ARBITRARY SPATIAL CONVOLUTION VIA CNN UNIVERSAL MACHINE WITH 3 X 3 TEMPLATES: METHODS AND ISSUES

by

Kenneth R. Crounse and Leon O. Chua

Memorandum No. UCB/ERL M96/5

22 January 1996

ARBITRARY SPATIAL CONVOLUTION VIA CNN UNIVERSAL MACHINE WITH 3 X 3 TEMPLATES: METHODS AND ISSUES

by

Kenneth R. Crounse and Leon O. Chua

Memorandum No. UCB/ERL M96/5

22 January 1996

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Arbitrary Spatial Convolution via CNN Universal Machine with 3 × 3 Templates: Methods and Issues

Kenneth R. Crounse* Leon O. Chua †

January 22, 1996

Abstract

We present a canonical form for the possible spatial linear filtering operations which can be performed on the CNN Universal Machine [1] when running a finite algorithm using single-transient CNN spatial convolution and image addition. Convolution by arbitrary kernels of any finite size can be shown to be implementable by algorithms of this form using only 3×3 templates. Equivalently it can be considered a general form capable of implementing arbitrarily large B-template effects by using only 3 × 3 connectivity. Methods for performing arbitrary convolution involving either summations of multiple applications of small B-templates[2] or partitioning and shifting[3] can be unified under this form. When implementing a desired convolution with this form the choice of templates is under-determined and either of these previous methods can be used to demonstrate completeness. A CNNUM algorithm is given for implementing the partition-shift approach with special concern given to practical issues of addition and accuracy. It is presented both as a constructive proof of the general convolution capabilities of the CNNUM and as a practical approach in some cases.

1 Introduction

In recent years, there has been much interest in using large scale homogeneous cellular arrays of simple circuits to perform image processing tasks. The Cellular Neural Network (CNN), first introduced[4] as an implementable alternative to fully connected neural networks, has evolved into a paradigm for these types of arrays[5]. The CNN Universal Machine (CNNUM) architecture [1] allows the results of CNN operations to be used in further processing stages.

As linear filtering is the workhorse of image processing algorithms, the question of whether the CNNUM can implement any convolution and how to do so is important. It has long been understood that the B-template of the simple CNN performs a correlation (reflected convolution) with the input image. However, due to implementation concerns the B-template, and therefore the convolution kernel, is restricted to be small in size – typically 3 × 3. By also using the A-template[6, gives an overview], or by cascading B-template operations, large kernel convolutions can be performed but the coefficients of the impulse response cannot be arbitrarily specified. However, the need to

*Sponsored under the Joint Services Electronics Program, Contract Number F49620-94-C-0038. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation herein.

†The authors are with the Electronics Research Laboratory, University of California at Berkeley, 94720.

exactly specify a large convolution mask arises in many situations, such as template matching for object detection, dilation or erosion by large structuring elements, and interpolation.

To overcome this problem it has been demonstrated that several series of small B-templates can be applied to the input, and the results combined, to implement larger effective B-templates[2]. The form of the decomposition, and a related specific example, was given for which the effect of arbitrary 5×5 template could be produced from five 3×3 templates. It was also observed that in principle such an approach could be generalized in form for the production of arbitrarily larger effective templates. The proof relies on induction, that is, a 7×7 template could be produced by series of 5×5 templates, etc[7]. To be usable, a systematic method for determining and computing an appropriate solution to a large system of under-determined equations needs to be devised. A second approach to overcome the problem is the partition and shift method introduced in [3]. The technique is to partition the desired convolution kernel into 3×3 blocks. These weighted sums can be performed by the B-template and then shifted to the correct output location, where they are accumulated. In fact, this technique can be considered a particular choice of solution of the generalized form.

Here we present a unified form for these techniques which includes a broader class of implementations, for instance, those that use the A-template. We demonstrate how the elementary operations used by this form can be implemented on the CNN Universal Machine. A simple constructive algorithm based on the partition-shift method is given to implement arbitrary convolutions (equivalently, arbitrarily large effective B-templates) to demonstrate completeness. The advantages of this choice are conceptual simplicity, ease of determining the decomposition, and ready application to nonlinear B-templates (where the template elements are operators). In addition, an example is given to illustrate some practical issues.

