

Object Recognition

Jeremy T. Newkirk
Intelligent Systems
February, 28 2006

Abstract

This paper discusses some of the difficulties encountered with object recognition and some of the methods used to overcome these difficulties. There are methods presented that are intended to distinguish objects within a cluttered environment and others for determining what the objects are. These methods are discussed in detail .

1 Introduction

The advancement of object recognition would be a great benefit to many different engineering projects incorporating vision. The ability to recognize objects would allow a computer to analyze its environment and accomplish many different tasks. These tasks could include autonomous navigation, manipulation of objects, surveillance and numerous others. Object recognition is much more difficult than it first appears to be though. Humans use it every day so it seems that it would be easy to teach a computer to do the same thing. However, people perfect their ability to recognize objects a countless number of times every day, yet people still make mistakes trying to recognize objects. This paper will explain some of the difficulties encountered in object recognition and what is being done to overcome them.

2 Difficulties with Object Recognition

Some of the major problems with object recognition include: distinguishing an object from its surroundings and having to compare objects with previously stored models. This section will explain these difficulties in more detail. The following sections will explain a number of different methods being tested in an attempt to overcome these problems.

2.1 Distinguishing Objects

Distinguishing an object from its surroundings can be very difficult depending on the environment the object is in. If the object is sitting on a table by itself then the problem is much simpler. However, even this problem can be somewhat difficult. For example, if the table is white and the object you are trying to detect with your camera is a white piece of paper laying flat on the table, then the problem is not so simple. In this case the paper blends in very well with the background. If the table was black then there would be a large contrast between the table and paper making it much easier to detect. Now imagine how much more difficult this problem becomes when a large number of objects are introduced into the camera image. Objects with all different shapes, sizes, colors and textures. Some of the objects may seem to blend together in a camera image making it very difficult for a computer to distinguish objects from one another or possibly even the background. If a computer can't separate the objects from one another then it has no chance of determining what the objects are.

2.2 Recognizing Objects

Even if a computer is able to separate all of the objects in a given image there is still the problem of determining what the object is. One method that could be used to overcome this problem would be to compare the objects in the image to stored models. This is not a very difficult problem if the object or objects in the image are known and the computer has models to use for comparison. However, if the environment is not known specifically then an enormous number of object models

must be stored in the computer. However, if the environment is unknown then there is no way to be sure that enough models are stored. A large number of models also means that the computer may take a long time comparing the object to all of the stored models. Another problem with this is that there are so many variations of some objects. Ball point pens are a good example of this. Some pens have a cap that goes over the ball point to keep the pen from writing unintentionally, while others have a clicker that allows the ball point to retract inside the body of the pen to perform the same function. Some pens are thin while others have a larger diameter. Some use a form of rubber where a person’s fingers normally grip the pen and others just have a plastic shaft that extends the entire length of the pen. They can be made from plastic, metal or a combination of both. This is for something as simple as a pen. There are many objects that have much more complexity. So, the problem is whether or not it is feasible for a person to store all of these different objects with all of their different forms, especially when objects are constantly being upgraded and modified.

3 Object Distinguishing Methods

Some of the methods for distinguishing objects include using spin-images [5], Hybrid Houghfield Neural Networks (HHN) [7], segment-based stereo vision [13] and OMNE-Vision [18]. This paper will discuss the segment-based stereo vision method and the method incorporating spin-images.

3.1 Segment-Based Stereo Vision

The segment-based stereo vision method uses boundary representation to describe a scene. The image is segmented into step edges using a technique developed by [15, 16]. This method uses the following steps:

- Step 1: Calculate edge strength and direction at each pixel $f(x, y)$ in the gray-level image.

$$|e(x, y)| = \sqrt{f_x(x, y)^2 + f_y(x, y)^2}, \tag{1}$$

$$\angle e(x, y) = \tan^{-1}\left(\frac{f_y(x, y)}{f_x(x, y)}\right) + \frac{\pi}{2},$$

where f_x and f_y are the first derivatives in x and y directions respectively. Any gradient operators, such as Sobel, are available.

- Step 2: Extract one-pixel wide edge segments from e by thresholding and thinning based on non-maximal edge suppression.
- Step 3: Extend strong but incomplete edges to form closed regions. First search for a terminal edge element e_0 of a strong edge by raster scanning. Then select an extended edge element that maximizes the function,

$$h(e_0, e_i) = |e_i| \cos(e_i - e_0), \tag{2}$$

from among three edge elements that are almost in the direction of the current edge element e_0 (edge e_5 , e_6 , and e_7 in Figure 1, where h gives a large value when the strength of e_i is large and the direction is the same as e_0).

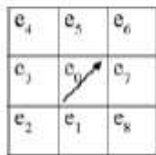


Figure 1: 3×3 window for edge detection. The arrow signifies the direction of e_0

- Step 4: Move the current position to the extended element, and repeat steps 1 through 3 until the selected edge element is an existing strong or extended edge element.

This creates an outline consisting of all the edges found in the image. An example of this is shown in Figure 2. Once this outline is created four kinds of feature points shown in Figure 3 are found, which consist of corner points, inflection points, transition points and branch points. This is important because the segment-based stereo vision algorithm requires a boundary to be broken up into straight, convex or concave segments, which can be achieved through the use of the feature points. These feature points are found using the boundary segmentation algorithm developed by [12]. Once these points are found the 3-D data can be reconstructed using segment-based stereo vision proposed by [6, 17]. The following steps outline this procedure:

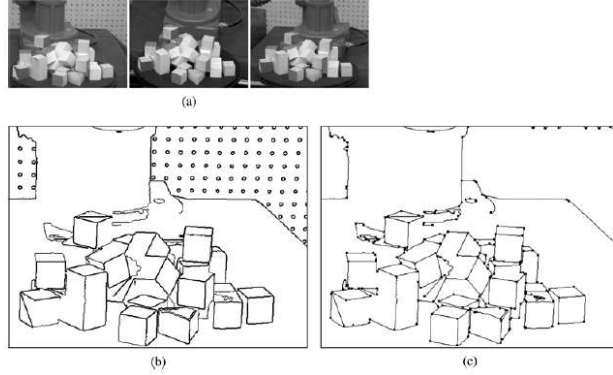


Figure 2: (a) Trinocular stereo images (640×480 pixels, 256 gray-level resolution), (b) Result of segmenting the left image, (c) 2D boundary representation built through small region elimination and boundary segmentation

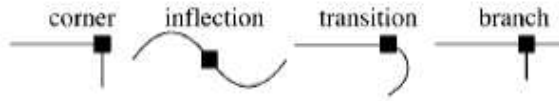


Figure 3: Feature points for boundary segmentation.

