NASA-Ames Research Center
Aircraft Guidance and Navigation Branch
Mail Stop 210-9
Moffett Field, CA 94305-1000

Obstacle Detection by Recognizing Binary Expansion Patterns

Yoram Baram* and Yair Barniv†

September 20, 1993

Abstract

This paper describes a technique for obstacle detection, based on the expansion of the image-plane projection of a textured object, as its distance from the sensor decreases. Information is conveyed by vectors whose components represent first-order temporal and spatial derivatives of the image intensity, which are related to the time to collision through the local divergence. Such vectors may be characterized as patterns corresponding to "safe" or "dangerous" situations. We show that the essential information is conveyed by single-bit vector components, representing the signs of the relevant derivatives. We use two recently developed, high capacity classifiers, employing neural learning techniques, to recognize the imminence of collision from such patterns.

*Y. Baram is with the Department of Computer Science, Technion, Israel Institute of Technology, Haifa 32000, Israel. He is also associated with the NASA Ames Research Center, Moffett Field, CA 94035.

†Y. Barniv is with the NASA Ames Research Center, Moffett Field, CA 94035.

1 Introduction

Obstacle detection is an area of considerable interest in such applications as rotorcraft nap-of-the-earth navigation and spacecraft landing. There are mainly two possible approaches to the associated vision problem, one based on the optical flow resulting from the motion of a single image sensor, the other based on stationary stereo. Both approaches have been taken in previous works on passive ranging (*e.g.*, [1, 2, 3, 4]). In this paper we take the single sensor, or monocular vision approach.

The motion of an imaging sensor causes each point of the scene to describe a time trajectory in the image plane. The trajectories of all image points constitute the optical flow. An imaging sensor, such as a TV camera or a Forward-Looking Infra-Red (FLIR), is typically used as the source of optical flow data. Similarly to other passive-ranging methods, we assume that the scene and its illumination sources are temporally stationary (see [5]).

The optical flow at any given point in the image plane may consist of three kinds of motion: lateral translation, expansion (or divergence), and rotation (or curl). When considering the vicinity of any given point, its time evolution can be described approximately by an affine transformation which, in the most general case, is defined by six parameters: four belonging to the 2×2 multiplying matrix, and two belonging to the vector of lateral translation. Most depth-estimation methods, such as those described in [6, 7], make use of the lateral motion alone. Two basic limitations are implicit in these methods. They perform poorly in the image plane close to the focus of expansion, and they can only use a relatively short distance between frames. As shown in earlier work [8], the depth estimation error is inversely proportional to this distance.

The idea of using divergence (or expansion) as a source of depth information is not new. The works of Longuet-Higgins and Prazdny [9], Prazdny [10, 11], Koenderink [12], Koenderink and van Doorn [13, 14], and Nelson and Aloimonos [15] elaborate extensively on this subject. An extension of the local divergence approach was recently reported in [16], where a diffusion process is used to roughly delineate objects and "paint" them according to the imminence of collision. This technique is especially useful for objects that have well-defined contours.

In this work we do not try to estimate the numerical depth value, but, rather, we try to classify an obstacle as being "safe" or "dangerous", using a pattern-recognition approach. We rely on the objects being textured, and extract information from local expansion, as illustrated by Figure 1. We look for patterns associated with high or low values of the local divergence. Since the latter is calculable from the image-plane temporal and spatial derivatives, we use those as the components of the pattern vector. Reducing these components to their sign function, we obtain binary pattern vectors which prove to be sufficient for our obstacle detection purposes. We employ two recently developed techniques, reported

