INTERNATIONAL JOURNAL OF CIRCUIT THEORY AND APPLICATIONS, VOL. 24, 409–424 (1996)

ANALOGIC CNN ALGORITHMS FOR 3D INTERPOLATION–APPROXIMATION AND OBJECT ROTATION USING CONTROLLED SWITCHED TEMPLATES†

L. NEMES*, G. TÓTH*, T. ROSKA AND A. RADVÁNYI

Analogical and Neural Computing Laboratory, Hungarian Academy of Sciences, Kende u. 13, H-1111 Budapest, Hundary

SUMMARY

Analogic CNN algorithms are presented for various interpolation and approximation tasks in 3D. They are designed on the basis of mechanical analogies. Symmetric space-variant operations are implemented by the CNN algorithms; with switched templates, a key example is object rotation. Direction and speed coding are shown in detail.

1. INTRODUCTION

Two types of problems are solved: interpolation and approximation in 3D and object rotation or motion on a prescribed route. The efficiency of CNN algorithms[1–4] has been shown mainly in cases where the CNN templates and/or the tasks are space-invariant. The first part of this paper raises the problem of space variance, the second part offers a solution without major modification of the regular structure. The space variance of the templates is controlled in a locally prescribed or controlled way as a function of a single value at each cell instead of by defining a space-variant template map for the whole network.

2. 3D INTERPOLATION AND APPROXIMATION

For solving a lot of problems it is necessary to fit a surface through given points. When points on the object are sparsely specified, the surface of the object is calculated on a relatively dense grid.

The problem can be solved by interpolation and approximation with digital computers,[5,6] but serial execution leads to low speed. If a cellular neural network is applied, it will work in a fully parallel manner. Next, two different fitting methods will be described: *interpolation* and *approximation*. In the case of interpolation the surface fits exactly through the given points. In the case of approximation it fits only approximately.

2.1. The model

There are innumerable surfaces that fit the given points. One of them must be selected. We have used a physical model whereby it was examined how a *thin plate* behaves under some special constraints.[5]

* Also: Technical University of Budapest, Budapest, Hungary
† Part of this research has been reported in the Proceedings of the 1994 IEEE International Workshop on Cellular Neural Networks and Their Applications held in Rome.

CCC 0098–9886/96/030409–16

Received 10 January 1995
Revised 10 July 1995

In the case of interpolation these constraints mean that the plate is fixed at known points. In the case of approximation the constraints are provided by springs. Springs are placed between the given point and the appropriate point of the surface, hence the surface will pass near the given points. It seems rational to imitate the thin plate by a cellular neural network.

Using a cellular neural network, the potential energy of the plate will be determined. At equilibrium the potential energy of a mechanical system is minimal, so the shape of the thin plate is easily determined. That surface is found which has the minimum potential energy and is consistent with the constraints.

Describing the surface by a function $z = v(x, y)$, Courant and Hilbert[7] give the potential energy as

$$E_{\text{thin plate}} = \iint_{\Omega} [\tfrac{1}{2}(\Delta v)^2 - (1-\sigma)(v_{xx}v_{yy} - v_{xy}v_{xy})]\,dx\,dy - \iint_{\Omega} gv\,dx\,dy - \int_{\partial\Omega} p(s)v\,ds - \int_{\partial\Omega} m(s)v_n\,ds + \tfrac{1}{2}w\sum_i [v(x_i, y_i) - z_i]^2 \tag{1}$$

The functional is composed of five terms. The first term gives the strain energy within the interior of the plate. The combining factor σ is called the *Poisson constant*. It accounts for the change in width as the plate material is stretched lengthwise. The second term is the result of the effect of gravity. The next two terms quantify energies imparted around the edges of the plate. The quantity $p(s)$ is the externally applied force at the boundary, $m(s)$ is the bending moment at the boundary, Ω is the surface of the plate, $\partial\Omega$ is the boundary of the plate, s denotes the arc length along the boundary and v_n denotes the directional derivative of v along the outward normal to the boundary. The last term is the energy of the springs, where w is the stiffness of the springs and x_i, y_i, z_i are the three co-ordinates of the ith given point.

In case of approximation there are no external forces applied and the effect of gravity need not be considered; if σ is zero, the formula can be expressed as

$$E_{\text{approximation}} = \iint (v_{xx}^2 + 2v_{xy}^2 + v_{yy}^2)\,dx\,dy + w\sum_i [v(x_i, y_i) - z_i]^2 \tag{2}$$

The factor $\frac{1}{2}$ is missing because it does not affect the function $v(x, y)$ belonging to the minimal energy. In the case of interpolation, springs are not used and the expression becomes

$$E_{\text{interpolation}} = \iint (v_{xx}^2 + 2v_{xy}^2 + v_{yy}^2)\,dx\,dy \tag{3}$$

2.2. *Discretization*

The surface is given as a function $z = v(x, y)$. Discretization is necessary to handle this surface with a digital computer or a cellular neural network. Grid values $v_{(i,j)}$ will be utilized instead of $z = v(x, y)$, where $v_{(i,k)}$ is the altitude of the surface at grid point (i, j). If an $m \times m$ grid is used, then $m \times m$ $v_{(i,j)}$-values must be specified. They can be placed in a vector $\mathbf{v}$.

To discretize the energy function, the differential terms will be approximated by difference terms:

$$\begin{aligned} v_{xx}^2(i,j) &= \frac{1}{h^2}(v_{(i+1,j)} - 2v_{(i,j)} + v_{(i-1,j)}) + O(h^2) \\ v_{yy}^2(i,j) &= \frac{1}{h^2}(v_{(i,j+1)} - 2v_{(i,j)} + v_{(i,j-1)}) + O(h^2) \\ v_{xy}^2(i,j) &= \frac{1}{h^2}(v_{(i,j)} - v_{(i+1,j)} - v_{(i,j+1)} + v_{(i+1,j+1)}) + O(h^2) \end{aligned} \tag{4}$$

The integrals are also approximated by sums. Afterwards the following discretized energy function[6] is written for the approximation (assuming that h is unity):

$$E(\mathbf{v}) = \sum_{i=1}^{m-2}\sum_{j=0}^{m-1}(v_{(i-1,j)} - 2v_{(i,j)} + v_{(i+1,j)})^2 + \sum_{i=0}^{m-1}\sum_{j=1}^{m-2}(v_{(i,j-1)} - 2v_{(i,j)} + v_{(i,j+1)})^2$$
$$+2\sum_{i=0}^{m-2}\sum_{j=0}^{m-2}(v_{(i,j)} - v_{(i+1,j)} - v_{(i,j+1)} + v_{(i+1,j+1)})^2 + w\sum_{i}(v_{(x_i,y_i)} - z_i)^2 \qquad (5)$$

In the case of interpolation the last term in this expression is omitted.