Spatial convolution is discussed in Section 2. The general form for performing convolutions on the CNNUM is given in Section 3. In Section 4 it is shown how each operation can be performed on the CNNUM. Finally, in Section 5, a formal approach for the automation of convolutions with desired kernels on the CNNUM is given. The use of this algorithm is demonstrated by an example which also includes some useful enhancements.

2 Spatial Convolution

Let $u(n_1, n_2)$ be the two-dimensional input sequence. Typically, on part of its support, this will be an image of interest. Areas off the image will be defined to be zero for our purposes. The output $y(n_1, n_2)$ of the filtering operation is defined by the spatial convolution as follows:

$$y(n_1, n_2) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} h(n_1 - k_1, n_2 - k_2) u(k_1, k_2) \tag{1}$$

where $h(k_1, k_2)$ is known as the convolution 'kernel', 'mask', or 'impulse response'. A reasonable way of interpreting this equation is that each pixel of the output is formed by a weighted average of the input around that pixel. Typically, convolution by a mask is considered to be a functional mapping of the input sequence to the output sequence and so the notation

$$y(n_1, n_2) = h * u(n_1, n_2)$$

is useful.

Two important facts about convolution are now stated:

$$h_2 * (h_1 * u(n_1, n_2)) = (h_2 * h_1) * u(n_1, n_2)$$

$$h_1 * u(n_1, n_2) + h_2 * u(n_1, n_2) = (h_1 + h_2) * u(n_1, n_2)$$

3 Canonical Form for CNNUM Spatial Convolution

We now introduce a canonical form for the linear spatial filtering operations which can be performed on the CNNUM. Let H be a spatial filtering mask which can be realized by a single set of CNN templates. The output of such an operation can be combined with other outputs by image addition or can be filtered again by another set of templates. A complex signal flow diagram of CNN filter blocks and addition units can be built up in this way. The implementation of these blocks is described in Section 4. Scalar multiplication, and specifically negation, can easily be accounted for in the CNN filter or addition blocks.

Now, by using the properties of convolution described above, these complex networks of cascading convolutions and summations can be written in the *sum of products* form:

$$\begin{aligned} y(n_1, n_2) = & \\ & H_{1,1} * H_{1,2} * \ldots * H_{1,P_1} * u(n_1, n_2) \\ & + H_{2,1} * H_{2,2} * \ldots * H_{2,P_2} * u(n_1, n_2) \\ & + \\ & \vdots \\ & + H_{Q,1} * H_{Q,2} * \ldots * H_{Q,P_Q} * u(n_1, n_2) \end{aligned} \tag{2}$$

Then, by using the properties of convolution above, the effective convolution kernel implemented by the CNNUM can be identified as:

$$\begin{aligned} h(n_1, n_2) = & \\ & H_{1,1} * H_{1,2} * \ldots * H_{1,P_1} \\ & + H_{2,1} * H_{2,2} * \ldots * H_{2,P_2} \\ & + \\ & \vdots \\ & + H_{Q,1} * H_{Q,2} * \ldots * H_{Q,P_Q} \end{aligned} \tag{3}$$

This equation describes the span of filtering operations which can be performed on the CNNUM although it may not be the most efficient or robust implementation. The methods presented in [2] and [3] can both be considered examples of this form. Even if the H are restricted to the FIR filtering of a B-template with 5-neighbor connection pattern, this form can be shown to be able to implement any desired (finite) impulse response. The choice of templates is, in fact, underdetermined for a given desired convolution as P and Q are left unspecified by the form. A proof was mentioned in [2] which depends on an inductive approach[7]. A constructive argument was given in [3] by which most of the convolutions are shifting operations. Since this partition-shift method is the most straightforward technique, an algorithm for its implementation on the CNNUM is given in this paper.

For the direct implementation of the canonical form, the only necessary operations are the single-transient spatial convolutions and accumulating of intermediate results. The next section focuses on how to implement these steps by the CNN Universal Machine.

4 Necessary CNN Operations

The simplest CNN cell has a single capacitor, giving it first-order dynamics, and is coupled to neighboring cells through non-linear controlled sources. A single transient of the CNN is described by:

$$\frac{d}{dt}x_{i,j}(t) = -x_{i,j}(t) + \sum_{k,l\in\mathcal{N}} A_{k,l}y_{i+k,j+l}(t) + \sum_{k,l\in\mathcal{N}} B_{k,l}u_{i+k,j+l} + I \tag{4}$$

where $\mathcal{N} = \{-r,\cdots,0,\cdots,r\}$ is the set of indices[1] for which the templates are defined, and with saturation output nonlinearity

$$y(x) = \frac{1}{2}\left[|x-1| - |x+1|\right]$$

The input, state, and output, represented by $u_{i,j}, x_{i,j}$,and $y_{i,j}$ respectively, are defined on $0 \le i \le N_1$ and $0 \le j \le N_2$. In a real circuit there are various ways to deal with the boundary.

