Monograph:
Banks, S.P. and Harrison, R.F. (1990) Simple Object Recognition by Neural Networks Application of the Hough Transform. Research Report. Acse Report 410 . Dept of Automatic Control and System Engineering. University of Sheffield

Simple Object Recognition by Neural Networks: Application of the Hough Transform

by

S. P. Banks and R.F.Harrison

Department of Control Engineering

University of Sheffield

Mappin Street

SHEFFIELD S1 3JD

Research Report No 410

October 1990

Abstract

A 'neural-like'parallel system is described for the real-time implementation of the Hough transform. It is applied to simple object recognition in computer vision.

1 Introduction

The Hough transform is a well-known technique for the recognition of simple objects in two-dimensional images ([4],[1]). However, it is very difficult to compute on a standard computer (in a reasonable time), and a variety of improvements have been proposed to cope with this problem ([2],[3]). In this paper we shall describe a neural- like parallel system which will implement the Hough transform very simply and quickly. First, a 'naive'network is proposed which requires many output neurons and then a multi-layer system which works by 'homing in'on the correct parameters is considered. The system is shown to work well with 'elementary'objects (circles,squares,etc.) and also with simple compound objects made up of combinations of elementary objects.

The weights in the network can be chosen *a priori* without the need for lengthy training cycles, although the system can be trained in this way if so desired. Also the final network will contain non-neural components, hence the expression 'neural-like'system. Here we are merely interested in the ideas of neural networks to show that the Hough transform can be computed easily in parallel using these types of systems.

2 The Hough Transform

Simple geometrical shapes can often be represented by simple algebraic equations involving a number of parameters. For example, a circle in the (x_1, x_2)-

plane is given by an equation of the form

$$(x_1 - \alpha_1)^2 + (x_2 - \alpha_2)^2 - \alpha_2^2 = 0$$

for some parameter vector $(\alpha_1, \alpha_2, \alpha_3) \in R^3$. Thus, for each fixed point $(\overline{x}_1, \overline{x}_2)$ in image space R^2, there are an infinity of possible circles passing through $(\overline{x}_1, \overline{x}_2)$ given by

$$(\overline{x}_1 - \alpha_1)^2 + (\overline{x}_2 - \alpha_2)^2 - \alpha_2^2 = 0.$$

This single equation defines a hypersurface in R^3 (parameter space). Let $h(\overline{x}_1, \overline{x}_2)$ denote this hypersurface and let $\mathcal{P}(R^3)$ denote the set of all subsets of R^3. Then the *Hough transform* is the map

$$H : R^2 \longrightarrow \mathcal{P}(R^3)$$

defined by

$$H(x_1, x_2) = h(x_1, x_2).$$

The Hough transform is usually applied numerically in the following way. First an image is processed to find the edges and then a point in the processed image is thresholded to 1 or 0 depending on whether or not a pixel lies on an edge. Now suppose we are looking for circles in the image. Then parameter space R^3 is divided into cubes and each pixel $(\overline{x}_1, \overline{x}_2)$ with value 1 is mapped into the hypersurface $H(\overline{x}_1, \overline{x}_2)$ so that a 1 is placed in each cube in parameter space through which $H(\overline{x}_1, \overline{x}_2)$ passes. This is repeated for each 'black'edge pixel in

the image, adding 1 to the value in any parameter cube through which the corresponding hypersurface passes. If the image contains a single circle with parameters $(\overline{\alpha}_1, \overline{\alpha}_2, \overline{\alpha}_3)$ then the value in the parameter cube containing the point $(\overline{\alpha}_1, \overline{\alpha}_2, \overline{\alpha}_3)$ will be large and all the values in other cubes will be small. (Of course, because of inaccuracies in the image, values in cubes 'near'$(\overline{\alpha}_1, \overline{\alpha}_2, \overline{\alpha}_3)$ will be greater than 1 and peak at $(\overline{\alpha}_1, \overline{\alpha}_2, \overline{\alpha}_3)$.) The location and size of the circle can then be found by thresholding the values in parameter space cubes.

In the next section we shall use these basic principles to design a 'naive'neural network to implement the Hough transform. This will require a large number of neurons, however, and so we shall modify the system by 'homing in'on the appropriate parameter values by using a set of similar networks.