2.3. *Finding the minimum using a* ***gradient method***

A vector $\mathbf{v}$ where $E(\mathbf{v})$ is minimal must be found. Using a gradient method, the following iteration rule can be used:

$$\mathbf{v}_{k+1} = \mathbf{v}_k - \beta \operatorname{grad} E(\mathbf{v})|_{\mathbf{v}_k}, \qquad \lim_{k\to\infty} \mathbf{v}_k = \mathbf{v}_{\min} \qquad (6)$$

For the cellular neural network approach a modified formula is used. The settled state of the next equation finds the vector $\mathbf{v}$ that belongs to the minimal energy:

$$\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} = -\beta \operatorname{grad} E(\mathbf{v}) \qquad (7)$$

From (5) the energy function can be expressed as the sum of products of two grid elements and a constant:

$$E(\mathbf{v}) = \frac{1}{2}\sum_{(i,j)}\sum_{(k,l)\in N_r(i,j)} Q(i,j,k,l)v_{(i,j)}v_{(k,l)}, \quad r = 2 \qquad (8)$$

A co-ordinate of the gradient is the sum of products of a grid element and a constant:

$$\frac{\partial E}{\partial v_{(i,j)}} = \sum_{(k,l)\in N_r(i,j)} Q(i,j,k,l)v_{(k,l)}, \ r = 2 \qquad (9)$$

Here the factors $Q(i,j,k,1)$ are constants:

$$Q(i,j,k,l) = \frac{\partial^2 E}{\partial v_{(i,j)}\,\partial v_{(k,l)}} \qquad (10)$$

If the formula of the gradient method and the formula of the cell's differential equation are compared, the template values can be determined. The interpolating network has the following features.

1. The current I is zero.
2. The matrix B is zero.
3. The cell non-linearity function is $g(x) = x$.
4. The matrix A, depending on the position of the cell, may be one of the following matrices or their reflected variants (assuming that the resistance R and capacitance C are unity in a cell):

$$\begin{bmatrix} 0 & 0 & -2 & 0 & 0 \\ 0 & -4 & 16 & -4 & 0 \\ -2 & 16 & -39 & 16 & -2 \\ 0 & -4 & 16 & -4 & 0 \\ 0 & 0 & -2 & 0 & 0 \end{bmatrix}, \quad \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -4 & 12 & -4 & 0 \\ -2 & 16 & -37 & 16 & -2 \\ 0 & -4 & 16 & -4 & 0 \\ 0 & 0 & -2 & 0 & 0 \end{bmatrix} \qquad (11a,b)$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -2 & 12 & -21 & 12 & -2 \\ 0 & -4 & 12 & -4 & 0 \\ 0 & 0 & -2 & 0 & 0 \end{bmatrix}, \quad \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 8 & -19 & 12 & -2 \\ 0 & -4 & 12 & -4 & 0 \\ 0 & 0 & -2 & 0 & 0 \end{bmatrix} \tag{11c,d}$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -4 & 12 & -4 & 0 \\ 0 & 12 & -35 & 16 & -2 \\ 0 & -4 & 16 & -4 & 0 \\ 0 & 0 & -2 & 0 & 0 \end{bmatrix}, \quad \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -7 & 8 & -2 \\ 0 & 0 & 8 & -4 & 0 \\ 0 & 0 & -2 & 0 & 0 \end{bmatrix} \tag{11e,f}$$

In Table I the matrix A of cells in positions (i,j) can be seen. The letters a–f denote the matrices in 11(a)–11(f) respectively; x, y and t denote reflection around co-ordinate axes x and y and matrix transposition respectively.

For example at position $(0, m-2)$, dtx can be found. The letter d indicates that the A-template of the cell is the matrix d. The letter t indicates that it must be transposed:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 8 & -4 & 0 \\ 0 & 0 & -19 & 12 & -2 \\ 0 & 0 & 12 & -4 & 0 \\ 0 & 0 & -2 & 0 & 0 \end{bmatrix} \tag{12}$$

At the end the letter x denotes reflection around the x-axis. The result is

$$\begin{bmatrix} 0 & 0 & -2 & 0 & 0 \\ 0 & 0 & 12 & -4 & 0 \\ 0 & 0 & -19 & 12 & -2 \\ 0 & 0 & 8 & -4 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{13}$$

The interpolating network is used in the following way. Some points of the 3D shape are given with three co-ordinates i, j, z. Co-ordinates i and j are the discretized co-ordinates x and y. The co-ordinate z is real. If the co-ordinates of a given point are i, j, z, then it means that the altitude of the surface at a grid point (i,j) is known. This altitude is z.

Before starting the network transient, the states of cells must be filled in. The matrix B is zero, so the input of the network is not used. If at a position (i,j) the altitude z of the surface is known, this altitude is filled into the state and it is kept fixed after starting the transient. If at a position (i,j) the altitude of the surface is not known, then a zero is filled into the state and a change in state is allowed.

Table I. Space-variant feedback templates (A) of cells depending on position

j	$i=0$	1	2	3	4	.	$m-3$	$m-2$	$m-1$
0	*f*	*d*	*c*	*c*	*c*	.	*c*	*dy*	*fy*
1	*dt*	*e*	*b*	*b*	*b*	.	*b*	*ey*	*dty*
2	*ct*	*bt*	*a*	*a*	*a*	.	*a*	*bty*	*cty*
3	*ct*	*bt*	*a*	*a*	*a*	.	*a*	*bty*	*cty*
4	*ct*	*bt*	*a*	*a*	*a*	.	*a*	*bty*	*cty*
.	.	.	.	.	.	.	.	.	.
$m-3$	*ct*	*bt*	*a*	*a*	*a*	.	*a*	*bty*	*cty*
$m-2$	*dtx*	*ex*	*bx*	*bx*	*bx*	.	*bx*	*exy*	*dtxy*
$m-1$	*fx*	*dx*	*cx*	*cx*	*cx*	.	*cx*	*dxy*	*fxy*

Next the network will be started. In the settled state the output of the network gives the fitted surface.

Suppose that a cellular neural network of 5 x 5 cells is used. Three points of the 25 are specified with altitudes 3, 1 and 3. Before the start the following states are filled in:

j	$i=0$	1	2	3	4
0	0	0	3	0	0
1	0	0	0	0	0
2	1	0	0	0	0
3	0	0	0	0	0
4	0	0	3	0	0

(14)

At the specified points the states are kept fixed. In the settled state the output is

j	$i=0$	1	2	3	4
0	1	2	3	4	5
1	1	2	3	4	5
2	1	2	3	4	5
3	1	2	3	4	5
4	1	2	3	4	5

(15)

The network fits a flat surface through the three given points. (The equation of the flat surface is $z=j+1$.)

2.4. *The approximating network*

The interpolating network fits the surface exactly through the given points. If there is only one wrong point among the given points, it will deform the whole shape. This problem can be eliminated by approximation. In this case the surface fits only approximately through the given points. The surface is expected to be smooth and fit well through the given points. A weight factor w determines the relative importance of the two requirements. If w is large, then the surface will fit almost exactly through the given points, so the network gives a result similar to interpolation. If w is small, then smoothness is more important than fitting, so the surface will not fit exactly through the given points but will be smooth.