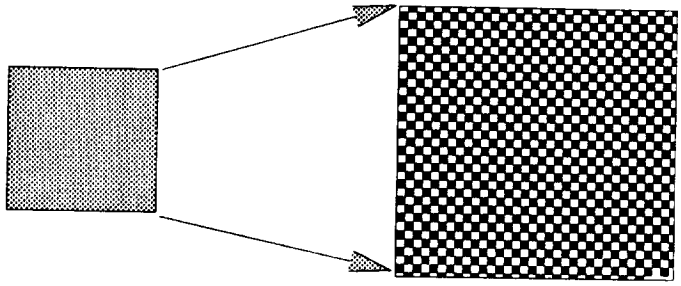

Figure 1: Texture expansion

in [17] and [18], for classifying the pattern vectors as representing safe or dangerous situations. These techniques are based on expanding the patterns into high dimensional space for high storage capacity, and applying learning techniques, previously developed in the neural networks literature, to the resulting sparse representations. The first technique, employing Rosenblatt's learning rule, offers relatively accurate results at the cost of relatively slow computation, while the other, employing the so-called Hebbian learning rule, which is less accurate, is considerably faster.

2 The divergence equations

In this section we present the mathematical background necessary to understand the particular choice of variables for the pattern vectors used by our recognition method.

Denoting by x and y the cartesian coordinates of points in the image plane, the divergence of the image-plane velocity vector (also referred to as the optical flow), $\mathbf{v} = [v_x\ v_y]^T$, at some given point $[x\ y]$ is known to satisfy [16]

$$\nabla \mathbf{v} = \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} = 2/\tau \tag{1}$$

where $\nabla = [\frac{\partial}{\partial x}\ \frac{\partial}{\partial y}]$ and τ is the time to collision. The velocity divergence, then, provides what we are looking for.

Let us now examine what measurements and relationships are available for calculating the divergence from its definition. The brightness constancy equation ([5]) is

$$\frac{\partial I}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}t} + \frac{\partial I}{\partial y}\frac{\mathrm{d}y}{\mathrm{d}t} = -\frac{\partial I}{\partial t}\ , \tag{2}$$

where I denotes brightness, or gray level, and t denotes time. In vector form

$$\mathbf{v}^T \nabla \mathbf{I} = -\frac{\partial I}{\partial t} \tag{3}$$

where $\nabla \mathbf{I} = \left[\frac{\partial I}{\partial x} \ \frac{\partial I}{\partial y}\right]$. Equation (3) does not have a unique solution for the velocity vector; it is satisfied by any vector having a component $\mathbf{v}_I$ parallel to $\nabla \mathbf{I}$. This is the "aperture problem". Solving for this component, we have

$$\mathbf{v}_I = -\frac{\partial I/\partial t}{|\nabla \mathbf{I}|^2}\nabla \mathbf{I} \tag{4}$$

where $|\cdot|$ denotes the length of a vector.

Since the direction of the local edge is generally unrelated to the local velocity vector, we cannot get the latter from (4). One possible way to circumvent both the aperture problem and the random nature of the edges is to apply (4) at least twice to close–by local edges, assuming that they experience the same $\mathbf{v}$ but have different orientations. Rewriting (4) twice for two such edges yields two equations with two unknowns — the two components of $\mathbf{v}$ itself. Taking one step further, the same equation can be applied many (say n) times and the two velocity components for some image point can be obtained from a least-squares solution of this equation set, which may be written in matrix form as

$$A[v_x \ v_y]^T = B \tag{5}$$

where

$$A \triangleq \begin{bmatrix} \left(\frac{\partial I}{\partial x}\right)_1 & \left(\frac{\partial I}{\partial y}\right)_1 \\ \left(\frac{\partial I}{\partial x}\right)_2 & \left(\frac{\partial I}{\partial y}\right)_2 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ \left(\frac{\partial I}{\partial x}\right)_n & \left(\frac{\partial I}{\partial y}\right)_n \end{bmatrix} \quad \text{and} \quad B \triangleq -\begin{bmatrix} \left(\frac{\partial I}{\partial t}\right)_1 \\ \left(\frac{\partial I}{\partial t}\right)_2 \\ \cdot \\ \cdot \\ \cdot \\ \left(\frac{\partial I}{\partial t}\right)_n \end{bmatrix} \tag{6}$$

