Morphological design of Discrete-Time Cellular Neural Networks
Brugge, Mark Harm ter

IMPORTANT NOTE: You are advised to consult the publisher's version (publisher's PDF) if you wish to cite from it. Please check the document version below.

Document Version
Publisher's PDF, also known as Version of record

Publication date:
2005

Link to publication in University of Groningen/UMCG research database

Citation for published version (APA):

Copyright
Other than for strictly personal use, it is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license (like Creative Commons).

Take-down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from the University of Groningen/UMCG research database (Pure): http://www.rug.nl/research/portal. For technical reasons the number of authors shown on this cover page is limited to 10 maximum.
Chapter 5 Template Decomposition

In this chapter the link established in the previous two chapters between Mathematical Morphology and DT–CNNs is used to derive a new template decomposition method. The method is based on the decomposition of morphological structuring elements. From a practical point of view, template decomposition is an important issue since image processing operations can be non-local, while the cells in physical DT–CNN realizations often communicate only with cells in their direct neighborhood. The method established in this chapter turns out to be far more efficient in terms of the number of templates in the decomposition than the traditional decomposition method.

This chapter is based on [11][12][18].

5.1 Introduction

The output of a cell $c$ in a DT–CNN is determined by the input and output of all other cells in the neighborhood of $c$. For a lot of simple operations like image inversion, edge extraction or isolated pixel removal, a $3 \times 3$ neighborhood is sufficient. For other operations like template matching the output of a cell is determined by the input and output of cells in a larger neighborhood. Theoretically, this is no problem since the definition of a DT–CNN allows neighborhoods of any size. Practically, neighborhood sizes are often limited to $3 \times 3$, or even worse, network cells are sometimes only connected to their 4–neighbors, which means that templates should have the following form:

$$T = < \begin{vmatrix} 0 & a_1 & 0 \\ a_2 & a_3 & a_4 \\ 0 & a_5 & 0 \end{vmatrix}, \begin{vmatrix} 0 & b_1 & 0 \\ b_2 & b_3 & b_4 \\ 0 & b_5 & 0 \end{vmatrix} >,$$  \hspace{1cm} (5-1)

The restrictions on the shape and size of the templates are imposed by the fact that heavily interconnected networks are extremely hard to realize in current VLSI technology. So on the one hand we would like to define operations in which cells can communicate with cells outside the direct–neighborhood ($3 \times 3$ neighborhood) and on the other hand the limited interconnectivity imposed
by VLSI technology demands that inter–cell communication may only be local. The solution to this problem is to obtain non–local behavior through iteration. Suppose we have a system in which cells can only communicate with cells in the direct neighborhood. One evaluation cycle in such a system propagates information from one cell to all cells in its $3 \times 3$ neighborhood. In the next evaluation cycle each of these cells propagates (part of) this information to all cells in their $3 \times 3$ neighborhood, which implies that in two steps cells can influence or may be influenced by cells in the $5 \times 5$ neighborhood. Examples of systems where information is propagated outside the direct neighborhood by iterative direct–neighborhood communication are hole filling and shadow creation (see Section 2.3).

For hole filling and shadow creation the decomposition of the large–neighborhood function into a sequence of direct–neighborhood functions is relatively simple because

- the decomposition can be obtained by logical reasoning on the functional level
- the decomposition is an iteration of identical operations

For most large neighborhood functions, however, these conditions do not hold and the task of function decomposition is much more complicated.

The remainder of this chapter is organized as follows. In Section 5.2 we will describe the traditional template decomposition method. Section 5.3 introduces the principles of morphological decomposition and overviews the research that has been done on this topic. In Section 5.4 two kind of morphological decomposition strategies are described with the purpose of making a worst–case analysis of the number of templates in a morphological decomposition and comparing this with the traditional decomposition. In Section 5.5 a more efficient morphological decomposition method is given that—under certain conditions—even is optimal. The chapter ends with some concluding remarks in Section 5.6.

## 5.2 Traditional Decomposition

The traditional method for decomposing templates into a sequence of smaller templates is presented in [99]. The described method is generic i.e. it applies to templates with a arbitrary neighborhood size $r > 1$ and does not assume special conditions on the coefficients of the template. The method is based on a recursive decomposition of the template. A template with a neighborhood size of $r$ is decomposed into a small number of templates with a neighborhood size of $r - 1$. This procedure is repeated until the neighborhood size of the templates in the decomposition is small enough.