Due to implementability concerns, the template neighborhood radius is generally restricted to be as small as possible and the templates are applied in a space-invariant manner. The methods shown here use templates no larger than 3×3, i.e. $r = 1$, although the ideas apply to larger neighborhoods as well.

The elementary image processing tasks performed on the input data by a single template set can be combined into more complicated operations by an invention known as the CNN Universal Machine[1]. The machine uses the simple CNN in a time-multiplexed fashion, analogous to the ALU of a microprocessor, by controlling the template weights and the source of the data inputs for each operation. The machine supplies memory and register transfers at each cell which allow the outputs of simple CNN operations to be combined and/or supplied to the inputs of the next operation, thereby allowing more complex algorithms to be implemented. For our purposes, we assume that the machine allows output (y, x) to be sent to and inputs $(x(0), u, I)$ taken from two local analog memories (LAM1, LAM2) which are used for temporary storage.

4.1 Single Transient Spatial Convolution

It has long been understood that by choosing $A = 0$ for the A-template, the CNN performs a spatial correlation with the B-template. This can be seen by setting the dynamics of Equation 4 equal to zero giving:

$$x_{i,j}(\infty) = \sum_{k,l\in\mathcal{N}} B_{k,l}u_{i+k,j+l} \tag{5}$$

This CNN processing step will be written as $B \star u$. Since convolution can be considered a correlation with the reflected convolution kernel, this step is equivalent to convolution by the reflected version of the B-template.

By using the A-template, convolutions with arbitrarily large impulse responses can be accomplished in a single transient. And, although the following discussion does not consider these possibilities, such templates may be very useful.

[1] Alternatively, the sums can be written in shifted-template form. The notation is then

$$\frac{d}{dt}x_{i,j}(t) = -x_{i,j}(t) + \sum_{(k,l)\in\mathcal{N}(i,j)} A_{k-i,l-j}y_{k,l}(t) + \sum_{(k,l)\in\mathcal{N}(i,j)} B_{k-i,l-j}u_{k,l}(t) + I$$

where $\mathcal{N}(i,j) = \{(k,l) : k-i, l-j \in \{-r,\cdots,0,\cdots,r\}\}$ is the set of indices for the cells in the r-neighborhood of cell (i,j). This was presented incorrectly in [6, page 584].

4.2 Shifting

Spatial translation of images by a single pixel can be considered a particular example of the correlations in the previous subsection. And, therefore shifting over a distance can be considered a cascade of convolutions. However, because this particular operation is so important, it is mentioned with special consideration.

Shifting of results can easily be performed by the CNN by use of a the B-template or by special shifting hardware within the CNNUM implementation. Implementations which support hardware shifting have been considered for various applications in addition to the fact that some one-directional shifting is almost always supplied by the read in/out circuitry.

It is well-known that by supplying the image to be shifted to the CNN input, the simple B-template

$$B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \tag{6}$$

with $A = 0$ will supply a left-shifted image to the CNN output. That is,

$$x_{i,j}(\infty) = u_{i+1,j} \tag{7}$$

Templates for shifts in the other 7 directions are obvious. This output can then be transferred to the input for further shifting. When shifting from the boundary, zeros are shifted into the array.

The CNNUM processing steps for shifting an image I cells to the right and J cells down (by whichever of the methods) will be written as SHIFT(I, J,image). Note that if diagonal shifting is available (as in the B-template shifting method) this will take only $\max(|I|, |J|)$ shifting steps plus the necessary register transfers needed to setup the shift.

4.3 Accumulating

Summing of images can be performed in various ways depending on assumptions about the CNNUM hardware. We mention three approaches.

The simplest method would be a hardware analog adder at the Local Analog Output Unit (LAOU) of each cell. This is currently being implemented in next generation CNNUMs.

