A.17 for the velocity and plot v as a function of the total neuron number (see Figure 13B).

The plotted equation reads:

$$v = \frac{6\mu}{N^2 + N - 4} + \frac{2\mu}{N^2 + N - 4}\sqrt{2N^2 + 2N + 1}\,. \tag{A.18}$$

Thus, the diagram shows for different frame rates the maximally allowed

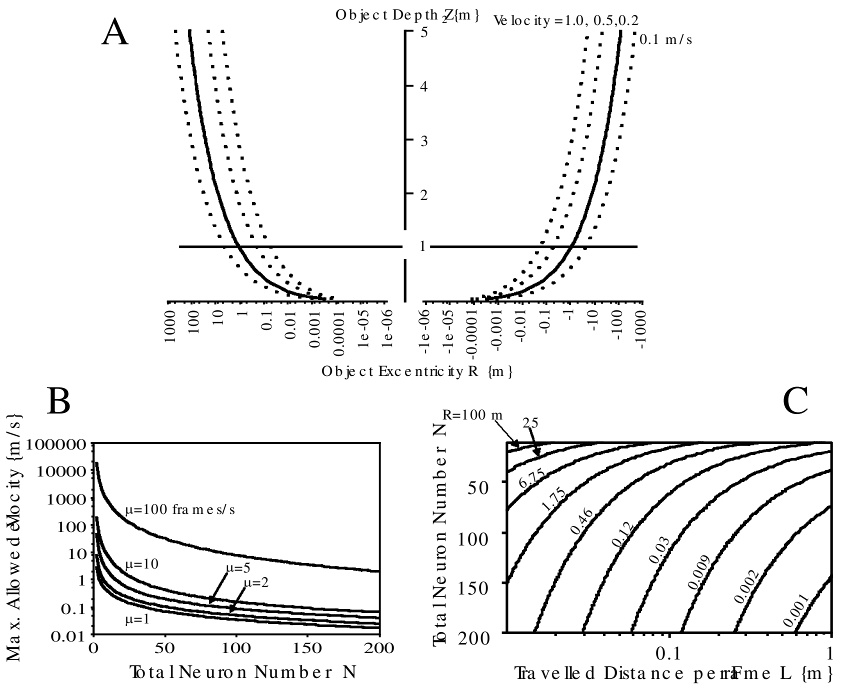

Figure 13: Aliasing behavior. For A–C we set $f = \rho = 0.025$ m. (A) The regions in which no aliasing occurs (in between the curves) for different robot velocities and with $N = 50$, $\mu = 5$. (B) The maximally allowed velocity for which no aliasing occurs at $(R, Z) = (\pm\frac{1}{2}, 1)$ [m] for different total neuron numbers using the frame rate as parameter. (C) A contour plot showing the allowed radial range R at $Z = 1$ m for different neuron numbers and different traveled distances per frame.

velocity at a given neuron number for which no aliasing occurs for an object with coordinates $R = \pm 0.5$ m, $Z = 1$ m. If such a system would be designed serially with conventional hardware (a program on a workstation) the computational effort will limit the frame rate, and it is reasonable to assume frame rates of about 10 Hz. In this case, velocities between 0.5 m/s and 1.5 m/s will be obtained for neuron numbers between 25 and 60. If the system would be designed by parallel processing hardware, much higher frame rates could be achieved. The top curve shows that robot velocity is not a limiting factor even for frame rates of 100 Hz for any neuron number below 200. Such a frame rate should be easily obtainable in a parallel processing system.

Finally we note that v/μ is identical to the distance L the robot travels between two images. Again we set $Z_2 = 1$ m and rewrite equation A.17 as:

$$R = \frac{2}{N^2 + N}\left(\frac{2}{L^2} + \frac{3}{L} + 1\right). \qquad \text{(A.19)}$$

This allows us to generate a contour plot (see Figure 13C) that shows the iso-radii that limit the range where no aliasing occurs for objects 1 m away as a function of both, N and L. The contour lines between $R = 0.46$ m and $R = 1.75$ m reflect a reasonable working range, and the total number of neurons N per radius can be chosen according to the desired speed and frame rate of the system.

It should be remembered that the analysis of the aliasing behavior is based on the limit case assumption—the assumption that an edge will—in the limit case— excite exactly two subsequent neurons in two subsequent frames. If the neurons have highly nonoverlapping excitable regions ("receptive fields"), then the projection of a thin edge could also fall between two neurons, exciting neither of them (dubbed an *in-between miss*). This could lead to a deterioration of the performance even for parameter settings that would be tolerated under the limit case aliasing condition. However, it can be expected that the problem of in-between misses is of minor relevance in any realistic situation, because edges usually are not infinitely thin. If the color on this edge surface is relatively similar over small distances, then neuron r_n will detect a different part of the edge as compared to neuron r_{n-1}, but at least it will not experience an in-between miss. This will lead to a small error in the depth estimate similar to the one introduced by the neuron placement problem discussed above. This error in the Z-component, however, is negligible (see Figures 4 and 6).