For the sake of simplicity the theory is described for a DT–CNN with zero control template and cell bias. The control template $A$ is an arbitrary template with a neighborhood size of $r > 1$. Recall from Chapter 2 that the output equation for such a system is given by:

$$y(k) = F(A \odot y(k - 1)) \quad (5-2)$$

In Chapter 2, the transfer function $F$ is the hardlimiter transfer function (Formula (2–4)) but for the described theory the author had to use a semi–linear transfer function:

$$F(x) = \begin{cases} 
-1 & \text{for } x \leq -1 \\
 x & \text{for } -1 < x < +1 \\
+1 & \text{for } x \geq +1 
\end{cases}$$
First it is shown that given the control template \( \mathcal{A} \), templates \( \mathcal{A}_1, \ldots, \mathcal{A}_5 \) with a neighborhood size of \( r - 1 \) can be found such that \( \mathcal{A} \) can be rewritten as:

\[
\mathcal{A} = \mathcal{A}_1 \odot \mathcal{A}_2 + \mathcal{A}_3 \odot \mathcal{A}_4 + \mathcal{A}_5
\] (5-3)

In this decomposition, templates \( \mathcal{A}_1, \ldots, \mathcal{A}_4 \) are responsible for generating the \( 8 \times r \) boundary coefficients of \( \mathcal{A} \), while template \( \mathcal{A}_5 \) compensates for the difference in the non-boundary coefficients generated by the composition of \( \mathcal{A}_1, \ldots, \mathcal{A}_4 \) and the non-boundary coefficients of template \( \mathcal{A} \). Equation (5-3) implies that the neuron output equation can be rewritten as:

\[
y(k) = \hat{\mathcal{A}}_1 \odot (\mathcal{A}_2 \odot y(k - 1)) + \hat{\mathcal{A}}_3 \odot (\mathcal{A}_4 \odot y(k - 1)) + \mathcal{A}_5 \odot y^c(k - 1)
\] (5-4)

Where \( \hat{\mathcal{A}}_1 \) and \( \hat{\mathcal{A}}_3 \) are templates which are related to \( \mathcal{A}_1 \) and \( \mathcal{A}_3 \) in the following manner:

\[
\begin{align*}
\hat{\mathcal{A}}_1(i,j) &= \mathcal{A}_1(-i,-j) \\
\hat{\mathcal{A}}_3(i,j) &= \mathcal{A}_3(-i,-j)
\end{align*}
\]

By scaling the template coefficients to guarantee operation in the linear part of the semi-linear transfer function, Equation (5-4) can be computed by a DT-CNN system with 5 layers, each having a template with a neighborhood size of \( r - 1 \).

Though the author does not mention it explicitly, the proposed decomposition method leads to an exponentially growing number of templates. This is proved as follows. Let \( P(r) \) denote the number of \( 3 \times 3 \) templates in the decomposition of a template with radius \( r \). It is easy to see that this function is defined by:

\[
P(r) = \begin{cases} 
1, & \text{for } r = 1 \\
5P(r - 1), & \text{for } r > 1
\end{cases}
\]

For \( r > 1 \), the closed form for \( P(r) \) is given by:

\[
P(r) = 5^{r-1}
\] (5-5)

which shows that \( P(r) \) is exponential in \( r \). As an example, the decomposition of a \( 9 \times 9 \) template results in 125 direct-neighborhood templates, while a \( 21 \times 21 \) template is decomposed into 1,953, 125 direct-neighborhood templates, which is unacceptable for practical purposes.

### 5.3 Morphological Decomposition

Recall from Chapter 3 that morphological functions are constructed from a small number of primitive operators; the traditional set operators and the erosion and dilation operators (and the translation operator, but this is a special case of dilation and can therefore be left out of the discussion). The traditional set operators are zero-neighborhood functions. For example, in the union of two images, the value of a pixel at position \( c \) is completely determined by the value of the pixels at position \( c \) in the two argument images. This can also be concluded from Table 3.2, which shows that all template coefficients outside the \( 1 \times 1 \) neighborhood are zero. The dilation and erosion operators are functions that operate on a larger neighborhood. For these operators the size of the neighborhood is determined by the size of the SE by which they are parameterized. This can also be concluded from Table 3.3 and Table 4.2, which show that the size of a templates that realize
the morphological dilation and erosion operator (possibly combined with a traditional set operator) is equivalent to the size of the SE.

So the large-neighborhood behavior of a morphological function is given by the size of the SEs that parameterize the erosion and dilation operators in the defining expression. Since each erosion and dilation operator (possibly combined with a traditional set operator) is implemented in the DT-CNN domain by a template whose size is identical to the size of the SE, reducing the size of the SEs in a morphological expression will reduce the size of the templates in the realizing DT-CNN system.