- Step 1: Search for correspondence candidates between left and right segments in 2-D boundary representations based on the epipolar constraint of trinocular stereo camera geometry. The candidates are expressed as pairs of boundary representation segments, $[l, r]$, where l and r are segments in the left and right boundary representation respectively. For each pair $[l, r]$, we calculate its *similarity value*,

$$S_{pair}(l, r) \equiv d_{l,r} \left(1 - \frac{\Delta I_{l,r}}{I_{max}} \right), \quad (3)$$

where $d_{l,r}$ is the length of the correspondence portion between segment l and r , $\Delta I_{l,r}$ is the intensity difference between the two segments, and I_{max} is the maximum intensity.

- Step 2: Check the connectivity pairs. Two pairs $[l_1, r_1]$ and $[l_2, r_2]$ are regarded as being connective when l_1 and l_2 are on the same boundary, the distance between the end point of l_1 and the start point of l_2 is small, and the distance between r_1 and r_2 is also small. As a result of checking connectivity for all possible pairs, a directed graph as shown in Figure 4 is obtained.

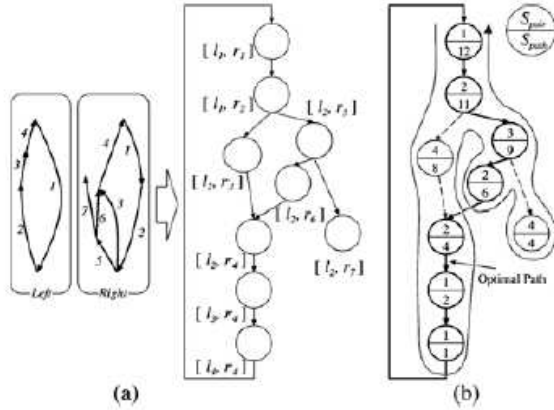


Figure 4: Search for stereo correspondence path. (a) The directed graph. (b) Search procedure for the best path.

- Step 3: Using a dynamic programming technique, search for the optimal correspondence path that maximizes the following global similarity value:

$$S_{path} \equiv \sum_{i \in path} S_{pair}(i). \quad (4)$$

- Step 4: Establish one-to-one correspondences by removing multiple correspondences.
- Step 5: Reconstruct the 3-D scene using parallax between images.

The 3-D position for each boundary feature is stored and the image is reconstructed using these boundary representations. An example of this is shown in Figure 5 by reconstructing the image shown in Figure 2. At this point boundary features are constructed at the feature points. A boundary feature is a pair of unit vectors from a feature point and there are two different types of boundary features:

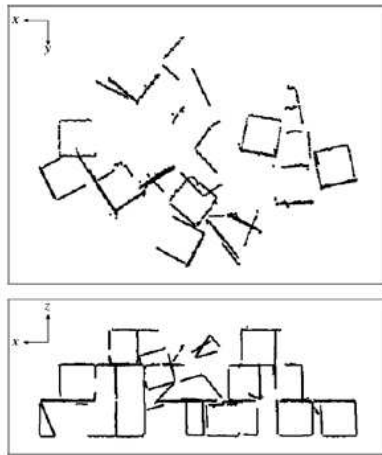


Figure 5: Front and top views of 3D boundary representation reconstructed by the segment-based stereo vision on the VVV system

BF-Vertex: A BF-Vertex consists of two tangent vectors of two adjoining segments from the feature point. The tangent vector is obtained from a line or a circle locally fitted to the segment from the

feature points as shown in Figure 6(a). As many data points as possible are used as long as the fitting error does not exceed a threshold. If they are available, all the points in the segment are employed.

BF-Arc: The orientation of the BF-Vertex at an inflection point or a transition point may be uncertain because the two vectors are linear. Alternatively, when a circle is fitted to a segment better than a line, a BF-Arc is defined at the feature point and two vectors (the normal vector of the plane defined by the circle and a vector directed from the center of the circle to the feature point) as shown in Figure 6(b).

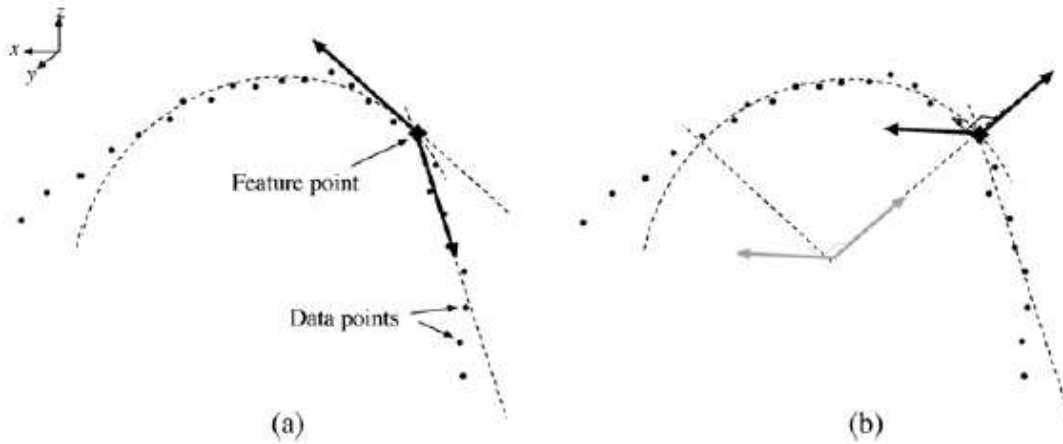


Figure 6: Boundary features. (a) BF-Vertex (b) BF-Arc

These boundary features are then compared to the boundary features of models that are stored in the computer. The contention is that using just the boundary features eliminates the problem of parts appearing to blend together.