The least-squares solution of (5) is

$$[v_x \ v_y]^T = \left(A^T A\right)^{-1} AB \tag{7}$$

Equation (7) has a unique solution if and only if the rank of A is 2, which means that at least two edges are oriented differently. The practical problems with such an approach are that it depends on the successful association of all the n points with a single object, and that the derivatives used in the solution are generally noisy. To suppress noise, one would like to use a large number of equations, but this entails identifying a large image area as belonging to the same object through some form of image segmentation. Since methods of image segmentation have their own pitfalls, a practical approach would be to assume that all points within a window of a certain size experience the same $\mathbf{v}$.

In this work we suggest an approach which, rather than solving equations, regards the relevant information as a pattern to be associated with either small or large divergence

values. The question is, what kind of information should constitute a pattern. Suppose that $\mathbf{v}$ coincides with $\mathbf{v}_I$. Applying (1) to (4), we have

$$\nabla \mathbf{v}_I = -\frac{\partial}{\partial x}\left(\frac{\partial I/\partial t}{|\nabla \mathbf{I}|^2}\frac{\partial I}{\partial x}\right) - \frac{\partial}{\partial y}\left(\frac{\partial I/\partial t}{|\nabla \mathbf{I}|^2}\frac{\partial I}{\partial y}\right) \tag{8}$$

which indicates that the relevant information for calculating divergence is captured by the first and second spatial and temporal derivatives of the image light intensity (represented by gray levels), that is by $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$, $\frac{\partial I}{\partial t}$, $\frac{\partial^2 I}{\partial x^2}$, $\frac{\partial^2 I}{\partial y^2}$, $\frac{\partial^2 I}{\partial x \partial y}$, $\frac{\partial^2 I}{\partial x \partial t}$, $\frac{\partial^2 I}{\partial y \partial t}$. Notice that these derivatives are required at the image point under consideration. Alternatively, it can be seen that only first spatial and temporal derivatives of I are required in the least squares solution represented by (7). However, these derivatives are required at several points in the neighborhood of the image point.

Guided by the latter observation, we chose to use the three first derivatives at $n = 13$ points of a grid, centered at the desired image point, as the components of the pattern vector on which classification is to be performed. Using noisy derivatives and real-valued pattern vectors, which do not lend themselves easily to classification, presents a potential difficulty. In this work we show that, for classification purposes, the real-valued components of the derivative vectors can be reduced to binary ones, representing the signs of those derivatives. The input vector to the classification algorithm will consist, then, of 39 binary components.

As we have shown in this section, the direct solution of the divergence equations is quite involved. Our experience shows that a numerical solution of equations (1) and (7) does not yield satisfactory results. It is therefore remarkable that the neural classifiers, described next, were able to produce correct detection results with a high rate of success, using the same information.

3 The pattern recognition algorithms

We employ two recently proposed classifiers, each offering a certain advantage over the other. The first, reported in [17], is based on transforming the input data into sparse binary internal representations, and learning the class assignment of the latter by applying Rosenblatt's rule. The second, reported in [18], is also based on sparse binary internal representation of the input data, but the learning rule is Hebbian. The first method, employing discriminatory storage of only those input vectors which are classified incorrectly, may require many training epochs. The second, employing indiscriminatory storage of the training input–output pairs, requires a single passage over the training set. Consequently, the second method is usually considerably faster than the first, which will, however, produce more accurate results, given a sufficiently long processing time. These relative advantages of the two techniques apply to both their hardware and software implementations.