3 Recognition of Circles in Images

In this section we shall examine the problem of recognizing circles in a given monochrome image. For simplicity we shall assume only binary values for the image pixels- i.e. black or white. We wish to design a neural network which will implement the Hough transform. The equation of a generic circle is given by

$$(x-\alpha)^2 + (y-\beta)^2 = \gamma^2. \tag{3.1}$$

The three parameters (α, β, γ) form the parameter space R^3 and the Hough transform is the map $H : R^2 \longrightarrow \mathcal{P}(R^3)$ defined by

$$H((x,y)) = \{(\alpha,\beta,\gamma) \in R^3 : (x-\alpha)^2 + (y-\beta)^2 = \gamma^2\}$$

i.e. $H((x,y))$ is the set of points in parameter space for which (x,y) lies on a circle with these parameters.

We shall design first a naive neural network to implement this transformation and consider ways to overcome its shortcomings later. Thus we shall discretize the image space into image pixels and the parameter space into 'parameter pixels'as in fig.1. Let the image pixels be numbered $p_1, \cdots, p_K$ and the parameter pixels $q_1, \cdots, q_L$. A simple neural network to implement the Hough transform is shown in fig.2. It is seen that we require a two-layer network with a neuron for each image pixel in layer 1 and a neuron for each parameter pixel in layer 2. The weights in layer 1 are all unity and the neurons are designed to give a unit output if the input pixel is black and zero output otherwise.

To describe the operation of the second layer neurons, recall that each neuron in this layer corresponds to a fixed set of circle parameters- say $(\alpha_i, \beta_i, \gamma_i)$ for neuron i. The weight w_{ij} on the input line from pixel j to neuron i is equal to 1 if the pixel is on a circle with parameters $(\alpha_i, \beta_i, \gamma_i)$ and zero otherwise. Thus the input to neuron i in layer 2 is given by

$$I_i = \sum_{j=1}^{K} w_{ij} p_j$$

where

$$p_j = \begin{cases} 1 & \text{if pixel } j \text{ is black} \\ 0 & \text{if pixel } j \text{ is white} \end{cases}$$

and w_{ij} is 1 if

$$(x_j - \alpha_i)^2 + (y_j - \beta_i)^2 = \gamma_i^2$$

where (x_j, y_j) is the position of pixel j in the image. The activation of layer 2 neurons is trivial and the output O_i is simply the thresholded input with some threshold, say

$$O_i = f_i(I_i)$$

where

$$f_i(\delta) = \begin{cases} 1 & \text{if } \delta \geq N_i \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

The number N_i is chosen to be proportional to γ_i (so that the larger the circle, the more pixels must be black to identify a circle).

As an example, consider the image in fig.3, which has a single circle.If the pixel edges are taken to be of unit length then the circle has the equation

$$x^2 + y^2 = 25$$

The parameters are therefore $(0, 0, 5)$. Suppose that neuron 1 in the output layer corresponds to these parameter values. Then if we choose $N_1 = 20$ the output of this neuron will be 1 and all others will be zero (provided we do not have any neurons corresponding to circles of radius of the order of a parameter pixel).The value of 20 is chosen to be less than the number of pixels on a perfect circle so that the network will recognize imperfect or incomplete circles.

Note that we have set the weights in this system. We could also train the system by presenting a large number of samples and using some training algorithm.

The main drawback with the above network is the large number of neurons required in the parameter layer (layer 2);if each side of the parameter cube is divided into n parts, then we require n^3 neurons in layer 2. We shall next investigate a modified network which requires relatively few neurons in this layer. It is based on the idea of finding a rough set of parameters and then using a finer grid 'near'this set. Thus we could, for example, choose just three divisions on each side of the parameter cube as shown in fig.4. Suppose there is a circle in the image with parameter vector as in this figure. Then a network of the form in fig.2 with 27 layer 2 neurons will place the parameter vector in the subcube C_1. We then treat the subcube C_1 in just the same way as C. Thus we subdivide C_1 into 27 subcubes and design another system of the form in fig.2 to place the circle parameters more accurately in C_1. The weights on the layer 2 neurons,however, must be selected in an appropriate way by the outputs of the layer 2 neurons in the 'C'circuit. This process can be repeated to any desired accuracy and clearly uses many fewer neurons than the 'naive'network by only discretizing appropriate subcubes of parameter space. The neural network for the parameter space with just the main cube C and a single subcube C_1 is shown in fig.5. The weight generating function block in fig.5 selects the correct weights for the C_1 neurons in an obvious way.