Acknowledgments

We thank R. Opara and B. Porr for their advice at several stages of this project. F. W. acknowledges the support from several grants of the Deutsche Forschungsgemeinschaft WO 388, as well as from the European Community ESPRIT Program (CORMORANT). Furthermore we thank the robotics group at the Computer Science Department of the University of Bonn for letting us use their facilities. Thanks are due to O. Güntürkün, J. Ostheim, and H. Wagner for providing detailed information and helping us interpret their animal behavior studies. This research was not sponsored by M&Ms, but we enjoyed a fair number of them during programming.

References

Ancona, N., & Poggio, T. (1993). *Optical flow from 1D correlation: Application to a simple time-to-crash detector* (A.I. Memo No. 1375). Cambridge, MA: Artificial Intelligence Laboratory, Massachusetts Institute of Technology. Available online at: http://www.ai.mit.edu/publications/bibliography/BIB-online.html.

Barron, J. L., Beauchemin, S., & Fleet, D. (1994a). *On optical flow*. Presented at the Sixth International Conference on Artificial Intelligence and Information-Control Systems of Robots, Bratislava, Slovakia, September 12–16 (pp. 3–14). The C source code is available on the FTP server at ftp://csd.uwo.ca/pub/vision.

Barron, J. L., Fleet, D., & Beauchemin, S. (1994b). Performance of optical flow techniques. *Int. J. Comp. Vis., 12*, 43–77. The C source code is available on the FTP server at ftp://csd.uwo.ca/pub/vision.

Bülthoff, H., Little, J., & Poggio, T. (1989). A parallel algorithm for real-time computation of optical flow. *Nature, 337*, 549–553.

Davies, M. N. O., & Green, P. R. (1988). Head-bobbing during walking, running and flying: Relative motion perception in the pigeon. *J. Exp. Biol., 138*, 71–91.

Davies, M. N. O., & Green, P. R. (1990). Optic flow-field variables trigger landing in hawk but not in pigeons. *Naturwissenschaften, 77*, 142–144.

Douglas, R., & Mahowald, M. (1995). Silicon neurons. In M. Arbib (Ed.), *The handbook of brain theory and neural networks* (pp. 871–875). Cambridge, MA: Bradford Books, MIT Press.

Duffy, C. J., & Wurtz, R. H. (1991). Sensitivity of MST neurons to optic flow stimuli. I. A continuum of response selectivity to large-field stimuli. *J. Neurophysiol., 65*, 1329–1345.

Duffy, C. J., & Wurtz, R. H. (1995). Response of monkey MST neurons to optic flow stimuli with shifted centers of motion. *J. Neurosci., 7*, 5192–5208.

Erichsen, J. T., Hodos, W., Evinger, C., Bessette, B. B., & Phillips, S. J. (1989). Head orientation in pigeons: Postural, locomotor and visual determinants. *Brain Behav. Evol., 33*, 268–278.

Fennema, C., & Thompson, W. (1979). Velocity determination in scenes containing several moving objects. *Comp. Graph. Image Process., 9*, 301–315.

Fleet, D., Jepson, A., & Jenkin, M. (1991). Phase-based disparity measurement. *Comp. Vision, Graphic and Image Proc., 53*, 198–210.

Fleet, D., & Jepson, A. (1990). Computation of component image velocity from local phase information. *Int. J. Comp. Vis., 5*, 77–104.

Franceschini, N. (1996). Engineering applications of small brains. *FED Journal, 7* (Suppl. 2), 38–52.

Franceschini, N., Pichon, J. M., & Blanes, C. (1992). From insect vision to robot vision. *Phil. Trans. R. Soc. Lond. B, 337*, 283–294.

Graziano, M. S., Andersen, R. A., & Snowden, R. J. (1994). Tuning of MST neurons to spiral motions. *J. Neurosci., 14*, 54–67.

Green, P. R., Davies, M. N. O., & Thorpe, P. H. (1992). Head orientation in pigeon during landing flight. *Vision Res., 32*, 2229–2234.

Heeger, D. (1988). Optical flow using spatiotemporal filters. *Int. J. Comp. Vis., 1*, 279–302.