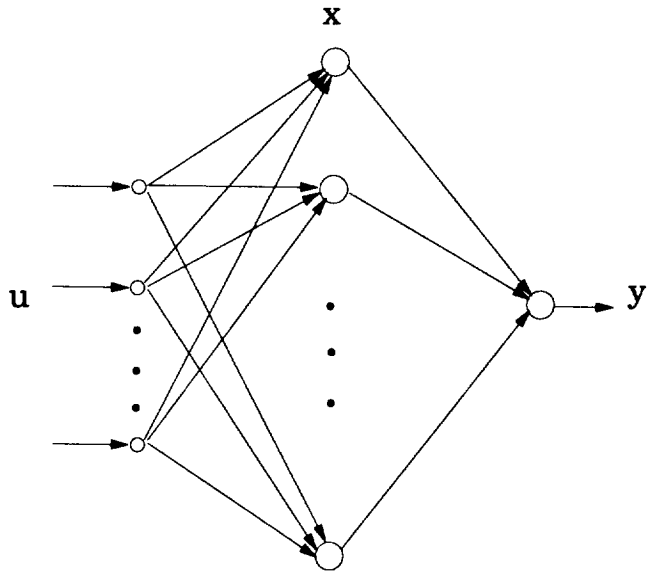

Figure 2: Classifier structure.

Both methods have the structure depicted in Fig. 2. A given 39-dimensional input vector u, whose components are $+1$ or -1, is transformed into a sparse N-dimensional binary vector x, whose components are 0 or 1, by a layer of N *internal* cells. The size N of the internal layer is to be determined. Each internal cell performs the function

$$x_i = \begin{cases} 1 & \text{if } (v^{(i)}, u) \geq t \\ 0 & \text{if } (v^{(i)}, u) < t \end{cases} \tag{9}$$

where $v^{(i)} \in \{\pm 1\}^n$ is a vector with randomly chosed binary components, $(v^{(i)}, u)$ denotes the inner-product between the two vectors and t is a positive scalar. The resulting vector x is the internal representation of the input vector u. The input weights of the internal cells define the centers of spheres of Hamming radius $r = 0.5(n - t)$ in $\{\pm 1\}^n$, so that the cell's output is 1 when the input falls within the sphere and 0 otherwise. (It should be noted that the activation radius r is selected first as an integer, then t is obtained as $t = n - 2r$).

The output function, representing the class assignment of u, is

$$y = \begin{cases} 1 & \text{if } (w, x) \geq 0 \\ 0 & \text{if } (w, x) < 0 \end{cases} \tag{10}$$

where w_j, the j'th element of w, is the weight of the connection between the j'th internal cell and the output, to be determined by the learning process.

The centers of the spheres represented by the internal cells are chosen randomly. The spheres should cover the input space with high probability. The minimal number of spheres that cover the space $\{\pm 1\}^n$ with probability of, at least, $1 - 2^{-\epsilon n}$, was derived in [19] as

$$N \geq \frac{(1+\epsilon) n 2^n \ln 2}{S_n(r)} = (1+\epsilon) n 2^{n[1-h_2(\rho)]+O(\log n)} \ln 2 \tag{11}$$

where

$$S_n(r) = \sum_{i=1}^{r} \binom{n}{i} = 2^{nh_2(\rho)+O(\log n)} \tag{12}$$

is the volume of a sphere of radius r in $\{\pm 1\}^n$, $\rho = n/r$, $h_2(\rho) = -\rho \log_2(\rho) - (1-\rho)\log_2(1-\rho)$, and $O(\log n) < c \log n$ for some positive scalar c.

The size of the minimal covering code for $n = 39$ and for covering probability 0.99 is plotted in Fig. 3 against the relative activation radius ρ, in terms of $\log N$. It can be seen that, for $\rho = 0.3$, which was found to be adequate from discretization considerations [18], we have $\log N = 0.3755$, hence, $N = 5600$. These are the values that will be used by the proposed classifiers.