Remark Note that we could feedback the weights from the weight generating function directly into the output neurons for block C (after a delay) thereby removing the need for an extra set of neurons for block C_1. Thus the system

for 'homing in'could be implemented using just 27 output neurons.

4 Other Shapes

The network developed above can be used to recognize other shapes in simple images. Suppose we wish to find the 'house'shape shown in fig.6. For simplicity we shall assume that the triangle is equilateral. Thus, both the triangle and the square can be described by four parameters:

(α_1, α_2) – coordinates of a fixed corner

α_3 – orientation angle

α_4 – side length

We shall use the 'homing in'method of figs.4 and 5, this time with a four-dimensional parameter space requiring 81 output neurons in each sub-block.

The system for recognizing the house shape in fig.6 will then consist of two subsystems of the form of fig.5 for the triangle and square, respectively, followed by a simple logic circuit to decide whether or not the parameters $(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ are identical for each shape -this will guarantee that the triangle is on top of the square in the correct orientation. Note, however, that there are three other possible configurations as shown in fig.7, where we have introduced the labels 'T'and 'S'for 'triangle'and 'square'respectively. For example, in fig.7(a), we have the relations

$$\alpha_1^S = \alpha_1^T - \alpha_4^T sin(\alpha_3^T - \pi/2)$$

$$\begin{aligned} \alpha_2^S &= \alpha_2^T - \alpha_4^T cos(\alpha_3^T - \pi/2) \\ \alpha_3^S &= \alpha_3^T - \pi/2 \\ \alpha_4^S &= \alpha_4^T \end{aligned} \tag{4.1}$$

and similar relations for (b) and (c). The complete system for recognizing the 'house'shape is then as shown in fig.8. In this figure the configuration logic determines if $\alpha_i^S = \alpha_i^T \ \ (i = 1, \cdots, 4)$ or if (4.1) (or the corresponding relations for (b) and (c) in fig.7) are satisfied. The output is 1 or 0 depending on whether or not a shape has been found.

5 Conclusions

In this paper we have described a simple, parallel, 'neural-like'system for implementing the Hough transform in real time. It can be applied to the problem of simple object recognition in images and could be useful in real-time computer vision. The novelty in the method is the 'homing in'facility in the system which cuts out many unnecessary parameters in the transform.

References

[1] D.H.Ballard, 'Generalizing the Hough transform to detect arbitrary shapes',*Pattern Recognition*, **13** (1981),111-122.

[2] S.R.Deans, 'Hough transform from the Radon transform', *IEEE Trans. Pattern Analysis and Machine Intelligence* , **3** (1981),185-188.

[3] G.Eichmann and B.Z.Dong, 'Coherent optical production of the Hough transform', *Applied Optics* ,**22** (1972),830-834.

[4] P.V.C.Hough, 'A Method and means for recognizing complex patterns',US Patent # 3,039,654,(1962).