The second-to-simplest requires two forms of input to the dynamics. For instance the standard input and a space variable bias. These two inputs would be added together directly into the dynamics making addition trivial by choosing $A = 0$.

The third method only requires the ability to halt the dynamics of the CNN at a specified time during the transient. This is a capability which has been incorporated into current CNNUM designs. Let

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1+a & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{8}$$

$$B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{9}$$

so that the CNN dynamics can be written in the linear region as

$$\frac{d}{dt}x_{i,j}(t) = ax_{i,j}(t) + bu_{i,j} \tag{10}$$

A straightforward way to perform addition with these templates is the special case[2]. of choosing $a = 0$. Then the solution to Equation 10 is

$$x_{i,j}(t) = x_{i,j}(0) + tbu_{i,j} \tag{11}$$

So, for instance, depending on the sign of b, stopping the transient at $t = 1/|b|$ provides either the sum or difference of the initial condition and the input images. Some scaling might be necessary beforehand to insure that the result fits within the linear region of the dynamics.

A more general approach is to choose $a < 0$ with $b = -a$. Then the solution to Equation 10 is

$$x_{i,j}(t) = e^{at}x_{i,j}(0) + (1 - e^{at})(-b/a)u_{i,j} \tag{12}$$

If the system is stopped at time t' the state can be written as the convex sum

$$x_{i,j}(t') = \alpha x_{i,j}(0) + (1 - \alpha)u_{i,j} \tag{13}$$

where

$$\alpha = e^{at'} \in (0, 1] \tag{14}$$

This approach has the advantage that the result will always lie in the linear region if the inputs do. The CNNUM processing steps for performing the convex sum of two images (by whichever of the methods) will be written as $\mathrm{ADD}(\alpha, x_0, u)$.

5 CNNUM Partition-Shift Algorithm

First, we present the most general algorithm for implementing convolution by a desired kernel $h(n_1, n_2)$ using the partition-shift method. More efficient algorithms are possible, rather, the intent is to demonstrate that the convolution implementation procedure can be automated for an arbitrary desired convolution kernel.

By making the following definitions:

$$B^{k_1,k_2}(l_1, l_2) = \begin{cases} h(-l_1 - 3k_1, -l_2 - 3k_2) & \text{for } -1 \le l_1, l_2 \le 1 \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

and

$$Y^{k_1,k_2}(m_1, m_2) = \sum_{l_1=-1}^{1} \sum_{l_2=-1}^{1} B^{k_1,k_2}(l_1, l_2)u(l_1 + m_1, l_2 + m_2) \tag{16}$$

the spatial convolution can be written as:

$$y(n_1, n_2) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} Y^{k_1,k_2}(n_1 + 3k_1, n_2 + 3k_2) \tag{17}$$

which is just the sum of shifted 3×3 correlations. Note that this decomposition can be considered to be in the canonical form, Equation 2, by recognizing that shifting operators can be written as simple B-template convolutions.

Now, this decomposition can be implemented directly. Choose M and N so that the desired impulse response fits snugly inside $\{(n_1, n_2) : -3M - 1 \le n_1 \le 3M + 1, -3N - 1 \le n_2 \le 3N + 1\}$. The k_1, k_2th B-template can now be found by Equation 15. Then, the convolution can be performed by the following CNNUM algorithm:

[2]The $a = 0$ case was first pointed out to the authors by Ákos Zarándy. Later, a particular example came to our attention of a similar approach used to implement the sums of a 4×4 convolution[8].

LAM1 $\leftarrow 0$
for $k_1 = -M$ to M
for $k_2 = -N$ to N
$Y^{k_1,k_2} = B^{k_1,k_2} \star U$
LAM2 $\leftarrow$ SHIFT$(-3k_1, -3k_2, Y^{k_1,k_2})$
LAM1 $\leftarrow$ ADD(LAM1, LAM2)
end
end

Note that a $3(2M+1) \times 3(2N+1)$ convolution can be implemented in $(2M+1)(2N+1)$ loop iterations. Each of these involves approximately $1 + 3\max(|k_1|, |k_2|) + 1$ CNN operations. For instance, a full 21×21 convolution can be performed in about 434 transients, and a full 9×9 convolution can be performed in about 42 transients – all independent of the input image size of the CNN.

Depending on the noise properties and speed of the CNNUM implementation under consideration, some improvements to the basic algorithm as presented will need to be considered.