3.2 Model Recognition using Spin-Images

This method uses a polygonal mesh that describes the shape of an object and a set of spin-images created from the surface mesh. The spin-images are created from oriented points, which are 3-D points with associated directions. The oriented point is defined at a surface mesh vertex using the position of the vertex and the surface normal at the vertex in 3-D space. An oriented point creates a coordinate frame. A radial coordinate α and an elevation coordinate β are defined as the perpendicular distance to the line through the surface normal and the signed perpendicular distance to the tangent plane defined by the vertex normal and position, respectively. Johnson and Hebert describe the creation of a spin-image as follows:

A 2-D accumulator indexed by α and β is created. Next, the coordinates (α, β) are computed for a vertex in the surface mesh that is within the support of the spin-image (explained below). The bin indexed by (α, β) in the accumulator is then incremented; bilinear interpolation is used to smooth the contribution of the vertex. This procedure is repeated for all vertices within the support of the spin-image. The resulting accumulator can be thought of as an image; dark areas in the image correspond to bins that contain many projected points. As long as the size of the bins in the accumulator is set on order of the median distance between vertices in the mesh [3], the position of individual vertices will be averaged out during spin-image generation. Figure 7 shows the projected

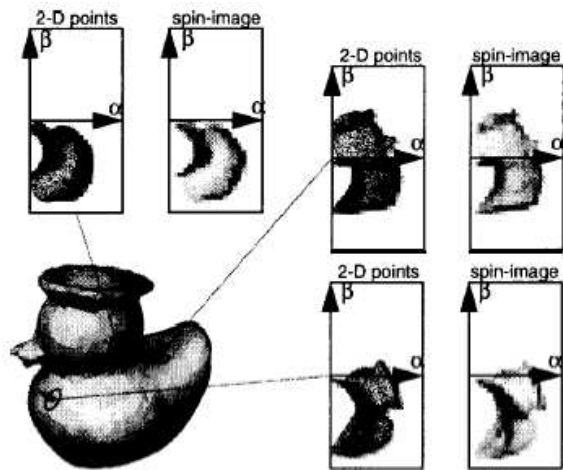


Figure 7: Spin-images of large support for three oriented points on the surface of a rubber duckie model.

α, β coordinates and spin-images for three oriented points on a rubber duckie model. For surface matching, spin-images are constructed for every vertex in the surface mesh.

The support of a spin-image is the part of the surface of an object around the oriented point basis that can contribute points to spin-image generation. There are two parameters that control the spin-image support. The first parameter, *support distance* D_s , sets the maximum distance between the oriented point and a point contributing to the spin-image. Support distance is analogous to window size in 2-D template matching. The second parameter, *support angle* A_s , limits the angle between the normal of the oriented point basis and the normal of points contributing to the spin-image. As shown in Figure 8, by changing the support parameters, spin-images can be smoothly transformed from global to local representations. Spin-images of small support are robust to clutter and occlusion while spin-images of large support are highly discriminating.

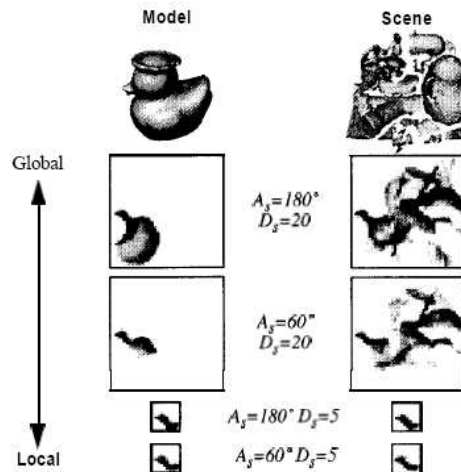


Figure 8: How spin-image generation parameters localize spin-images to reduce the effect of scene clutter and occlusion on matching

Now that a number of spin-images have been created the matching process can begin. Spin-images are compared to one another by computing the correlation coefficients. When spin-images

are highly correlated then a point correspondence between surfaces is established. The point correspondences are then grouped together with outliers being thrown out. These groups are then used to align one surface with another through calculations of rigid transformation. The surface matching method is described in more detail in [4]. The spin-images are then compared with the spin-images of the models stored in the computer to determine what the object is and its orientation.

4 Object Recognition Methods

Numerous methods have been proposed for object recognition. This paper will discuss two such methods: object recognition using neural networks and surface signatures [19] and a parallel-hierarchical neural network [10]. Both of these methods are efficient and robust and are good examples for demonstrating the complexities involved in recognizing objects.

4.1 Object Recognition Using Neural Networks and Surface Signatures

This method begins by establishing "surface signatures" for each 3-D point obtained rather than using the 3-D coordinates of the points. The idea of obtaining a "signature" at each surface point is not a novel approach [5, 11, 1].

Each point P on a free-form surface is defined by its 3-D coordinates and the normal U_p at the patch where P is the center of gravity. As shown in Figure 9 every other point P_i can be characterized by:

1. The distance $d_i = \|P - P_i\|$
2. The angle $\alpha_i = \cos^{-1} \left(\frac{U_p \cdot (P - P_i)}{\|P - P_i\|} \right)$

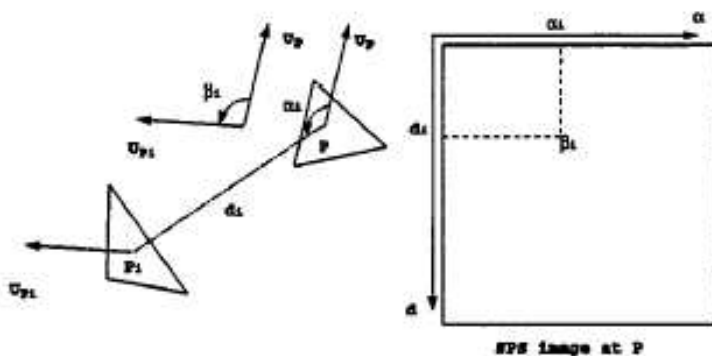


Figure 9: For each point P an SPS image is generated where the image axes are the distance d between P and each other point on the surface and the angle α between the normal at P , U_p and the vector P to each other point. The image encodes the angle β which represents the change in normal at the points from U_p

This creates polar coordinates for each point P_i relative to P of the SPS image. The Cartesian coordinate locations of each point can easily be identified now. The size of the SPS image depends on the size of the object and distance from the camera. For the matching process the images must be normalized, so in this method the object is normalized to the maximum length. This allows the computer to recognize an object regardless of its size or distance from the camera. Next, the angle between the normal of P and the normal of every other point P_i is calculated by $\beta = \cos^{-1}(U_p \cdot U_{p_i})$. Some SPS images taken from a statue and a telephone are shown in Figure 10.