The templates can now be determined. The network is close to the interpolating network. The differences are the following.

1. At cells (i,j) that belong to specified points, the central element of the matrix A is smaller than before and the current I is not zero:

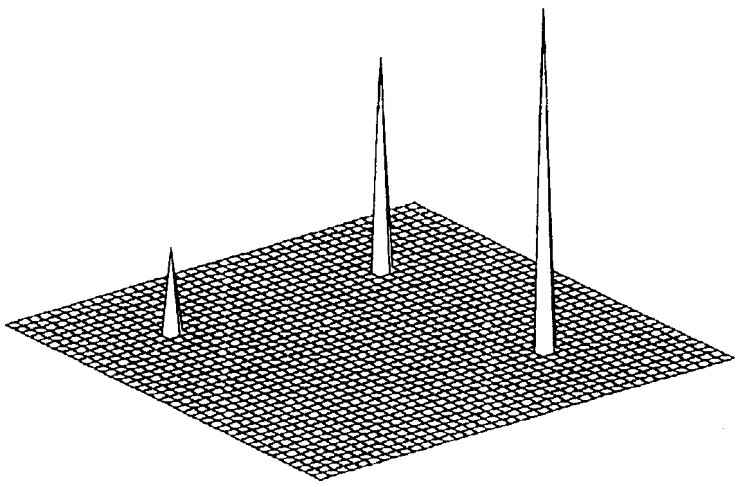
$$a^{\text{approx}}_{(3,3)} = a^{\text{int}}_{(3,3)} - 2w \qquad I^{\text{approx}} = 2wz_i$$

2. No cell states are kept fixed.

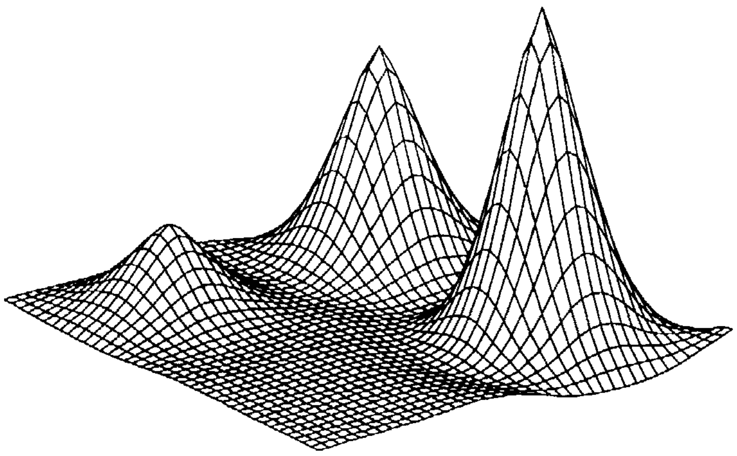
2.5. *Examples*

The first example shows that the interpolating network fits the surface through three given points. Figure 1(a) shows the initial state. There are only three specified points, hence the network should fill the large unknown area. Figures 1(b) and 1(c) show the states after 100τ and 17550τ respectively (where τ is the time constant of the cell). The fitted surface is a flat surface. The reason for the long convergence time is that only a few points were given. This is not a practical example, but the correct operation of the algorithm is demonstrated.

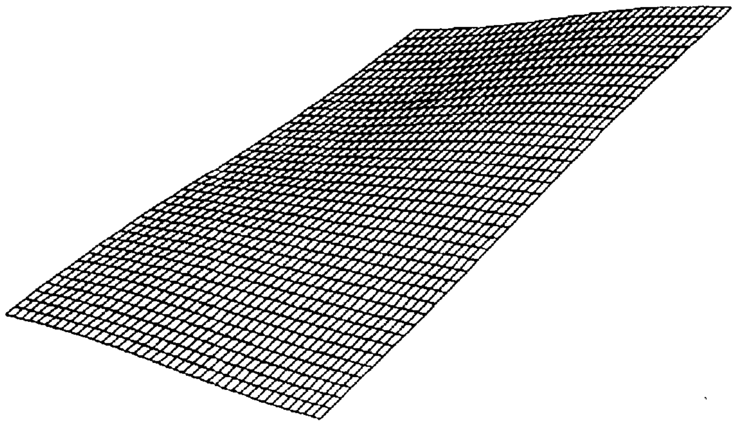
The second example is a ball surface reconstruction. Only 10 per cent of the points are known in advance. Figure 2(a) shows the initial state. Figures 2(b) and 2(c) show the outputs of the network after $0{\cdot}75\tau$ and 30τ respectively. In this case the density of specified points was identical on the whole area and it was higher than before.



(a)



(b)



(c)

Figure 1. Fitting a surface through three given points: (a) initial state; (b) state after 100τ; (c) state after 17550τ