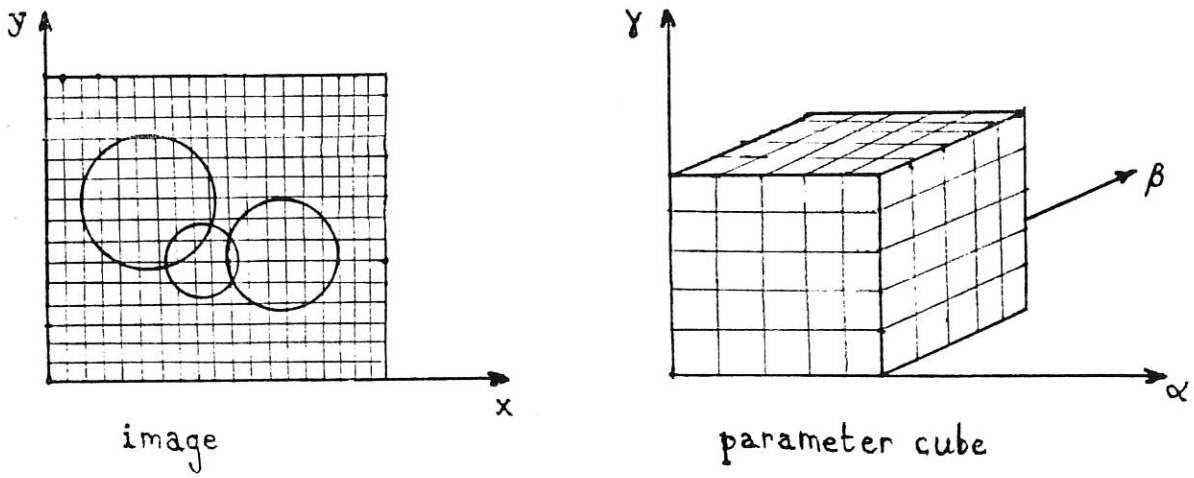

fig.1. Disretizing the problem

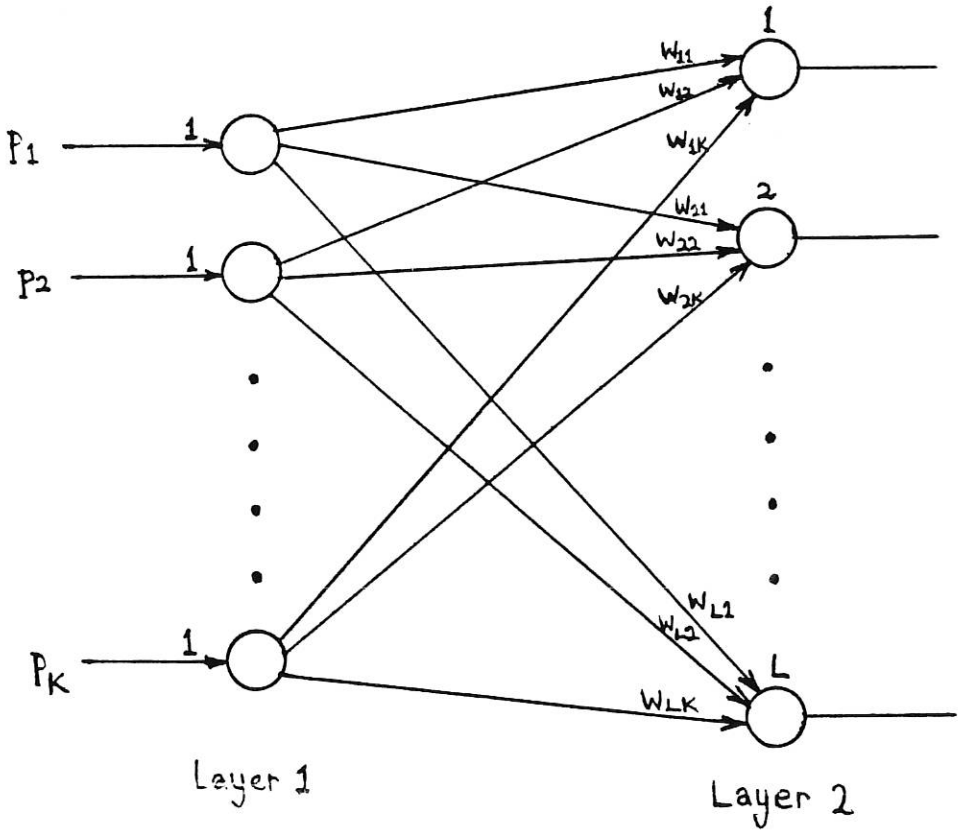