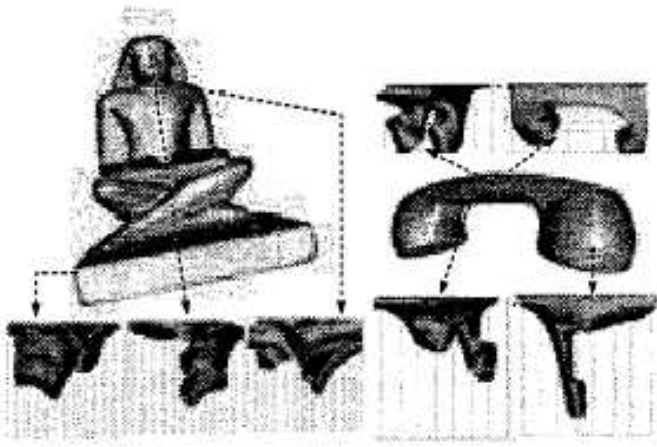


Figure 10: Examples of SPS images taken at different locations. Notice how the image features the curvature information. The dark intensity in the image represents a high curvature seen from the point while the light intensity represents a low curvature. Also notice how the image reflects the point location

Next, this method finds "important points". Finding these points is not a new idea [11, 1, 14]. Most of the points on an object have little information because they have low curvature values. These points are eliminated and the signature images are created for the remaining points. A modified version of *Weingarten Map* [9] is used for curvature estimation of a free-form surface. As shown in Figure 11 the normal to the surface can be obtained for each patch of the surface along with its neighbors. These normals can be used to define the curvature of the surface at the center of mass P by

$$C_S(P) = 1 - \frac{1}{n_g} \sum_{i=0}^{n_g} U_p \cdot U_i \quad (5)$$

where n_g denotes the total number of neighbors to this patch. The point P is considered to be an important point labeled as P_I if $C_S(P) > t_h$ where t_h is a predefined threshold.

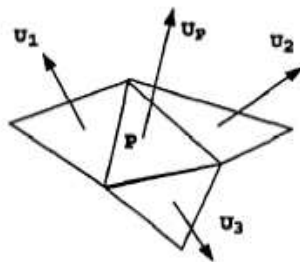


Figure 11: For a free-form surface, each triangular patch is defined by its three points and the normal U_p to the surface at the center of mass P . The patch also may be connected to n_g number of neighboring patches

Two examples of objects and their scanned models are shown in Figure 12. Figure 13 shows the important points of these objects using different values of t_h . This shows that most of the landmark points are found for each of the objects, which should make comparisons to stored object models relatively easy.

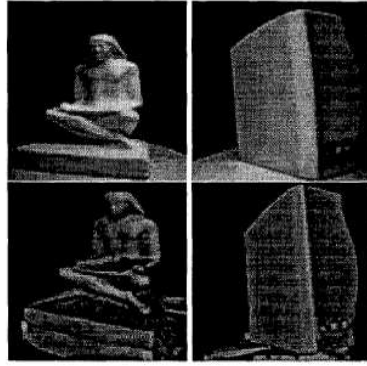


Figure 12: (Top)Two examples of real objects, a statue and a speaker. (Bottom)Rendered views of the scanned 3D model of the objects. The statue 3D model consists of 22541 patches and the speaker 3D model consists of 16150 patches.

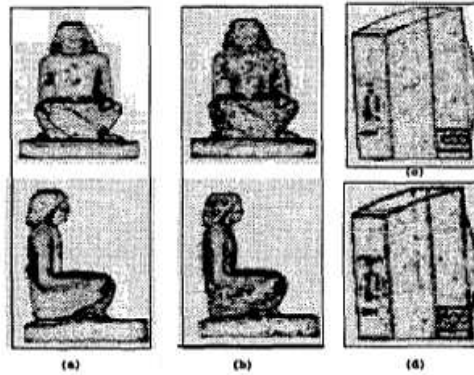


Figure 13: The important points obtained for the statue and speaker models using (a) $t_h = 0.66$, (b) $t_h = 0.1$, (c) $t_h = 0.3$ and (d) $t_h = 0.1$.

Four methods for SPS matching are proposed using neural network implementations. These methods are shown in Figures 14 and 15.

The first method, shown in Figure 14(a), uses the whole SPS image as input and the output are the (x, y, z) coordinates of the point at which the SPS image was generated. The training procedure begins by constructing an input-output map using many SPS images for the object models stored on the computer. Standard error back-propagation learning procedure is used [20]. The matching process gives SPS images from the object to the network which returns the best (x, y, z) coordinates of a point on a stored model. This is done with numerous points and the stored model that has the most matching values is identified. The pose can then be estimated.

The neural network can become very complex if the whole SPS image is used, so the second proposition uses only the horizontal and vertical projections of the SPS images as inputs to the neural network. This is shown in Figure 14(b). This reduces the accuracy of the correspondence, but this can be overcome by using more learning samples.

Sometimes an object cannot be fully seen due to clutter and/or occlusion. This will cause problems with the methods mentioned above, since the number of matching points may not be sufficient. This problem can be solved using the proposed network configuration shown in Figure 15(a). This method uses the entire SPS image as the input to the network as in the first method proposed. However, a rejection/acceptance criteria is built based using the number of matches and the model with the most matches is identified as before. The pose can then be estimated.

The final proposed network configuration is shown in Figure 15(b). This method uses horizontal

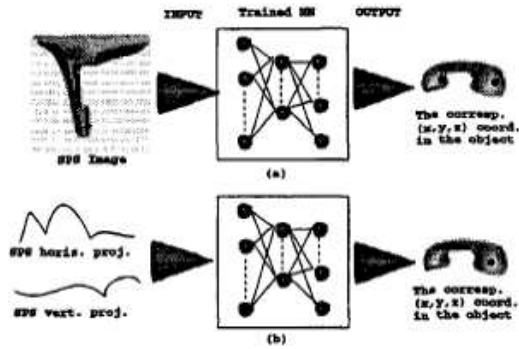


Figure 14: The two different neural network configurations to solve the 3D registration problem using the SPS representation. (a) The whole SPS image is used as an input while the desired response is the (x, y, z) coordinates of the point at which this SPS image was generated. (b) The horizontal and vertical projections of the SPS images are used as inputs to the neural network rather than the whole image.