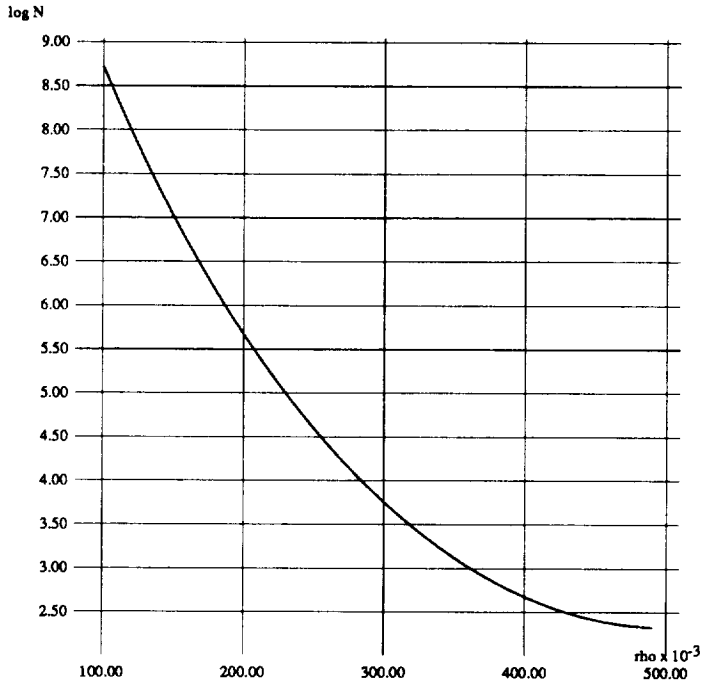

Figure 3: Minimal covering code size for input dimension $n = 39$.

Rosenblatt learning

Given a training set A of pairs $u^{(i)}, y^{(i)}$, $i = 1, \ldots, M$, where $u^{(i)}$ is an input vector and $y^{(i)}$ is the correct output for $u^{(i)}$, classification is learnt by applying Rosenblatt's perceptron learning rule to the corresponding pairs of internal representations and outputs: Start with $w_j = 0$, $j = 1, \ldots, N$. Pick a vector $u^{(i)}$ from A and present it as an input. If the output y agrees with $y^{(i)}$, leave w unchanged and proceed to the next training input. If $y = -1$ and $y^{(i)} = 1$, add the internal representation to the weights vector. If $y = 1$ and $y^{(i)} = -1$, subtract the internal representation from the weights vector. This learning rule may be

written as

$$w_j = \begin{cases} w_j + x_j y^{(i)} & \text{if } y \neq y^{(i)} \\ w_j & \text{if } y = y^{(i)} \end{cases} \tag{13}$$

The perceptron convergence theorem [20] implies that this learning process will converge to a final value of the weights vector w, so that the internal representations of the training vectors are classified correctly, provided that their assignment is not ambiguous and that such a vector exists. The existence of such a vector is guaranteed if the internal representations are linearly independent. It may, however, take many iterations for the algorithm to converge to final values, which stands in contrast to the storage method described next.

Hebbian storage

The second learning method calculates the connection weights between the internal cells and the output by the Hebbian rule:

$$w_j = \sum_{i=1}^{M} x_j^{(i)} y^{(i)} \quad j = 1, \ldots, N \tag{14}$$

This is an indiscriminatory correlative storage. Our input–space covering requirement guarantees that, given sufficiently many training pairs, the entire input space will be represented by the external weights, so as to associate every input vector with a class. It is shown in [18] that the learning capacity of this classifier is $2N$, and that it has a substantial generalization capability. It takes a single passage over the training set (a single training epoch) and is, consequently, considerably faster than Rosenblatt's learning. On the other hand, the effective discretization of the input space generated by this method is coarser, and, consequently, it will often produce less accurate result than the one based on Rosenblatt learning.

4 Simulation

4.1 Creating the pattern vectors

A scenario simulation has been developed to generate the required samples. It consists of a helicopter flying along a pre-chosen trajectory, including possible maneuvers (that is, pitch, yaw and roll components of motion), towards a vertical wall normal to the line of sight. The wall has a textured surface, generated by filtering white Gaussian noise through a Gaussian-shaped point-spread-function having a 2-pixel spatial correlation width. The simulation program allows us to control all the parameters of interest, such as the texture fineness, the dynamic range of the gray-levels and the distance from the focus of expansion. Figure 4 shows frames 12 and 24 of the simulated textured wall as it is seen in forward flight.