fig.2. A Neural Hough Transformer

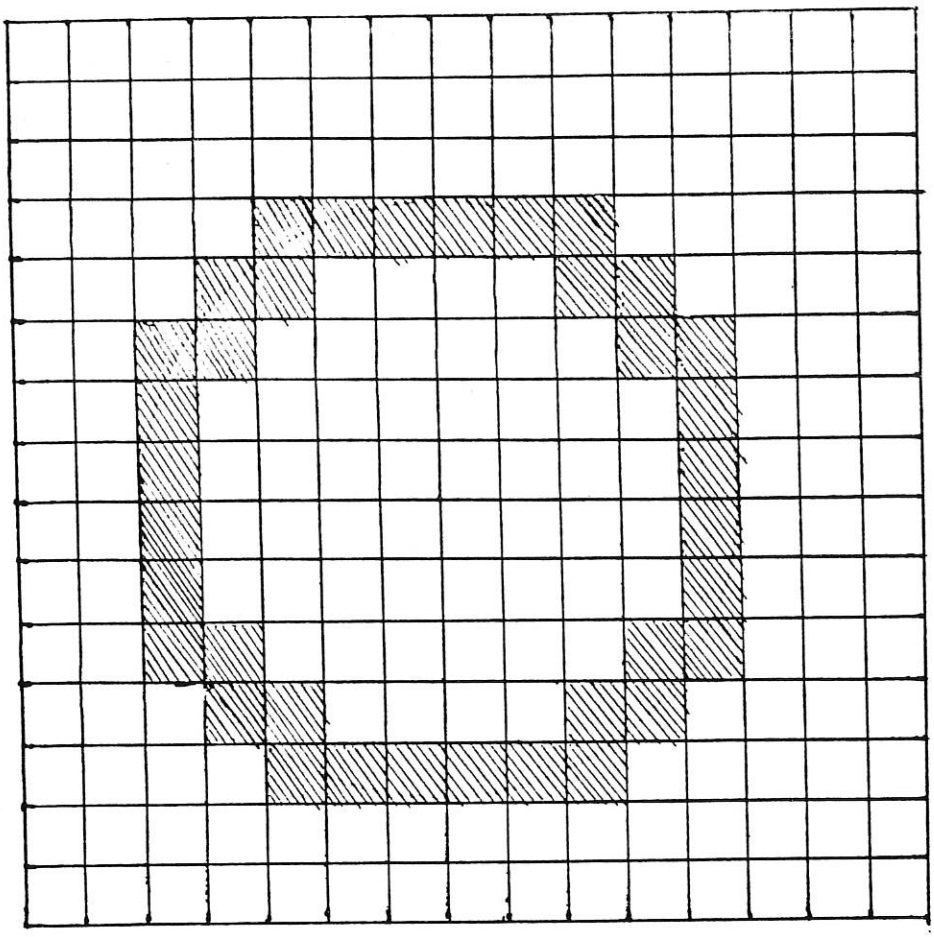

fig.3. An Image with a Single Circle

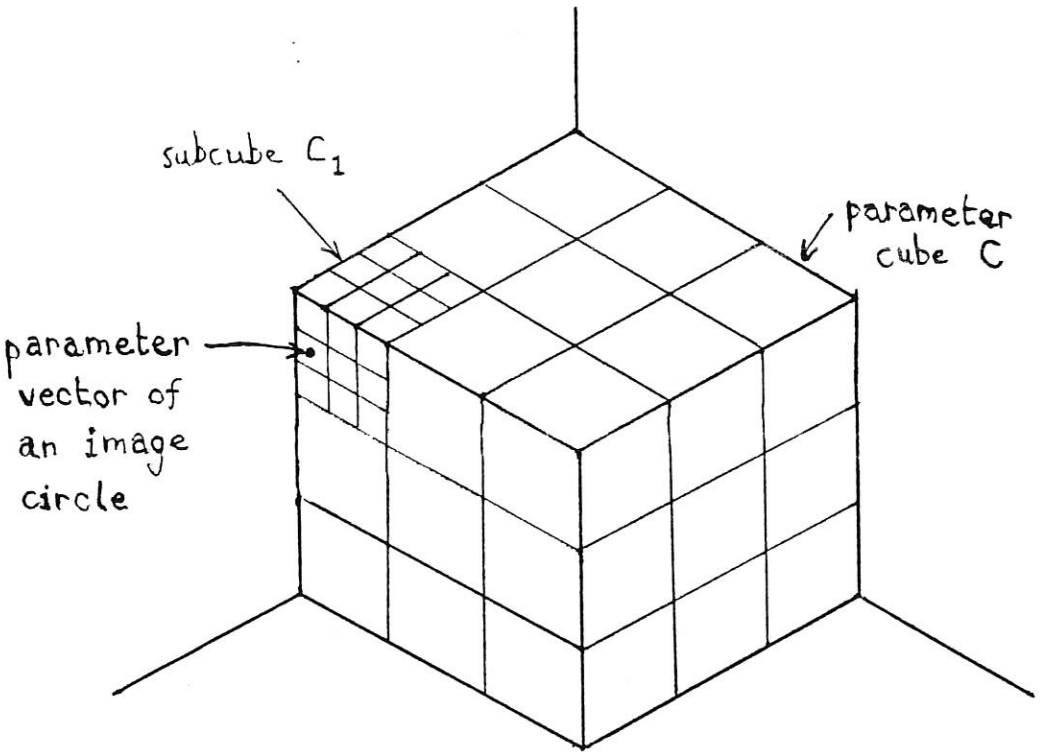