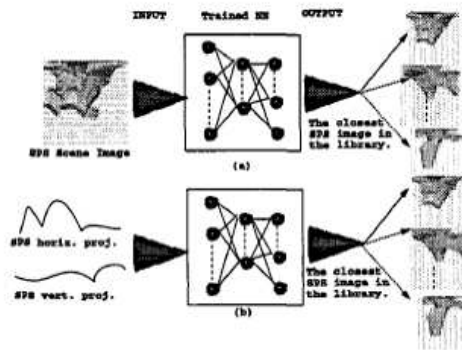


Figure 15: Another two neural network configurations to solve the 3D registration problem using the SPS representation. (a) The whole SPS image is used as an input while the desired response is a matching score to each SPS image in the library. (b) The horizontal and vertical projections of the SPS images are used as inputs to the neural network rather than the whole image.

and vertical projections of the SPS images as in the second proposed method. The idea is the same as in method three.

4.2 Parallel-Hierarchical Neural Network

The structure of the parallel-hierarchical neural network is shown in Figure 16. The network receives an input image and outputs a recognition result. The amount of transformation is also quantified which includes the rotation, scaling or occluded portion of the object. The following will explain how the network works.

4.2.1 Basic Structure

The basic structure of two arbitrary layers are shown in Figure 17. The relationship shown between these two layers applies to any combination of layers except the distributed representation layer and recognition layer. For the distributed representation layer and the recognition layer, each cell plane of the recognition layer only receives input from the corresponding cell plane in the distributed representation layer. The generalized basic structure has cell layers that consist of cell planes arranged

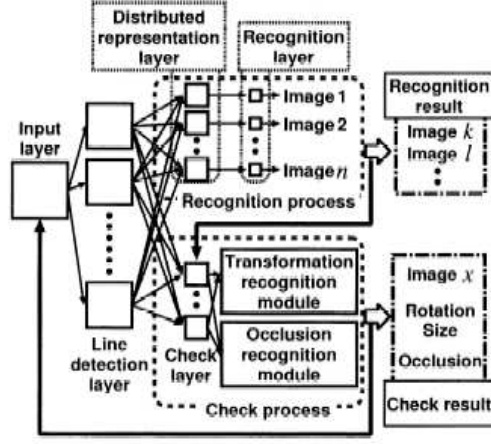


Figure 16: Structure of the proposed network.

in a row and each cell plane has an array of cells arranged in a 2-D plane. The cells will be denoted by the vector \mathbf{n}_x by using the 2-D position on the plane.

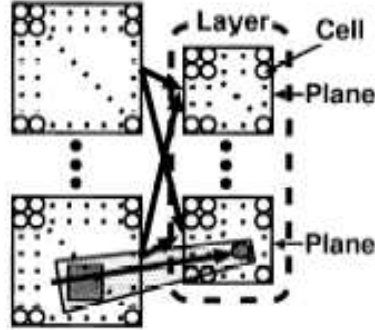


Figure 17: Basic structure of the proposed network.

Figure 18 shows an enlargement of the part shaded in Figure 17. The receptive field, consisting of a range of cells, sends input to each cell in the next plane. The input value is the sum total of input \times weight. A search is performed within a moving range, established for the receptive field, and the movement of the receptive field that corresponds to the maximum input value is selected and input to cell \mathbf{n}_x .

The center \mathbf{n}_{x-1} of the moving range of the receptive field in cell \mathbf{n}_x of the x layer is determined as

$$\mathbf{n}_{x-1} = \left[\frac{\text{side of cell plane of } x-1 \text{ layer}}{\text{side of cell plane of } x \text{ layer}} \right] \quad (6)$$

If \mathbf{k}_x denotes the amount of movement from \mathbf{n}_{x-1} , then the input value $I_x(\mathbf{n}_x, \mathbf{k}_x)$ from the receptive field is obtained using the equation

$$I_x(\mathbf{n}_x, \mathbf{k}_x) = \sum_{\mathbf{v}_x \in \mathbf{V}_x} \omega(\mathbf{n}_x, \mathbf{v}_x) \times O_{x-1}(\mathbf{n}_{x-1} + \mathbf{k}_x + \mathbf{v}_x) \quad (7)$$

where \mathbf{V}_x is the receptive field and $\omega(\mathbf{n}_x, \mathbf{v}_x)$ is the association weight related to \mathbf{n}_x between the $x-1$ layer and the x layer.

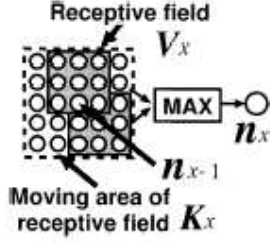


Figure 18: Action of a cell.

The amount of movement \mathbf{m}_x of the receptive field that corresponds to the maximum input value is obtained by using the equation

$$I_x(\mathbf{n}_x, \mathbf{m}_x) = \max_{\kappa_x \in \mathbf{K}_x} (I_x(\mathbf{n}_x, \kappa_x)) \quad (8)$$

where \mathbf{K}_x is the moving range of the receptive field.

The output $O_x(\mathbf{n}_x)$ of a cell is determined as

$$O_x(\mathbf{n}_x) = \varphi [I_x(\mathbf{n}_x, \mathbf{m}_x), \theta] \quad (9)$$

where $\varphi[]$ is the threshold function represented as follows:

$$\varphi[x, \theta] = \begin{cases} 1, & x > \theta \\ 0, & \text{otherwise} \end{cases} \quad (10)$$

where θ is the threshold value. The action when the output of a cell is $O_x(\mathbf{n}_x) = 1$ is referred to as cell firing.

4.2.2 Weighting Method

The weighting method associated with the distributed representation layer, recognition layer and the check layer are described in this section. The weighting methods for the other layers are described in the following sections corresponding to those layers.

Weights are determined by sequentially presenting learning images. When a new learning image is presented, a cell plane is generated in each layer. These cell planes can remember the new learning image. In other words, the cell planes of the distributed representation layer, recognition layer and check layer are associated with all of the other cell planes in these layers with a one-to-one correspondence. The learning image is processed in a similar manner as during execution once the generation of cell planes is finished for all layers. Association weights are determined before the newly generated cell planes are processed.

Association weights are determined for cells within the receptive field in three stages: during initialization, gradation and suppression. However, weights are determined for the recognition layer only during initialization. The reason for this will be explained shortly.