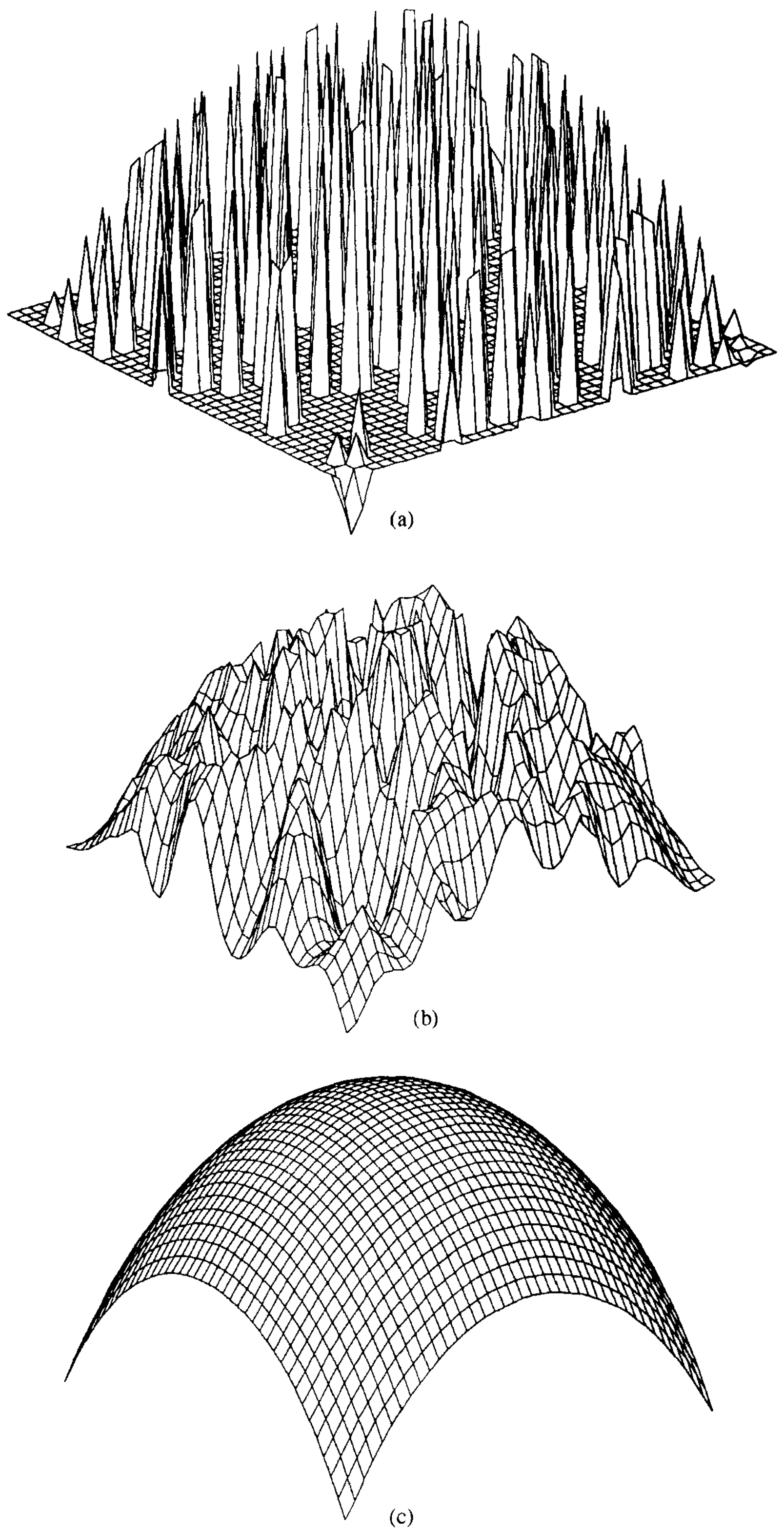


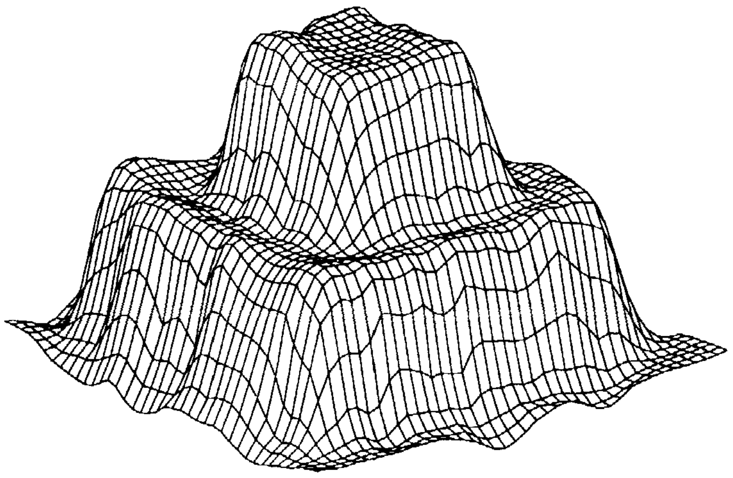
Figure 2. Ball surface reconstruction in case of 10 per cent specified point density: (a) initial state; (b) state after $0{\cdot}75\tau$; (c) state after 30τ

The last example shows the approximation of a *wedding cake* shape. The specified point density was 30 per cent. Two results are shown. In Figure 3(b) the weight factor w was too low. The fitted surface does not fit enough through the given points. Figure 3(a) shows a much better approximation.

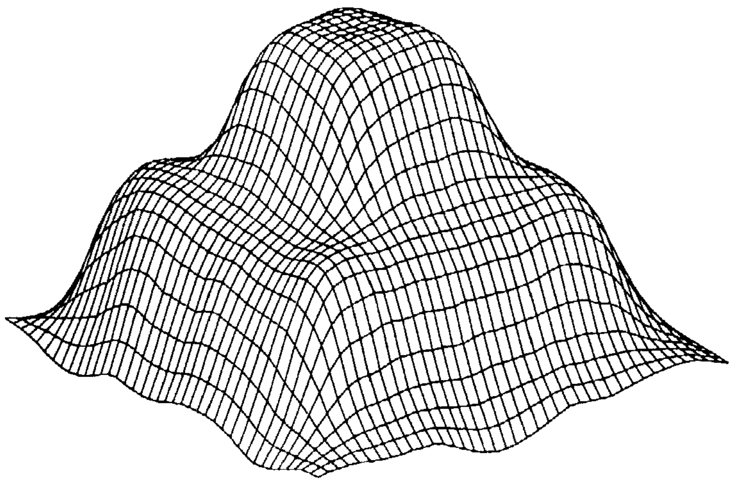
2.6. *Dependence of settling time on sparsity*

In practical applications the settling time is one of the most important parameters. It depends on the positions and quantity of specified points, first of all the density of them. In our example with three points the convergence time was 17550τ, in the ball shape example with 30 per cent specified point density it was only 30τ.

Next an error function is constructed characterizing the distance between the actual and the settled state. In the error functions the following quantities are used: $v_{(i,j)}$, the altitude at grid point (i, j); $\mathbf{v}$, a vector containing the $v_{(i,j)}$-values of the $m \times m$ grid; $w_{(i,j)}$, the altitude at grid point (i, j) in the settled state; $\mathbf{w}$, a vector containing the $w_{(i,j)}$-values of the $m \times m$ grid. Two error measures will be used: (i) the maximum



(a)



(b)

Figure 3. Wedding cake shape reconstruction: Results for weight factor (a) 5 and (b) 0·5

absolute value measure

$$M = \max_{i,j} |w_{(i,j)} - v_{(i,j)}| \tag{16}$$

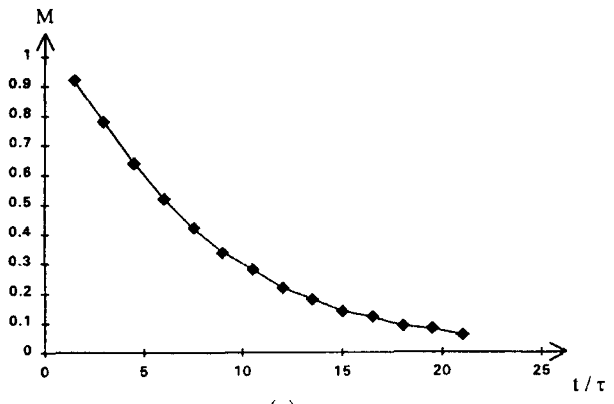
and (ii) the error vector length

$$L = |\mathbf{w} - \mathbf{v}| = \sqrt{\left(\sum_{i,j} (w_{(i,j)} - v_{(i,j)})^2\right)} \tag{17}$$

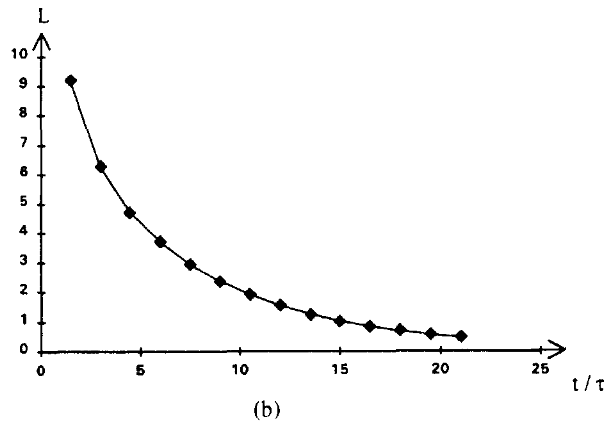
Next M and L will be examined experimentally in the case of ball surface reconstruction with 10 per cent specified point density. The results are shown in Figures 4(a) and 4(b) respectively.