fig.4. 'Homing in' on the Correct Parameters

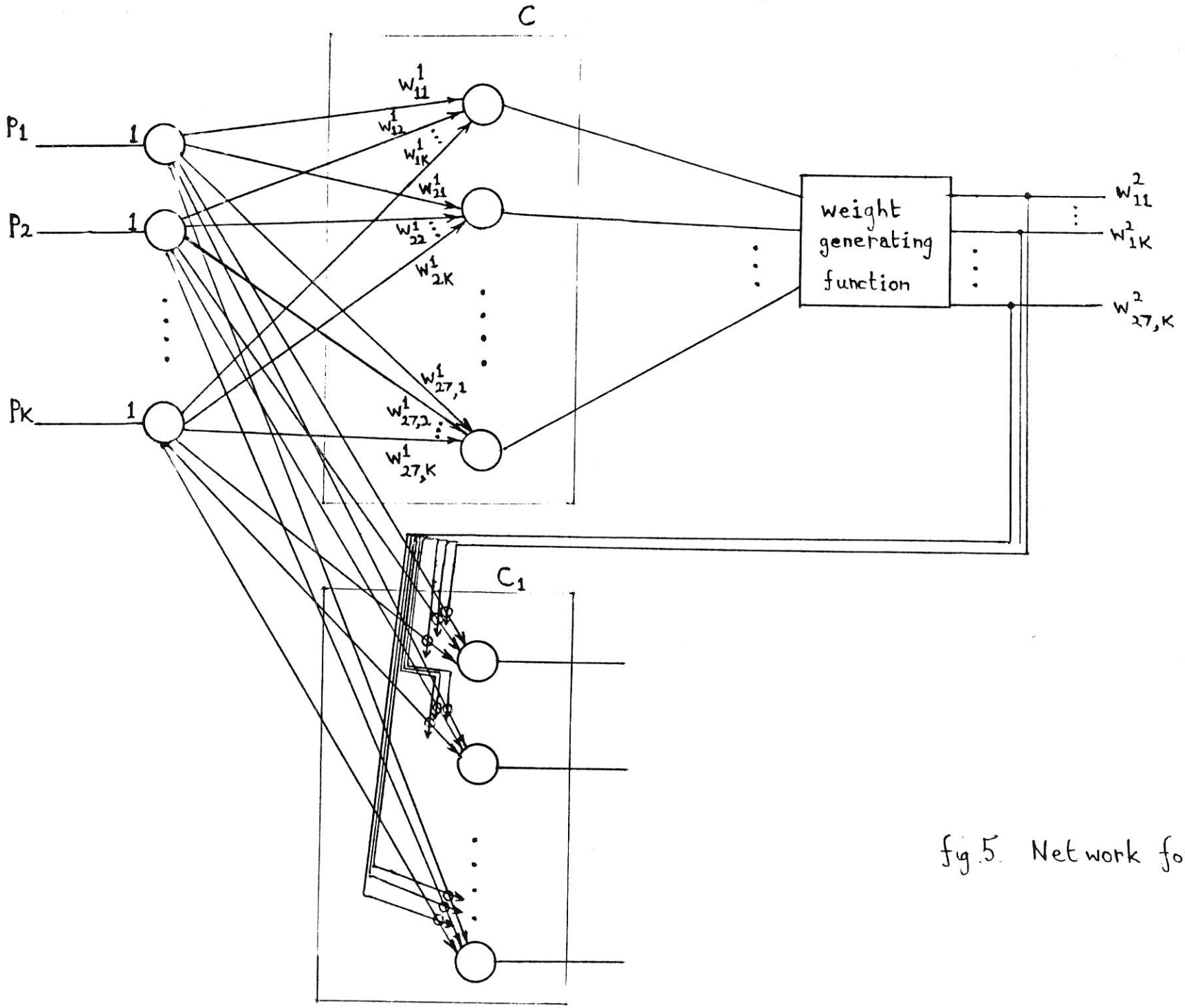

fig 5 Network for 'homing in'