4.2.3 Initialization

Association weights are initialized as follows according to the firing conditions within the receptive field during weighting.

$$\omega_i(\mathbf{n}_x, \mathbf{v}_x) = \begin{cases} 0, & o = 0 \\ \frac{1}{\sum_{\mathbf{v} \in \mathbf{V}_x} O_{x-1}(\mathbf{n}_{x-1} + \mathbf{v})}, & o = 1 \\ o = O_{x-1}(\mathbf{n}_{x-1} + \mathbf{v}_x) \end{cases} \quad (11)$$

As a result, if the same input is entered that was used during weighting the input value to cell \mathbf{n}_x is 1. If a firing cell does not fire during weighting then the input value decreases linearly. Also, a cell that does not fire during weighting does not affect the input value.

4.2.4 Gradation

Performing gradation during weighting enables a positional shift of the firing pattern. This process multiplies association weights that were obtained during initialization by a function that monotonically decreases with distance and adds these products together. This increases the association weights of nonfiring cells around a cell that is firing during weighting. This means that input can be received even if the firing pattern during weighting was shifted.

Gradation is given by

$$\omega_s(\mathbf{n}_x, \mathbf{v}_x) = \sum_{\mathbf{v} \in \mathbf{V}_x} G(\mathbf{v}_x - \mathbf{v}, \sigma_s) \times \omega_i(\mathbf{n}_x, \mathbf{v}) \quad (12)$$

where $G()$ is a Gaussian function represented by

$$G(\mathbf{x}, \sigma) = a \times \exp\left(-\frac{|\mathbf{x}|^2}{\sigma^2}\right) \quad (13)$$

where σ_s is the gradation spreading coefficient, which is determined according to the size of \mathbf{V}_x of the receptive field. Larger values of σ_s correspond to more lenient positional shifting constraints, and small values correspond to stricter constraints.

4.2.5 Suppression

Suppression reduces the association weights of nonfiring portions during weighting and causes an association weight that is not affected much by gradation to become negative. If an input passes this association weight during execution, it becomes a negative input and will be a penalty for an invalid input.

Suppression is given by

$$\omega(\mathbf{n}_x, \mathbf{v}_x) = \frac{H(\omega_s(\mathbf{n}_x, \mathbf{v}_x), \alpha_p)}{\sum_{\mathbf{v} \in \mathbf{V}_x} H(\omega_s(\mathbf{n}_x, \mathbf{v}), \alpha_p) \times O_{x-1}(\mathbf{n}_{x-1} + \mathbf{v})} \quad (14)$$

$$H(x, \alpha) = x \times (1 + \alpha) - \alpha$$

where α_p is the suppression modulation coefficient.

4.2.6 Input Layer

The input layer transforms a gray scale image to a fine line image. Concentration transformation, edge detection and fine line creation processing are executed for the input image to generate the fine line image.

During concentration transformation, the dynamic range is magnified so that the concentration values of the input image are 0 to 255. The following linear transformation is used for the concentration transformation:

$$C(\mathbf{n}_0) = 255 \times \frac{I_0(\mathbf{n}_0) - I_{0_{min}}}{I_{0_{max}} - I_{0_{min}}} \quad (15)$$

$$I_{0_{min}} = \min_{\mathbf{n}_0 \in \mathbf{N}_0} I_0(\mathbf{n}_0) \quad (16)$$

$$I_{0_{max}} = \max_{\mathbf{n}_0 \in \mathbf{N}_0} I_0(\mathbf{n}_0) \quad (17)$$

where \mathbf{N}_0 is the entire area of the input image.

A Sobel filter is used for edge detection. The following interpolation is performed here by using an edge image according to a small threshold value θ_s for a portion in the edge image according to

a large threshold value θ_l where there are few edges in the neighboring pixels, which are referred to as the association window:

$$Q(\mathbf{n}_0) = \begin{cases} S(\mathbf{n}_0, \theta_l), & R(\mathbf{n}_0, \theta_l) > \theta_0 \\ S(\mathbf{n}_0, \theta_s), & \text{otherwise} \end{cases} \quad (18)$$

$$R(\mathbf{n}_0, \theta_l) = \frac{1}{K} \times \sum_{\boldsymbol{\kappa}_0 \in \mathbf{K}_0} S(\mathbf{n}_0 + \boldsymbol{\kappa}_0, \theta_l) \quad (19)$$

where $S()$ is the Sobel function, θ_0 is the association threshold value, \mathbf{K}_0 is the association window and K is the number of cells contained in the association window.

Fine line creation transforms an edge that has a width to a center line of width 1. Fine line creation is performed according to a method that uses a mask pattern [8].

4.2.7 Line Detection Layer

The fine line image in the line detection layer finds linear components in various directions. Each cell plane detects a linear component in one direction. Cells within the same cell plane have the same weight, and the features of this weight determine the direction of the linear component that can be detected. The receptive field moving ranges of cells in the line detection layer are set so that there is no overlap between the cells. Since the weights of neighboring cells are the same, if the receptive field moving ranges overlapped, both of the neighboring cells might end up detecting the same line.

The weight of each cell is determined according to the equation

$$\omega(\mathbf{n}_1, \mathbf{v}_1) = P(\mathbf{n}_1, \mathbf{v}_1) + a - \frac{1}{V_1} \sum_{\mathbf{v} \in V_1} P(\mathbf{n}_1, \mathbf{v}) \quad (20)$$

where $P()$ is the λ_0 direction side suppression function:

$$P(\mathbf{n}_1, \mathbf{v}) = \frac{1}{2\pi} \left(\frac{\exp \frac{-K(\lambda, \mathbf{v})^2}{\sigma_a^2}}{\sigma_a^2} - \frac{\exp \frac{-K(\lambda, \mathbf{v})^2}{\sigma_b^2}}{\sigma_b^2} \right) \quad (21)$$

where $K()$ is the distance between the λ_0 direction line and \mathbf{v}_1 :

$$K(\lambda, \mathbf{v}) = \sqrt{\frac{1}{2} \{ (c_{1_x} - v_{1_x}) \sin \lambda + (c_{1_y} - v_{1_y}) \cos \lambda \}^2} \quad (22)$$

$$\mathbf{v}_1 = (v_{1_x}, v_{1_y}) \quad (23)$$

where a is the weight modulation coefficient, (c_{1_x}, c_{1_y}) are the center coordinates of the receptive field and λ is the angle of the line direction to be learned.