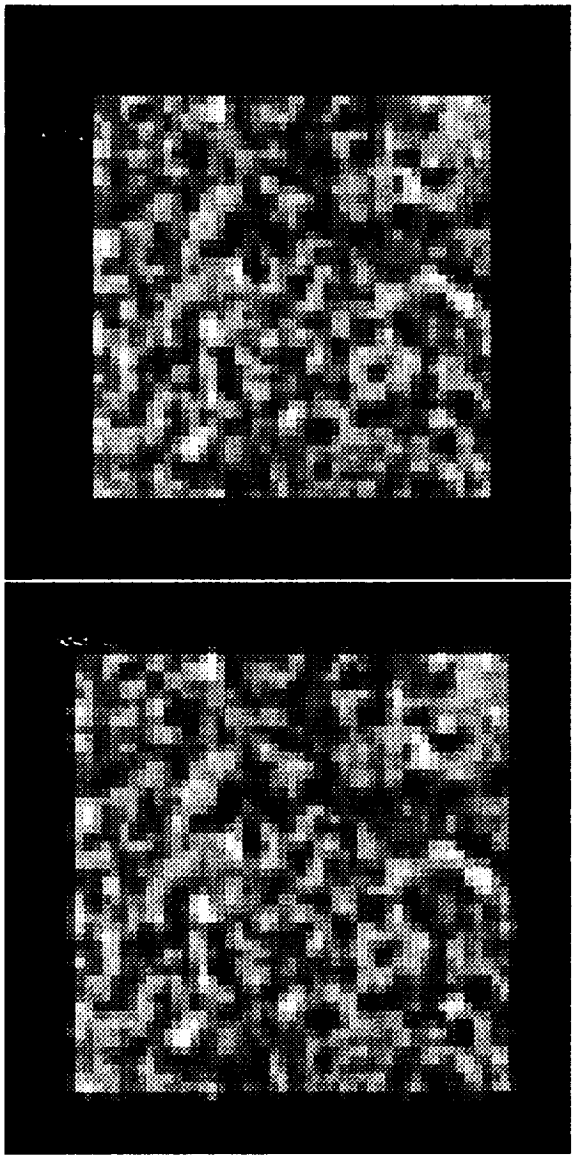

Figure 4: Frames 12 and 24 of simulated textured wall as seen in forward flight

We scan this wall in order to pick up samples for training the classifier. A single sample vector is composed of the first spatial and temporal derivatives, $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$, $\frac{\partial I}{\partial t}$, for 13 grid points centered at a given sample point. The grid arrangement is shown in Figure 5. We note that, since the first spatial derivatives are given for each of the grid points, information about the second spatial derivatives at the center point is also implicit in the data. Next, we replace the real values of the 39 vector components by their signs, representing a negative sign by -1 and a positive sign by 1. To each sample vector we attribute an output value of -1 if it corresponds to a safe distance and $+1$ if it corresponds to an unsafe one.

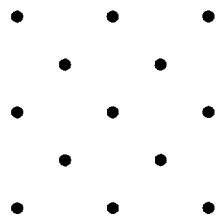

Figure 5: Grid points at which first derivatives are taken to construct a pattern vector for the center point

The simulated flights, at a speed of 0.5 m/f (meter/frame), start 150 m from the textured wall with the camera initially pointing to the wall center. The flights end at a distance of 40 m from the wall. Maneuvering flights exercise a roll of 0.02 rad/f in addition to pitch and yaw motions at rates of 0.0005 rad/f. In one set of experiments, the close (or dangerous) zone, was defined as the points whose distances from the wall are between 40 m and 70 m, while the far (or safe) zone, as the points at distances beween 110 m and 150 m. In another set, points in the range 40 – 80 m were considered close and those between 90 m and 130 m were considered far. The second set, having a narrow deadband of only 10 m between the two zones, is considerably more challenging than the first, having a 40 m deadband.