Let us define the settling times in measures M and L as T_M and T_L respectively. T_M and T_L are the times when M and L decrease to 0·05, and 0·25 respectively. (The surface is about one unit high, so $M = 0{\cdot}05$ means 5 per cent maximum error.) The concrete values are not relevant, they were determined to get a practically proper state after T_M and T_L and to show a qualitatively characteristic graph. In Figure 5 experimental values are shown.

If there are areas with different densities, the situation is more complex. Suppose that on one half of the ball 30 per cent of the points are known and on the other half only 10 per cent. In Figures 6(a) and 6(b) the settling process for the maximum measure (M) and the length measure (L) respectively can be seen. In the case of mixed 10 per cent/30 per cent specified point density the convergence was faster than in the case of 10 per cent uniform density but slower than in the case of 30 per cent uniform density.



(a)



(b)

Figure 4. Ball surface reconstruction in case of 10 per cent specified point density: curves of error functions (a) M and (b) L

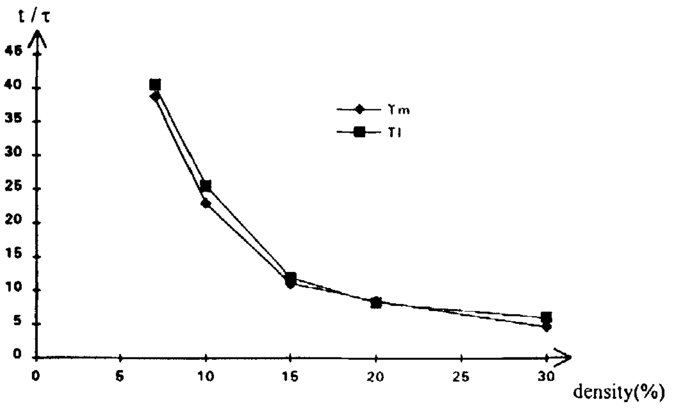
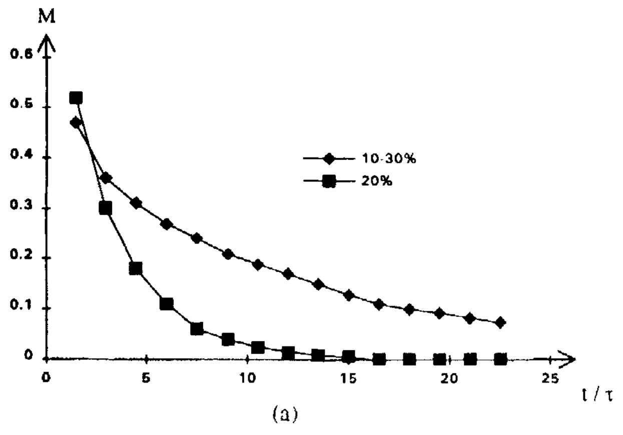


Figure 5. Ball surface reconstruction with different specified point densities: curves of settling times T_M and T_L



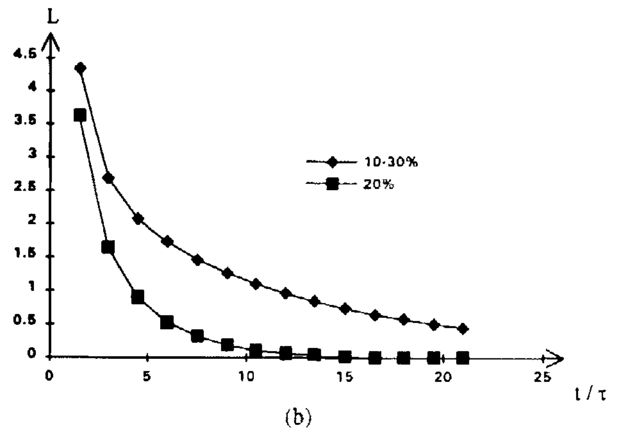


Figure 6. Ball surface reconstruction in case of 20 percent and mixed 10 percent/30 per cent specified point density: curves of error functions (a) M and (b) L

2.7. *Conclusion*

Two surface-fitting methods were realized using CNNs: interpolation and approximation. The networks presented here have only one layer and a simple structure. Omitting parallel processing the new algorithms can achieve higher speed than digital computers that solve this problem using grid methods with serial execution.

The most significant application is surface reconstruction. In many cases, e.g. in 3D shape reconstruction, the characteristic points of an object are known and the surface of the object is requested. The algorithm fits a surface through these characteristic points that is as smooth as possible.

3. SYMMETRIC SPACE-VARIANT OPERATIONS WITH CNNs—AN EXAMPLE: OBJECT ROTATION

3.1. *Introduction*

Space-variant CNN templates in general cause implementation problems. Fortunately, most of the 'useful' space-variant transformations have their own symmetry. Our intent is to implement some space-variant operations on a CNN without losing the benefits of fast programmability and without major modification of the regular space-invariant structure by exploiting their symmetry properties. More exactly, we try to implement operations whose space invariance can be encoded locally cell-by-cell by a singlė value only. In this paper a new method for space-variant processing of binary images is presented.

3.2. *Example: rotation*

Rotation is a useful space-variant operation (e.g. in pattern recognition tasks, when the pattern to be recognized is rotated) which possesses the required symmetry properties. The CNN structure used for the task consists of two layers: one contains the encoded space-variant information, in the present case the circular pathway, and is called the *control layer*; the second carries the actual object to be rotated and is called the *reference layer*. Local analogue arithmetic is used for calculating the inputs of the *decoding function* whose output provides the elements of a space-variant template applied to the reference layer in each position (Figure 7).