4.2.8 Distributed Representation Layer

The distributed representation layer detects the local features from the linear components in various directions. Each cell fires when the same pattern is contained in the receptive field as was contained during weighting. Therefore, when a learning image is entered, for example, all cells fire in the corresponding cell plane.

In the distributed representation layer, cells within the cell plane compete during weighting, and cells that lose the competition are deleted. This competition causes the cells to be deleted in portions where cells with small features within the receptive field and cells with large features are clustered together. Since only cells with a high degree of importance are retained in this way, the amount of calculations can be efficiently reduced. The multi-winner competitive process [2] was used for this competition. However, while the input to a cell in [2] was from all cells of the previous layer, in the proposed method the input is assumed to be only from cells in the receptive layer.

4.2.9 Recognition Layer

The recognition layer recognizes an input image based on the firing percentage of the distributed representation layer. A cell plane in the recognition layer consists of a single cell. Since the cell planes in the recognition layer are in a one-to-one correspondence with the cell plane in the distributed representation layer, cells in the recognition layer are in a one-to-one correspondence with cell planes in the distributed representation layer.

Recognition layer weights are determined only during weight initialization. When all cells of a cell plane in the distributed representation layer that receives input are firing, the input value is 1.0, and the input value decreases linearly as the number of firing cells is reduced. The input value to each cell is the firing percentage of the cell plane in the distributed layer.

In the recognition layer, cells for which the input value is among the highest p values fire. Learning images that correspond to cells that fired become recognition candidates.

4.2.10 Check Layer

The check layer determines one recognition result from among the recognition candidates that were output by the recognition layer. Check layer plane cells and learning images are in a one-to-one correspondence, and only cell planes that correspond to recognition candidates are subject to checking. The learning image that corresponds to the cell plane having the highest firing rate becomes the recognition result.

Various cells in the check layer fire when the same pattern is contained in the receptive field as was contained during learning. If a learning image is entered, all cells fire in the corresponding cell plane. The amount of movement \mathbf{m}_x of the receptive field that corresponds to the maximum input value shown in eq.(8) is output to the transformation recognition module.

4.2.11 Transformation Recognition Module

The transformation recognition module recognizes translation, rotation, and scaling from the amount of movement \mathbf{m}_x of the receptive field in the check layer that corresponds to the maximum input value. Figure 19 shows the structure of the transformation recognition module. The transformation

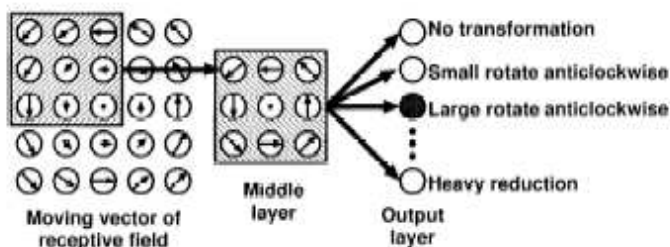


Figure 19: Structure of the transformation recognition module.

recognition module consists of a middle layer and an output layer. The middle layer detects the average movement direction within the receptive field. Each cell of the output layer, which is in a one-to-one correspondence with a transformation, outputs the degree of similarity of the learning pattern with the output of the middle layer. Tables 1 and 2 show learning patterns for counterclockwise rotation and reduction. These are examples for a 3×3 middle layer. The x within the tables is determined by the size of the moving range of the receptive field and size of the rotation or reduction to be detected. The learning pattern for a clockwise rotation is obtained by reversing the sign of the x in the learning pattern for a counterclockwise rotation. Similarly, the learning pattern for magnification is obtained by reversing the sign of the x in the learning pattern for reduction.

The cell having the largest output from this output layer is selected, and the transformation that corresponds to that cell becomes the output of the transformation recognition module.

y \ x	-1	0	1
-1	$(-\frac{x}{\sqrt{2}}, \frac{x}{\sqrt{2}})$	$(-x, 0)$	$(-\frac{x}{\sqrt{2}}, -\frac{x}{\sqrt{2}})$
0	$(0, x)$	$(0, 0)$	$(0, -x)$
1	$(\frac{x}{\sqrt{2}}, \frac{x}{\sqrt{2}})$	$(x, 0)$	$(\frac{x}{\sqrt{2}}, -\frac{x}{\sqrt{2}})$

Table 1: Learning pattern for counterclockwise rotation.

y \ x	-1	0	1
-1	$(\frac{x}{\sqrt{2}}, \frac{x}{\sqrt{2}})$	$(0, x)$	$(-\frac{x}{\sqrt{2}}, \frac{x}{\sqrt{2}})$
0	$(x, 0)$	$(0, 0)$	$(-x, 0)$
1	$(\frac{x}{\sqrt{2}}, -\frac{x}{\sqrt{2}})$	$(0, -x)$	$(-\frac{x}{\sqrt{2}}, -\frac{x}{\sqrt{2}})$

Table 2: Learning pattern for reduction.

The average movement \mathbf{t} is obtained as

$$\mathbf{t} = \frac{1}{N_i} \sum_{\mathbf{n}_i \in \mathbf{N}_i} \mathbf{O}_i(\mathbf{n}_i) \quad (24)$$

The output layer of the middle layer cell \mathbf{N}_m is determined by the equation

$$\mathbf{O}_m(\mathbf{n}_m) = \frac{1}{V_m} \sum_{\mathbf{v}_m \in \mathbf{V}_m} \{\mathbf{O}_i(\mathbf{n}_i + \mathbf{v}_m) - \mathbf{t}\} \quad (25)$$

where \mathbf{V}_m is the middle layer receptive field, V_m is the number of cells contained in \mathbf{V}_m , and \mathbf{n}_i is the input layer cell.

The output of output layer cell \mathbf{n}_o is determined by the equation

$$O_o(\mathbf{n}_o) = \sum_{\mathbf{v}_o \in \mathbf{V}_o} G\{\mathbf{O}_m(\mathbf{n}_m + \mathbf{v}_o) - \mathbf{p}(\mathbf{n}_m + \mathbf{v}_o, \mathbf{n}_o), \sigma_m\} \quad (26)$$

where \mathbf{V}_o is the output layer receptive field, $\mathbf{p}(\mathbf{n}_m, \mathbf{n}_o)$ is the learning pattern cell \mathbf{n}_m of cell \mathbf{n}_o , σ_m is the distribution coefficient and $G()$ is the Gaussian function in EQ.