Each simulation generates 220 images of 128 $\times$ 128 pixels, separated by a distance of 0.5 m normal to the wall. Sampling considerations suggest that image plane points having an absolute velocity value greater than 1.5 pixel per frame be eliminated. This effectively confines the sample points to a circle centered at the focus of expansion whose radius decreases with the distance from the wall.

The nature of the simulation is such that data points corresponding to the far zone are created before those corresponding to the close one. Since the first classification method is sensitive to the ordering of the data, we apply a mixing algorithm that rearranges the data in a random order. The last step in preparing the data is dividing it into two exclusive sets: the first, consisting of about 70 percent of the sample points, is used for training the classifiers, while the other is used for testing their performance on new unlearnt data.

We applied the two classification algorithms described in the previous section to each

of the simulated cases. Allowing 10 training epochs for the Rosenblatt learning algorithm, the classifier based on this method was about 10 times slower than the one employing Hebb learning, but it produced more accurate results in most cases. We shall refer to the two algorithms as the slow one and the fast one, respectively.

4.2 Experimental results

Experiment 1

We first simulated a flight path normal to the wall without maneuvers. The close and the far zones are separated by a deadband of 40 m. The simulation created 10460 input–output pairs, of which 3157 were close and 7303 were far. After mixing the data, we selected 7500 pairs for training and 2960 pairs for testing.

The average error was 0.0929 for the fast algorithm and 0.0475 for the slow one.

Experiment 2

Reducing the deadband between the two zones to 10 m in a nonmaneuvering flight, 9985 samples were created, of which 3824 were close and 6161 were far. The training set consisted of 7000 points and the testing set of 2985 points.

The fast algorithm yielded an average error of 0.1272, and the slow one an average error of 0.0928.

Experiment 3

Simulating a maneuvering flight with a 40 m deadband between the two zones, we produced 3909 samples, of which 1390 were close and 2519 were far. The training set consisted of 3000 points and the test set of 909 points.

The results were 0.1418 for the fast algorithm and 0.0990 for the slow one.

Experiment 4

Repeating the maneuvering flight with a 10 m deadband, the total number of samples generated was 5738, of which 2132 were in the close zone and 3606 in the far one. The training set consisted of 4000 points and the test set of 1738 points.

The fast classifier yielded an average error of 0.1944 for this case and the slow one an error of 0.0990.

Experiment Number 5

The purpose of this experiment was to check the classifier's response to motion consisting of stricly lateral maneuvers. Such maneuvers present no danger of collision, and should be classified as safe. Training was performed on the dataset of experiment 3. The test data consisted of points, all representing a lateral maneuver, at a distance of 75 m from the wall.

The fast algorithm produced an average classification error of 0.0435 while for the slow one the error was 0.0570. This shows that the classification methods are reliable in the sense that they do not produce a substantial false alarm, and that both classifiers perform well in non-ambiguous situations.

Changing the grid

Reducing the horizontal and the vertical distance between grid points from 3 to 2 pixels has caused the fast method to produce less accurate results. Increasing this distance to 4 pixels reduced the error, but increasing it to 5 increased it again. (For instance, the average error produced by the fast method for experiment 4 increased from 0.1944 to 0.2382 when the distance was reduced from 3 to 2 pixels. The average error was 0.1655 for a distance of 4 pixels and 0.1740 for a distance of 5 pixels). However, the performance of the slow method, which is more accurate, did not follow the same trend (the average error for a grid distance of 4 pixels was 0.1584, higher than the value of 0.0990 obtained for a distance of 3 pixels). We conclude that the optimal distance between grid points was 3 for our experiment.