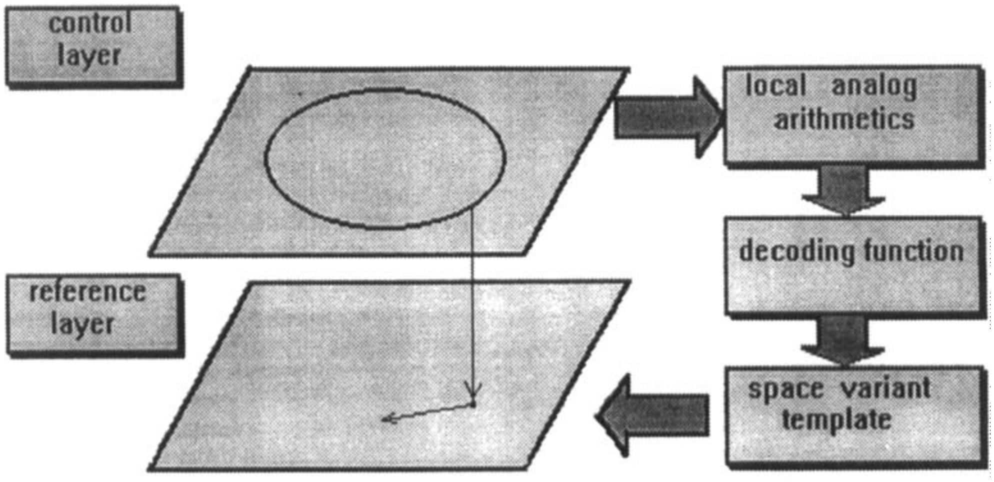


Figure 7. Two-layer CNN structure for space-variant operations

Let us consider the following discrete-time state equation describing the behaviour of the presented structure ($C = 1$, $R = 1$):

$$\begin{aligned} x_{(k+1),\mathrm{ref}} &= (1-h)x_{k,\mathrm{ref}} + h\left(\sum_r \tilde{a}_{ij,kl} v_{yij,\mathrm{ref}} + \sum_r b_{kl} v_{uij,\mathrm{ref}}\right) + I \\ \tilde{a}_{ik,kl} &= \tilde{f}(v_{yij,\mathrm{con}} - v_{ykl,\mathrm{con}}) \end{aligned} \tag{18}$$

where b_{ij} are the elements of the linear control template and $\tilde{a}_{ij,kl}$ are the elements of the space-variant non-linear feedback template. In the present example the space variance is encoded locally as the difference in the neighbouring pixel values, which is the argument of $\tilde{f}$.

3.2.1. Moving single black pixel object on circular pathway. Let us consider a single black pixel object somewhere off centre position on the image. Our purpose is to move the black value on a circular pathway

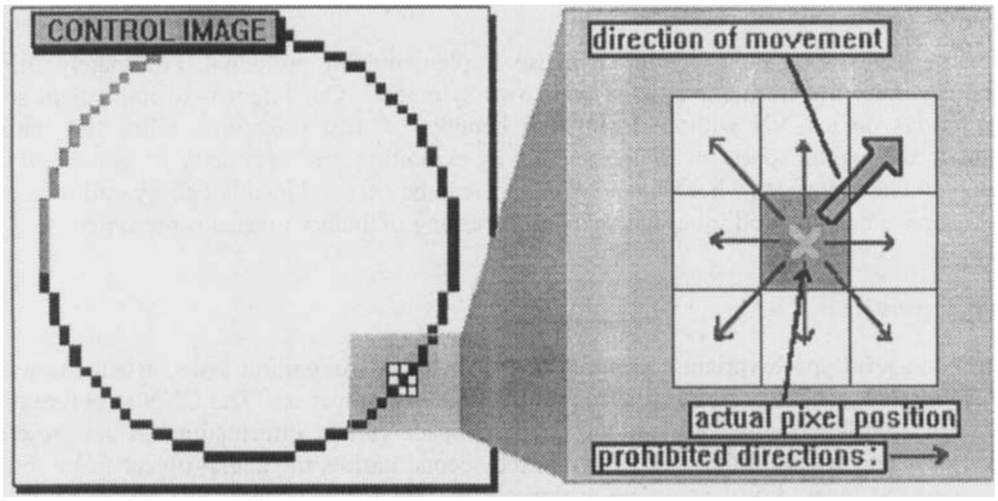


Figure 8. The space variant information needed for rotation can be locally encoded in a so called control image. The black pixel in any position on the circular pathway can only move in one direction of the possible nine and this as well as the speed information can be included in a 3 × 3 window

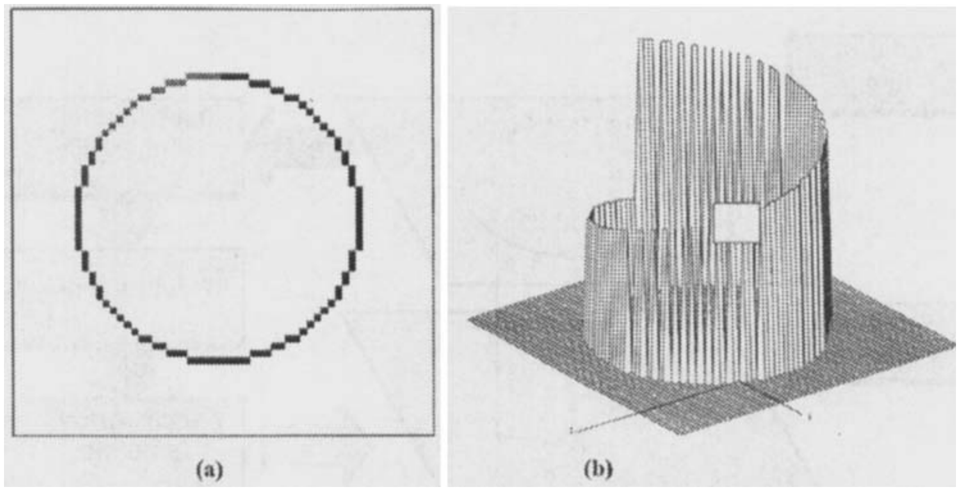


Figure 9. Control image and its 3D representation for moving a single black pixel object on a circular pathway. Details of the blank window are shown in Figure 10

around the centre. The direction and speed of movement depend on the pixel position with respect to the centre. This information can be encoded in a 3×3 window; in other words, the rotation can be coded locally (Figure 8). The direction coding needed for moving a black pixel on a circular pathway can be included in a circular step function as shown in Figure 9.

The step size of this function is a well-determined value δ. Therefore in the window in Figure 8 the differences in pixel values with respect to the central element are δ in the direction of movement, $-\delta$ in the opposite direction and significantly different (abs(difference) $\gg \delta$) in all other directions (prohibited directions).

In order to understand how the direction information can be gained from the control image and realized as space-variant templates, let us consider the well-known CCD template. It contains only three values different from 0: the central 2, -1 in the direction of movement and 1 in the opposite direction. Normally it moves black points on a straight path, but if we could relocate the -1 and 1 values depending on the position, we could make it space-variant. This can be achieved using the control image described above and a simple non-linear decoding function (Figure 11).