First, some of the 3×3 B-templates might have only very little energy. It may be wise to scale-up the template values to improve signal to noise ratio during shifting and accumulating. Then, they can be scaled back to their original size during further accumulation.

Second, if the kernel is sparse or lopsided, it will not be necessary to perform and shift blocks corresponding to zero templates.

One might wonder if it is necessary to shift the partial correlation results immediately to their destination but instead accumulate them with other partial results on the way home. The number of shifts could be drastically reduced by such a method. However, there are some complicated boundary management problems when partially accumulated results get shifted off the array. In principle this could be solved by some complex buffering scheme.

It is also worth mentioning that there is an equivalent general algorithm to the one described above where the input, rather than the intermediate results, is shifted about the array to perform the partial correlations.

An Example

Figure 1 shows an impulse response for which we will write the CNNUM algorithm. The function is mostly zero excepting along a line which ends in a blob. This might be considered a simulation of motion blur. Figure 2 shows the 9×9, albeit sparse, B-template which must be implemented on the CNN to perform the specified convolution. In order to implement this on a CNN with 3×3 templates, the desired template is partitioned according to Equation 15, as shown in the figure. Since most of these templates have all zero entries, they do not need to be performed. The whole process of performing the convolution can be written out:

$Y^{-1,1} = (10 \times B^{-1,1}) \star U$
LAM1 $\leftarrow$ SHIFT$(3, 3, Y^{-1,1})$

$Y^{0,0} = (5 \times B^{0,0}) \star U$
LAM2 $\leftarrow Y^{0,0}$
LAM1 $\leftarrow$ ADD$(\frac{1}{3}$,LAM1, LAM2)

$Y^{0,1} = (3\frac{1}{3} \times B^{0,1}) \star U$
LAM2 $\leftarrow$ SHIFT$(0, -3, Y^{0,1})$
LAM1 $\leftarrow$ ADD($\frac{1}{2}$,LAM1, LAM2)

$Y^{-1,0} = (2\frac{1}{2} \times B^{-1,0}) \star U$
LAM2 $\leftarrow$ SHIFT$(3, 0, Y^{-1,0})$
LAM1 $\leftarrow$ ADD($\frac{6}{10}$,LAM1, LAM2)

The total number of CNN operations can be estimated to be about 17, although it will depend on the Universal Machine implementation. Note that each B-template was scaled so that the correlations could take advantage of the full dynamic range. The partial results were then re-scaled when being accumulated. The partial accumulated sum was also kept at maximal scale throughout the calculation. These scalings are likely to improve the overall signal-to-noise ratio during each calculation. Finally, care was taken to perform the accumulation of templates which have the smallest overall effect first as a heuristic to form meaningful summations.

6 Conclusion

A general canonical form describing the span of linear spatial filtering operations achievable on the CNNUM was given. It is a superset of previous methods which were shown to be complete even when only using 3×3 templates, and therefore the CNNUM is known to be able to implement any desired convolution.

We have constructively demonstrated how this can be done by choosing a specific solution of the canonical form. When using the shift-accumulate method the number of transients needed increases dramatically with kernel size. The point at which the scheme becomes impractical will depend on the speed and analog error of the particular CNNUM implementation. It is our expectation that this approach will be most useful when implementing templates that have a complicated non-zero support such as those used for binary shape detection by correlation, those with a sparse but broad spatial extent, or those that are asymmetric. Other convolutions will probably be better implemented by using other solutions to the canonical form, for instance by using CNNs with A-templates.

Acknowledgments

The authors would like to thank Prof. Frank Werblin for suggesting that we investigate this problem and Prof. Tamas Roska for his advice regarding the manuscript.

References

[1] T. Roska and L. O. Chua, "The CNN Universal Machine: An analogic array computer," *IEEE Transactions on Circuits and Systems - II*, vol. 40, pp. 163–173, Mar. 1993.

[2] K. Ślot, "Large-neighborhood template implementation in Discrete-time CNN Universal Machine with a nearest-neighbor connection pattern," in *Proceedings of the Third IEEE International Workshop on Cellular Neural Networks and Their Applications*, pp. 213–218, Dec. 1994.

[3] S. Schwarz, A. Rotter, and W. Mathis, "Composition and decomposition of local operators for Cellular Neural Networks," in *Kleinheubacher Berichte, Band 37*, pp. 253–262, Oct. 1993.