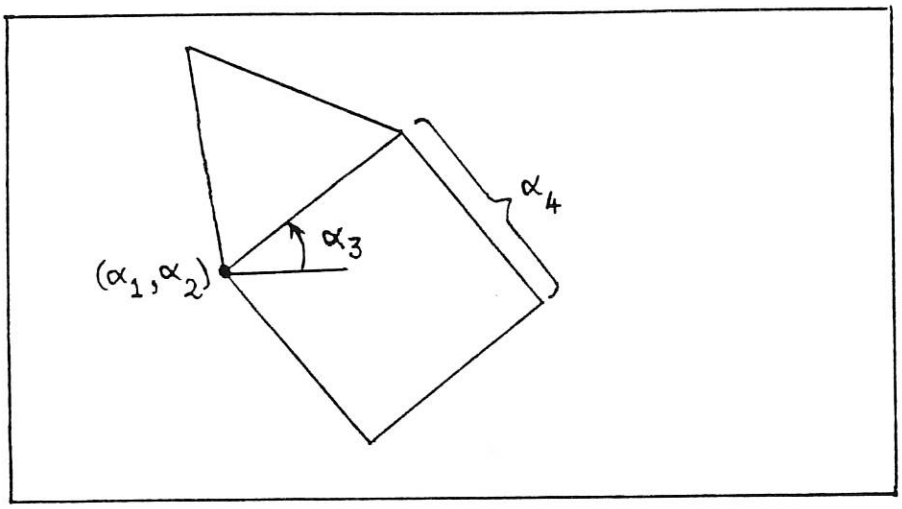

fig. 6. Simple 'House' Shape

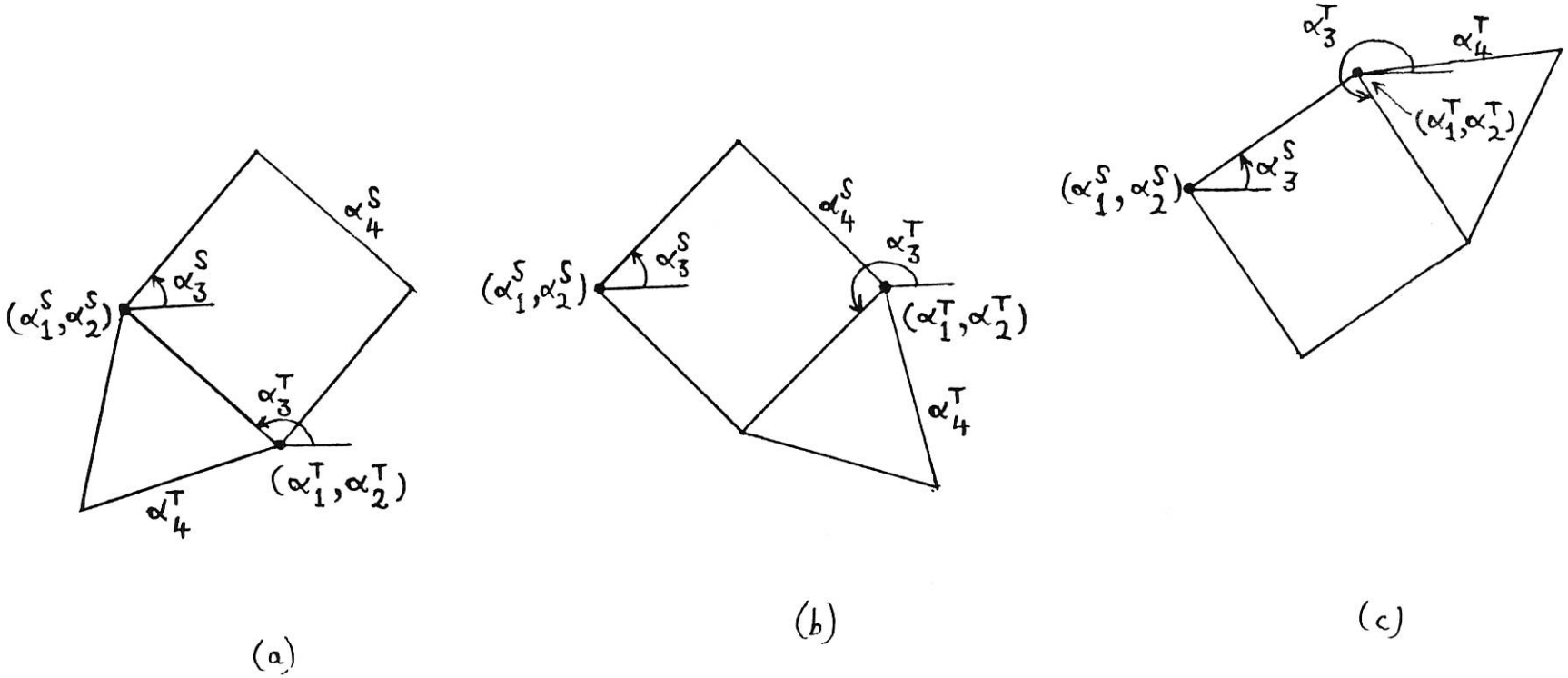