The possible differences in the given neighbourhood and the corresponding outputs of the decoding function are

$$a_{ij,kl} = \tilde{f}(v_{ij,\mathrm{con}} - v_{kl,\mathrm{con}}) = \begin{cases} 0 & \text{if } v_{ij,\mathrm{con}} - v_{kl,\mathrm{con}} = 0 \\ 1 & \text{if } v_{ij,\mathrm{con}} - v_{kl,\mathrm{con}} = \delta \\ 1 & \text{if } v_{ij,\mathrm{con}} - v_{kl,\mathrm{con}} = -\delta \\ 0 & \text{if } v_{ij,\mathrm{con}} - v_{kl,\mathrm{con}} \gg \delta \end{cases} \tag{19}$$

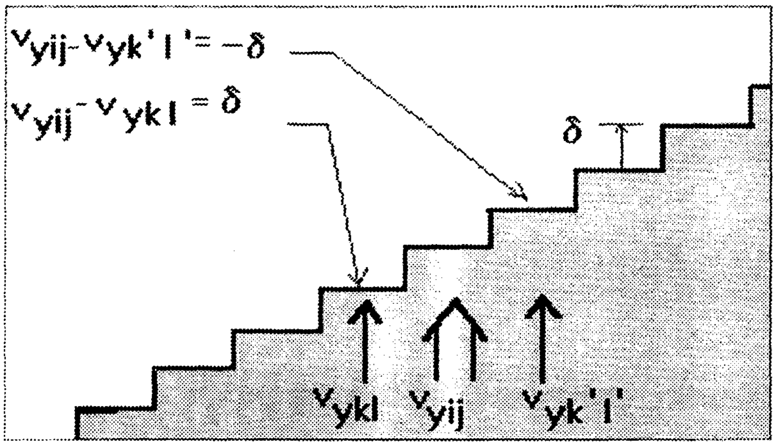


Figure 10 A slice of the circular pathway in the position of the blank window on Figure 9

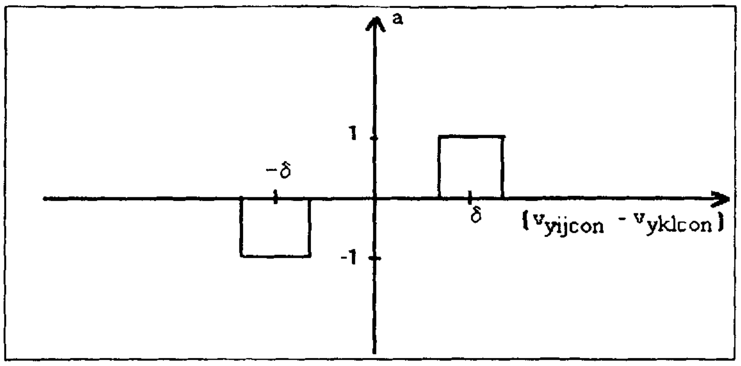
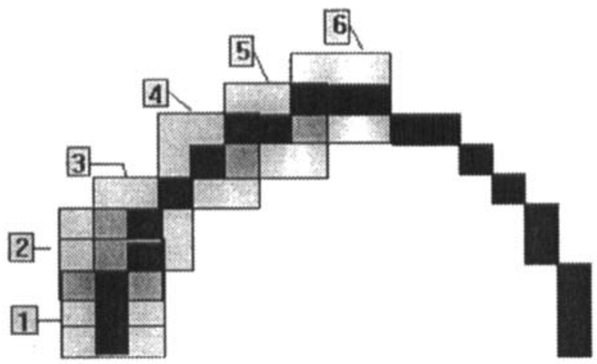


Figure 11 The non-linear function which assigns the necessary template values to the difference in control picture pixel values

Figure 12 shows how the decoding function assigns proper values to the CCD template element in each position according to the differences between the pixel values on the control image within the 3 × 3 window.

3.2.2. Introducing speed adjustment. Expressing the pixel co-ordinates in polar co-ordinates with the centre of rotation as the origin, the direction of movement depends on the angular variable while the speed depends on the distance from the centre. This latter property can also be expressed by applying a somewhat more complicated non-linear template function and multiple step-functions in which the speed is encoded in the step size. Figure 13 shows the decoding function in the case of two points with different radial distances. The relevant control image can be seen in Figure 14.

Another example can be seen in Figure 15 for moving objects of any size on the image within the outermost path with no speed adjustment.



$$\begin{bmatrix} a & a & a \\ a & 2 & a \\ a & a & a \end{bmatrix}$$

$$\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \\ \begin{bmatrix} 0 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & -1 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 1 & 0 \\ 0 & 2 & 0 \\ -1 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 1 \\ 0 & 2 & 0 \\ 0 & -1 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 1 \\ 0 & 2 & 0 \\ -1 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 1 \\ -1 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 \\ -1 & 2 & 1 \\ 0 & 0 & 0 \end{bmatrix} \end{array}$$

Figure 12. Position-dependent values of CCD template values are calculated as the output of a non-linear decoding function. In positions where in a 3 × 3 window on the pathway the differences in the neighbouring pixel values are δ and $-\delta$, the substituted values are 1 and −1 respectively; in all other directions in which the differences are significantly different, 0s are substituted in templates applied to the corresponding positions on the reference image

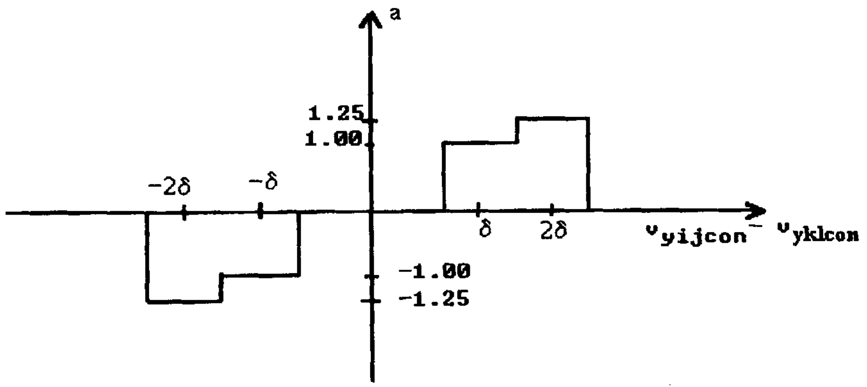


Figure 13. Encoding the radial distance in the non-linear function which substitutes the properly weighted values in the space-variant template for moving two pixels of different distances from the centre on the circular pathway with the same angular velocity