[4] L. O. Chua and L. Yang, "Cellular Neural Networks: Theory," *IEEE Transactions on Circuits and Systems*, vol. 35, pp. 1257–1272, Oct. 1988.

[5] L. O. Chua and T. Roska, "The CNN paradigm," *IEEE Transactions on Circuits and Systems – I*, vol. 40, pp. 147–156, Mar. 1993.

[6] K. R. Crounse and L. O. Chua, "Methods for image processing and pattern formation in Cellular Neural Networks: A Tutorial," *IEEE Transactions on Circuits and Systems – I*, vol. 42, pp. 583–601, Oct. 1995.

[7] K. Ślot. Personal communication. Feb. 1, 1996.

[8] O. Moreira-Tamayo and J. Pineda de Gyvez, "Wavelet transform coding using Cellular Neural Networks," in *Int'l Symposium on Nonlinear Theory and Its Applications*, pp. 541–544, Dec. 1995.

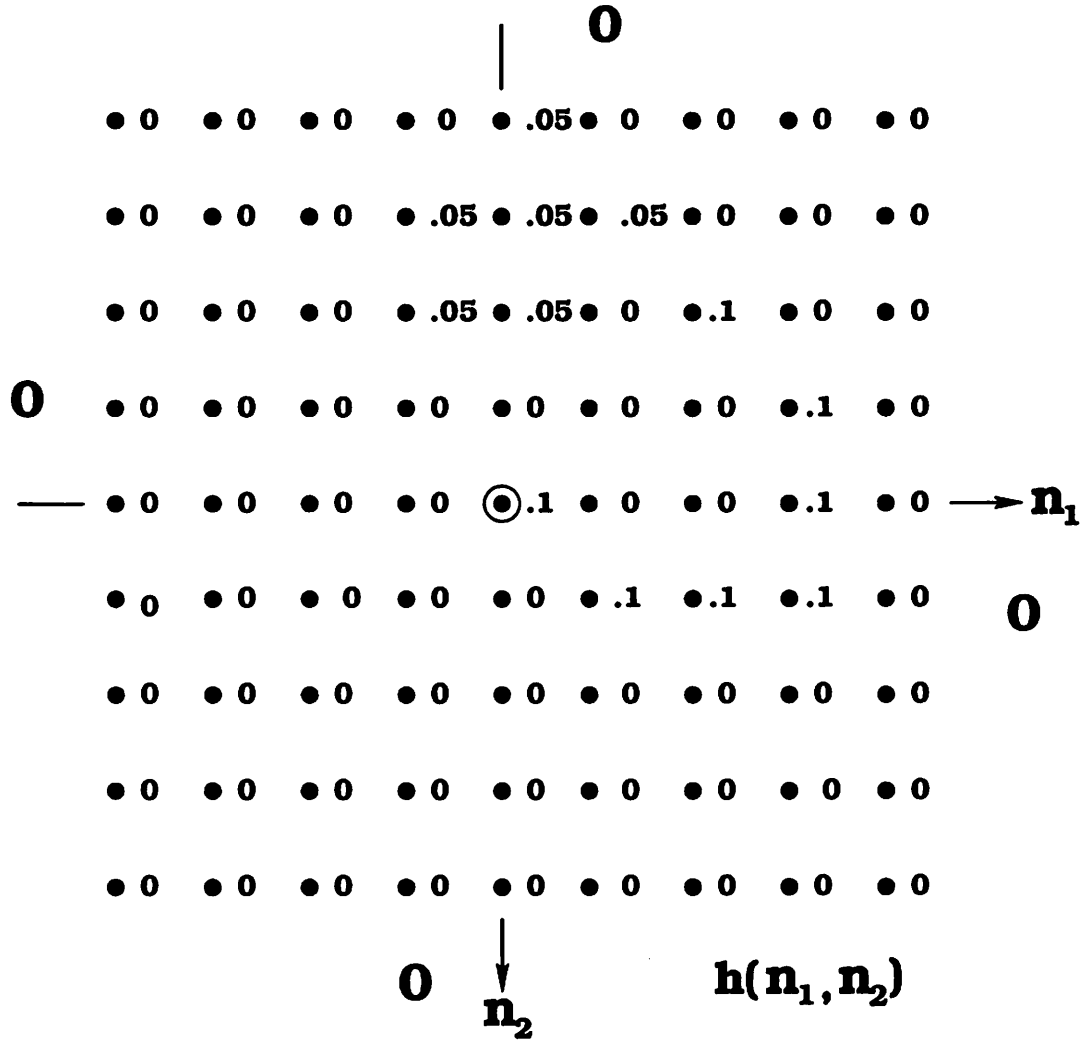

Figure 1: The example impulse response.