5 Conclusion

We have presented a method for obstacle detection from optical flow information employing two recently proposed classifiers. It is based on the local divergence of the optical flow in the image plane, which is inversely proportional to the time to collision. The input to the classifiers consists of binary pattern vectors, whose components are the signs of the first spatial and temporal derivatives of the light intensity at several points in the neighborhood of a given image point. The classifiers, detailed in [17] and [18], employ high-dimensional sparse binary internal representations, and either Rosenblatt or Hebbian learning of the class assignment. Performance was tested by several scenario simulations, including direct approach, maneuvering approach and strictly lateral motion. The rate of success for each of these cases was over 90 percent.

References

[1] L.H. Wegner. On the accuracy analysis of airborne techniques for passively locating electromagnetic emitters. Report R-722-PR AD 729 767, NTIS ASTIA D.C., Rand

Corp, 1971.

[2] J.L. Poirot and G.V. McWilliams. Application of linear statistical models to radar location techniques. *IEEE Trans. on Aerospace and Electronic Systems*, 10(6):830–834, November 1974.

[3] D.J. Torrieri. Statistical theory of passive location systems. *IEEE Trans. on Aerospace and Electronic Systems*, 20(2):183–198, March 1984.

[4] M. Gavish and E. Fogel. Effect of bias on bearing-only target location. *IEEE Trans. on Aerospace and Electronic Systems*, 26(1):22–25, January 1990.

[5] B.K.P. Horn and B.G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(3):185–203, August 1981.

[6] B. Sridhar and A.V. Phatak. Simulation and analysis of image-based navigation system for rotorcraft low-altitude flight. In *Proceedings of the AHS Meeting on Automation Application for Rotorcraft*, Atlanta, GA, April 1988.

[7] B. Sridhar, V.H.L. Cheng, and A.V. Phatak. Kalman filter based range estimation for autonomous navigation using imaging sensors. In *Proceedings of the 11th Symposium on Automatic Control in Aerospace*, Tsukuba, Japan, July 1989.

[8] Y. Barniv. Error analysis of combined optical-flow and stereo passive ranging. *IEEE Trans. on Aerospace and Electronic Systems*, to be published, October 1992.

[9] H. C. Longuet-Higgins and K. Prazdny. The interpretation of a moving retinal image. *Proc. R. Soc., London B*, 208:385–397, 1980.

[10] K. Prazdny. Determining the instantaneous direction of motion from optical flow generated by a curvilinear moving observer. *Computer Vision, Graphics, and Image Processing*, 17:238–248, 1981.

[11] K. Prazdny. Egomotion and relative depth map from optical flow. *Biological Cybernetics*, 36:87–102, 1980.

[12] J. Koenderink. Optic flow. *Vision Research*, 26(1):161–180, 1986.

[13] J.J. Koenderink and A.J. van Doorn. Invariant properties of the motion parallax field due to the movement of rigid bodies relative to an observer. *Optica Acta*, 22(9):773–791, 1975.

[14] J.J. Koenderink and A.J. van Doorn. Local structure of movement parallax of the plane. *Journal of Optical Society of America*, 66(7):717–723, July 1976.

[15] R. C. Nelson and J. Aloimonos. Obstacle avoidance using flow field divergence. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 11(10):1102–1106, 1989.

[16] D. L. Ringach and Y. Baram. A diffusion mechanism for obstacle detection from size-change information. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15(12), 1993.

[17] Y. Baram. Classification of binary and real bounded data by learning sparse representations. Technical report, CIS Report No. 9320, Technion, Israel Institute of Technology, Department of Computer Science, Center for Intelligent Systems, Haifa, Israel, 1993.

[18] Y. Baram. Pattern correction and association and sequence regeneration by correlative memory. Technical report, CIS Report No. 9313, Technion, Israel Institute of Technology, Department of Computer Science, Center for Intelligent Systems, Haifa, Israel, 1993.

[19] P. Delsarte and P. Piret. Do most binary linear codes achieve the goblick bound on the covering radius? *IEEE Trans. on Information Theory*, IT-32(6):826 - 828, November 1986.

[20] M.L. Minsky and S. Papert. *Perceptrons.* MIT, Cambridge, 1988.

List of Figures