Fig. 7. Other Possible Configurations

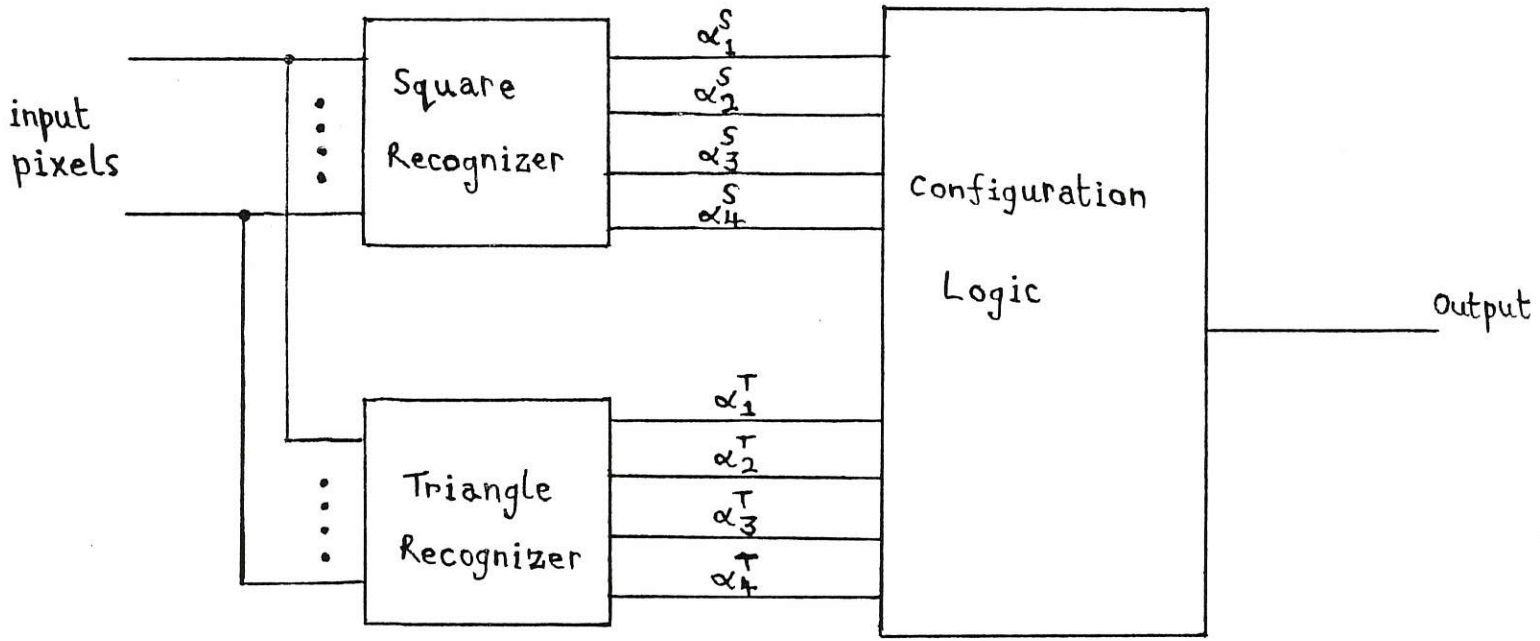

fig. 8. 'House' Shape Recognizer