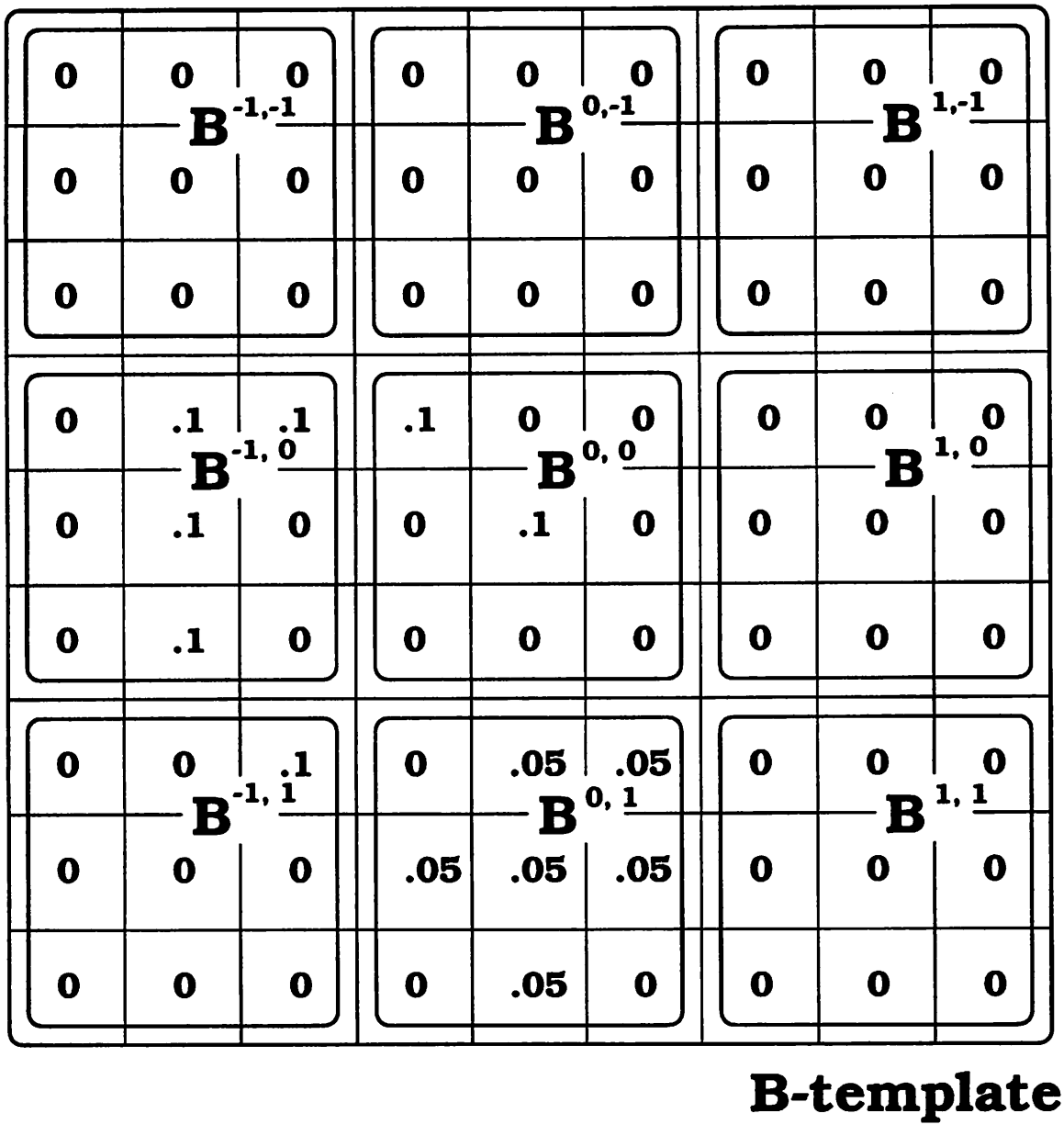

Figure 2: The equivalent B-template and the 3×3 templates used for implementation.