4.2.12 Occlusion Recognition Module

The occlusion recognition module detects occluded portions from the firing conditions of the check layer. Since cells in the check layer are not firing in the occluded portions, portions where nonfiring cells are clustered together are assumed to be occluded portions.

Firing cell in the check layer are set to -1 , nonfiring cells are set to 1 , these values are gradated by a function that monotonically decreases according to distance, and threshold value processing is performed to detect occluded portions.

The output of cell \mathbf{n} in the check layer is given by

$$\mathbf{O}(\mathbf{n}) = \begin{cases} 1, & F(\mathbf{n}) > \theta \\ 0, & \text{otherwise} \end{cases} \quad (27)$$

$$F(\mathbf{n}) = \sum_{\mathbf{v} \in \mathbf{V}} I(\mathbf{n} + \mathbf{v}) \times E\{I(\mathbf{n} + \mathbf{v}), \mathbf{n} + \mathbf{v}\} \quad (28)$$

$$E\{I(\mathbf{n} + \mathbf{v}), \mathbf{n} + \mathbf{v}\} = \begin{cases} G(\mathbf{n} + \mathbf{v}, \sigma_a), & I(\mathbf{n} + \mathbf{v}) = 1 \\ G(\mathbf{n} + \mathbf{v}, \sigma_b), & I(\mathbf{n} + \mathbf{v}) = -1 \end{cases} \quad (29)$$

where \mathbf{V} is the receptive field, σ_a is the gradation spreading coefficient of a nonfiring cell, and σ_b is the gradation spreading coefficient of a firing cell.

4.2.13 Iterative Processing

An iterative process is used to transform the input image until it approaches a learning image. This is done by an affine transformation in the opposite direction of the transformation recognition result. This also reduces the effect that a feature of an occlusion object has on recognition by removing the occluded portion of the image.

The iterative process is shown in Figure 20. When the firing rate in the check layer is low during

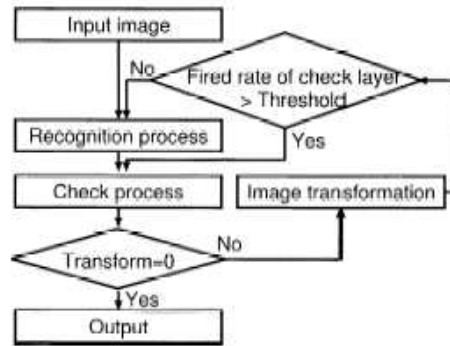


Figure 20: Iterative processing flow.

the initial stage of iteration, image recognition is considered to fail, and both the recognition process and check process are performed. When the firing rate in the check layer exceeds the threshold value, only the check process is performed. When the affine transformation on the input is the same as the one performed in the previous iteration, iteration processing is halted, and the output at that time becomes the final output. If the firing rate is low anywhere other than the occluded portion in the check layer regardless of whether iteration proceeded, recognition is considered to fail, and the iterative process is halted and the output at that time becomes the final output.

5 Conclusion

All of the methods presented in this paper are efficient and robust methods for identifying objects. Each of these methods can determine what an object is as long as a model of the object is stored in the computer. This works great as long as the computer is working within a known environment. However, for a system to be completely autonomous it must also be able to work within an unknown environment. This means that an enormous number of models must be stored within the computer and the computer must be constantly updated with models of newly created objects. This is a nearly impossible task and even if it were possible the amount of time the computer would take to compare an object to all of the stored models would be astronomical. Until this problem can be solved a system will have to work within a confined workspace, where there are a limited number of known objects.

References

- [1] C.S. Chua and R. Jarvis. Point signatures: A new representation for 3d object recognition. *International Journal of Computer Vision*, 1997.
- [2] J. Huang and M. Hagiwara. A multi-winners self-organizing neural network. *Transactions IEICE*, 1998.
- [3] A. Johnson. *A Representation for 3-D Surface Matching*. PhD thesis, Carnegie Mellon University, 1997.
- [4] A. Johnson and M. Hebert. Object recognition by matching oriented points. *Proc. Computer Vision and Pattern Recognition*, 1997.
- [5] Andrew E. Johnson and Martial Hebert. Efficient multiple model recognition in cluttered 3-d scenes. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1998.
- [6] Y. Kawai et al. Stereo correspondence using segment connectivity. *International Conference on Pattern Recognition*, 1998.
- [7] Jung H. Kim, Sung H. Yoon, and Kwang H. Sohn. A robust boundary-based object recognition in occlusion environment by hybrid hopfield neural networks. *Pattern Recognition*, 1996.
- [8] S. Murakami. Image processing engineering. *Tokyo Denki University Publishing*, 1996.
- [9] J. Opera. *Differential Geometry and its Applications*. Prentice Hall, 1997.
- [10] Noriaki Sato and Hagiwara Masafumi. Parallel-hierarchical neural network for 3d object recognition. *Systems and Computers in Japan*, 2004.
- [11] F. Stein and G. Medioni. Structural indexing: Efficient 3-d object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1992.
- [12] K. Sugimoto and F. Tomita. Boundary segmentation by detection of corner, inflection and transition point. *IEEE Workshop on Visualization and Machine Vision*, 1994.
- [13] Yasushi Sumi et al. 3d object recognition in cluttered environments by segment-based stereo vision. *International Journal of Computer Vision*, 2002.
- [14] J.P. Thirion. Extremal points: Definition and application to 3d image registration. *IEEE Conference on Computer Vision and Pattern Recognition*, 1994.
- [15] F. Tomita and H. Takahashi. Algorithms for a b-rep of an image as its intermediate description. *Information and Communication Engineers*, 1986.
- [16] F. Tomita and S. Tsuji. *Computer Analysis of Visual Textures*. Kluwer Academic Publishers, Norwell, MA.
- [17] T. Ueshiba. An efficient matching algorithm for segment-based stereo vision using dynamic programming technique. *IAPR Workshop on Machine Vision Applications*, 1998.
- [18] Shankar Vaidyanathan and Shivakumar Raman. Omne-vision: Object measurement in a noisy environment using vision. *Computers in Industry*, 1995.
- [19] Sameh M. Yamany et al. Object recognition using neural networks and surface signatures. *IEEE Proceedings of the International Joint Conference on Neural Networks*, 1999.
- [20] J.M. Zurada. *Introduction to Artificial Neural Systems*. West, 1992.