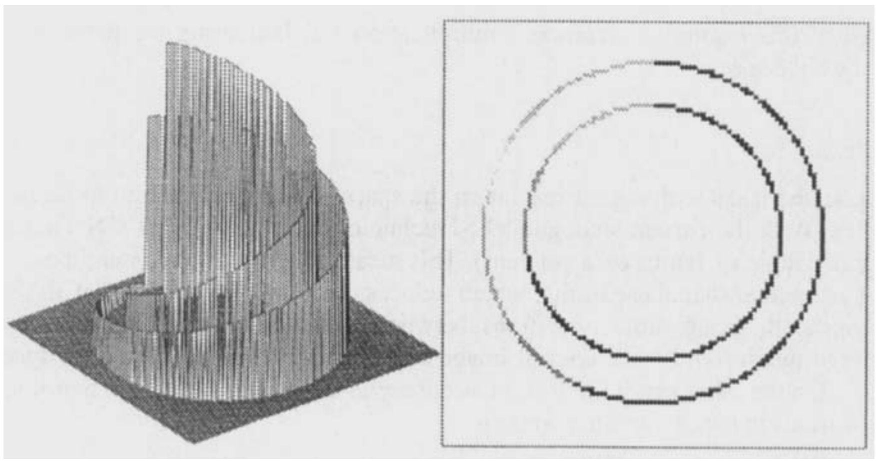


Figure 14. Control image and its 3D representation for moving two pixels parallelly with speed adjustment

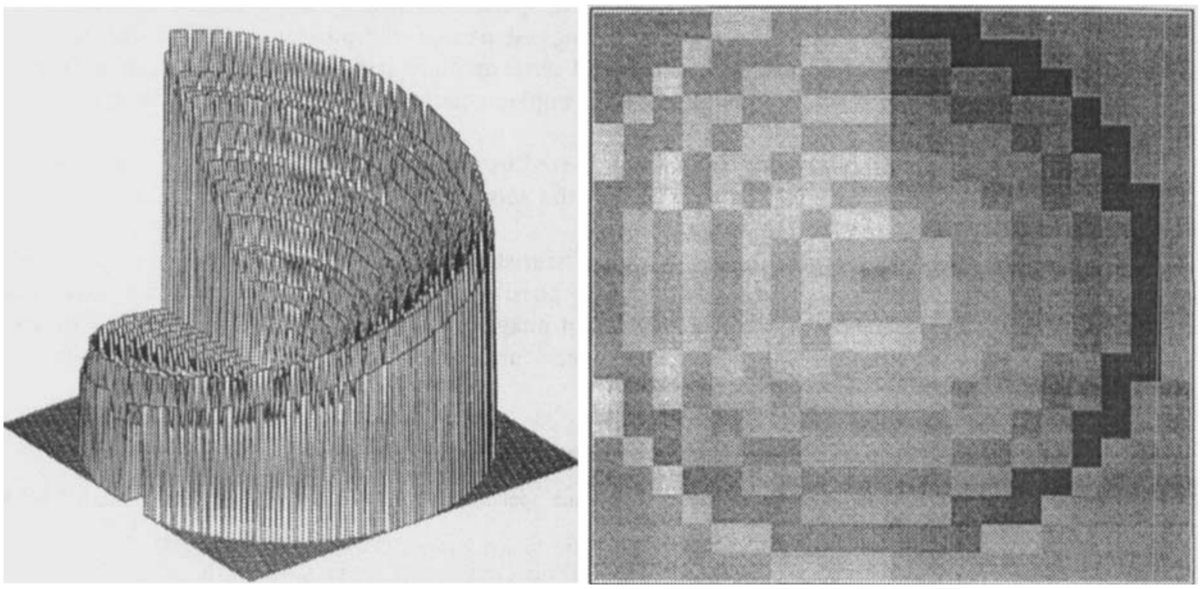


Figure 15. Control image and its 3D representation for moving full-screen objects

3.3. *Generalization*

The arrangement used for object rotation can be generalized. In this way we can implement several other space-variant operations which can be locally encoded (e.g. diminishment, enlargement).

Note that the space variant template can in general be in either feedback or feedforward position and that the carrier of the space-variant information can be either the input or the output of any layer in a multilayer CNN structure, which enables us to implement recursive processes. The space variance can also be encoded in various ways. The generalized form of (18) is

$$\tilde{a}_{ij,kl} = \tilde{f}(v_{yij,\text{con}} \otimes v_{ykl,\text{con}}) \tag{20}$$

We can define numerous basic operations denoted by ⊗ in equation (20) between the pixels of the control layer within the given neighbourhood. These can be any simple, locally realizable operation such as subtraction (used in this example), addition, multiplication etc. Increasing the neighbourhood is another way to enlarge the code space.

3.4. Practical limitations

In more complicated tasks with higher resolution the space-variant information to be encoded gets more and more complex. With the current analogue VLSI technology the accuracy of CNN computations cannot exceed 7–8 bits (a couple of tenths of a per cent). This means that in the given neighbourhood the distance of codes cannot be smaller than these limits, which reduces the number of choices of different codes. With a wider neighbourhood, using more operations between the pixel values can increase the number of variations, although the design of the control image and the optimization in the code space become more and more difficult. On the other hand, the lack of accuracy in the realization of the non-linear function may cause severe distortions in space-variant operation.

3.5. Conclusion

Our purpose is to exploit the CNN's high computing speed provided by the analogue dynamics. This means that in a compound task we try to solve the biggest part of the problem possible with the CNN without unnecessary data transfers between parallel and serial architectures. However, there can be subtasks such as space-variant operations which may cause implementation problems with the regular space-invariant CNN structure.

As a space-variant operation, rotation has been achieved using two space-invariant CNN layers, one of which is used as a local space-variant control of the template values. This method can be further generalized.

Our method for designing control images is rather heuristic at this stage of research, though in some special cases straightforward computing methods seem possible. There are further exploitable possibilities in the recursive algorithms when we modify the control image during the process. The behaviour of these algorithms, however, is extremely sophisticated and needs further examination.

REFERENCES

1. L. O. Chua and L. Yang, 'Cellular neural networks: theory and applications', *IEEE Trans. Circuits and Systems*, **CAS-35**, 1257–1290 (1988).
2. L. O. Chua and T. Roska, 'The CNN paradigm', *IEEE Trans. Circuits and Systems*, **CAS-40**, 147–156 (1993).
3. T. Roska and L. O. Chua, 'The CNN universal machine', *IEEE Trans. Circuits and Systems*, **CAS-40**, 163–173 (1993).
4. T. Roska and L. O. Chua, 'Cellular neural networks with non-linear and delay-type template elements and non-uniform grids', *Int. j. cir. theor. appl.*, **20**, 469–481 (1992).
5. D. Terzopoulos, 'Multilevel computational processes for visual surface reconstruction', *Comput. Vision, Graphics, Image Process.*, **24**, 52–96 (1983).
6. W. E. L. Grimson, 'An implementation of a computational theory of visual surface interpolation', *Comput. Vision, Graphics, Image Process.*, **22**, 39–69 (1983).
7. R. Courant and D. Hilbert, *Methods of Mathematical Physics*, Vol. I., Interscience, New York, 1953, pp. 250–252.
8. M. Csapodi, L. Nemes, G. Tóth, T. Roska and A. Radványi, 'Some novel analogic CNN algorithms for object rotation, 3D interpolation–approximation, and a 'door-in-a-floor' problem', *Proc.CNNA-94*, IEEE, Rome, 1994, pp. 435–439.