Heeger, D. J., & Jepson, A. D. (1990). Visual perception of three-dimensional motion. *Neural Comp., 2*, 129–137.

Hildreth, E. C., & Koch, C. (1987). The analysis of visual motion: From computational theory to neuronal mechanisms. *Annu. Rev. Neurosci., 10*, 477–533.

Horn, B. K. P., & Schunck, B. (1981). Determining optical flow. *Artif. Intell., 17*, 185–203.

Koenderink, J. J. (1986). Optic flow. *Vision Res., 26*, 161–180.

Lappe, M., Bremmer, F., Pekel, M., Thiele, A., & Hoffmann, K. P. (1996). Optic flow processing in monkey STS: A theoretical and experimental approach. *J. Neurosci., 16*, 6265–6285.

Little, J. J., & Verri, A. (1989). Analysis of differential and matching methods for optical flow. *IEEE Proc. of Visual Motion Workshop* (pp. 173–179).

Longuet-Higgins, H. C., & Prazdny, K. (1980). The interpretation of a moving retinal image. *Proc. Roy. Soc. Lond., B-208*, 385–397.

Lucas, B. D., & Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *DARPA Image Understanding Workshop* (pp. 121–130).

Mahowald, M., & Douglas, R. (1991). A silicon neuron. *Nature, 354*, 515–518.

Matthies, L., Kanade, T., & Szeliski, R. (1989). Kalman filter–based algorithms for estimating depth from image sequences. *Int. J. Comp. Vis., 3*, 209–236.

Marr, D., & Poggio, T. (1976). Cooperative computation of stereo disparity. *Science, 194*, 283–287.

Nelson, R. C., & Aloimonos, J. (1989). Using flow field divergence for obstacle avoidance in visual navigation. *IEEE Trans. PAMI, 11*, 1102–1106.

Opara, R., & Wörgötter, F. (1998). A fast and robust cluster update algorithm for image segmentation in spin-lattice models without annealing—Visual latencies revisited. *Neural Comp., 10*, 1547–1566.

Poggio, G. F., & Poggio, T. (1984). The analysis of stereopsis. *Annu. Rev. Neurosci., 7*, 379.

Poggio, T. A., Torre, V., & Koch, C. (1985). Computational vision and regularization theory. *Nature, 317*, 314–319.

Prazdny, K. (1980). Egomotion and relative depth map from optical flow. *Biol. Cybern., 36*, 87–102.

Pardo, F. (1994). *Development of a retinal image sensor based on CMOS technology* (Tech. Rep.). Genova, Italy: Laboratory for Integrated Advanced Robotic, University of Genova. Available online at: http://afrodite.lira.dist.unige.it:81/LIRA/expsetup/ccd.html and http://afrodite.lira.dist.unige.it:81/LIRA/expsetup/retina.html.

Qian, N. (1997). Binocular disparity and the perception of depth. *Neuron, 18*, 359–368.

Sanger, T. D. (1988). Stereo disparity computation using Gabor filters. *Biol. Cybern., 59*, 405–418.

Schwartz, E. L. (1977). Spatial mapping in primate sensory projection: Analytic structure and relevance to perception. *Biol. Cybern., 25*, 181–194.

Schwartz, E. L. (1980). Computational anatomy and functional architecture of striate cortex: A spatial mapping approach to perceptual coding. *Vision Res., 20*, 645–669.

Tistarelli, M., & Sandini, G. (1993). On the advantages of polar and log-polar mapping for direct estimation of time-to-impact from optical flow. *IEEE Trans. PAMI, 15*, 401–410.

Ullman, S. (1979). The interpretation of structure from motion. *Proc. R. Soc. London Ser. B, 203*, 405–426.

Wagner, H. (1986). Flight performance and visual control of flight of the free-flying housefly (*Musca domestica l.*). I. Organization of the flight motor. *Phil. Trans. R. Soc. Lond., B-312*, 527–551.

Wallman, J., & Letelier, J.-C. (1993). Eye movements, head movements and gaze stabilization in birds. In H. P. Zeigler & H. J. Bischof (Eds.), *Vision, brain and behavior in birds*. Cambridge, MA: MIT Press.

Wang, R. Y. (1996). A network model for the optic flow Computation of the MST neurons. *Neural Networks, 9*, 411–426.

Yuille, A. L., & Ullman, S. (1987). *Rigidity and smoothness of motion (A. I. Memo No. 989)*. Cambridge, MA: Artificial Intelligence Laboratory, Massachusetts Institute of Technology. Available online at: http://www.ai.mit.edu/publications/bibliography/BIB-online.html.

Received April 28, 1997; accepted May 14, 1998.