It is easy to see that morphological algebra can be used to reduce the size of the SEs that parameterize dilation and erosion. Let us concentrate on the dilation operator first. Recall from Table 3.1 that the dilation operator has the following associative and distributive properties:

\[
X \oplus (A \oplus B) = (X \oplus A) \oplus B \\
X \oplus (A \cup B) = (X \oplus A) \cup (X \oplus B)
\] (5-6)

These rules imply that if a SE \(A\) can be decomposed into smaller SEs \(A_i\) using only set union and dilation, then the dilation of an image \(X\) by \(A\) can be rewritten as a union of sequences of dilations of \(X\) by SEs \(A_i\). For example, suppose we want to dilate an image \(X\) by the SE \(A\) and suppose that \(A\) has the following decomposition:

\[A = A_1 \oplus A_2 \cup A_3 \oplus A_4\]

Then the dilation of \(X\) by \(A\) can be rewritten as:

\[X \oplus A = X \oplus (A_1 \oplus A_2 \cup A_3 \oplus A_4) = (X \oplus A_1) \oplus A_2 \cup (X \oplus A_3) \oplus A_4\]

Similar to dilation, erosion with a large SE can be rewritten as an number of successive intersections and erosions with smaller SEs. This follows directly from the associative and distributive properties of the erosion operator:

\[
X \ominus (A \ominus B) = (X \ominus A) \ominus B \\
X \ominus (A \cup B) = (X \ominus A) \cap (X \ominus B)
\] (5-7)

The decomposition strategies that have been developed over the years differ in the type of SEs that can be decomposed (convex, concave, etc.), the type of decomposition (additive: using only dilations, hybrid: using dilations and unions), the SEs that are allowed in the decomposition (3 \(\times\) 3 SEs, 4-connected SEs, etc.), the cost criterion that is used for comparing the quality of different solutions (number of 3 \(\times\) 3 SEs, the accumulated city-block distance of the elements in the SEs), quality of the solution (optimal or sub-optimal) and how the solution is obtained (directly, using genetic algorithms, combinatorial search). In the following, we will give an overview of the most important articles.

The problem of binary structuring element decomposition was first addressed in [111]. In that paper a method is described to determine a decomposition into two-point sets (SEs with two pixels). The set of SEs that can be decomposed this way is a subset of the set of convex SEs. The proposed method produces an additive decomposition with a minimum number of two-point sets. The solution is found by combinatorial search. In [107] two decomposition strategies are presented. The first additively decomposes a convex SE into a minimum number of 3 \(\times\) 3 SEs. The second is similar to the first but gives a decomposition into a minimum number of subsets.
of the $3 \times 3$ rhombus (see Fig. 3.5.b). Both methods consist of giving an initial decomposition in so-called $3 \times 3$ prime factors, which is obtained in $O(1)$ time. This solution is then transformed to an optimal solution in linear time (with respect to the size of the SE that is to be decomposed) by iteratively replacing two or more $3 \times 3$ SEs by a single one. Like the second method given in [107], [87] also describes an optimal decomposition into subsets of the $3 \times 3$ rhombus. The cost criterion used in the latter paper, however, is not the number of SEs but the total number of 4-connected shifts required by the SEs in the decomposition, which is suitable for the implementation on 4-connected parallel array processors. The decomposition method given [88] is an extension of [107] and can be applied not only to convex SEs but also to a set of concave SEs that satisfy certain conditions. The price that has to be paid for this increase in applicability is that the optimal solution must be found by combinatorial search. In practice this is not really a problem since a solution can be easily tested for optimality and the search can be aborted once an optimal solution is found. In [7] and [8] genetic algorithms are used to find hybrid decompositions. The genetic approach applies to arbitrarily shaped SEs and the cost function can be adapted easily. The price to be paid for this generality is that an optimal solution is not guaranteed to be found and the search procedure is iterative; in each iteration the quality of a population of solutions is improved by mutation and combination. Finally, [89] also gives a method for finding a hybrid decomposition for arbitrarily shaped SEs. The method gives sub-optimal solutions in terms of the number of $3 \times 3$ SEs. The method efficiently decomposes the SE into a union of SEs that can be decomposed further using the method described in [88]. Clearly, the solution obtained by this method is found by combinatorial search.

Summarizing, both dilation and erosion with a SE $A$ can be decomposed into a number of dilations and erosions with smaller SEs $A_i$, if the SE $A$ can be decomposed into $A_i$ using set union and dilation. The problem of finding the SEs $A_i$ is known by the name of binary structuring element decomposition and is an extremely interesting problem that has been extensively investigated up until today by many researchers.

In the remainder of this chapter we will focus on decompositions into so-called neighborhood subsets ($3 \times 3$ SEs) since the allowed templates in the DT–CNN realization may be $3 \times 3$ template. Notice that if we would restrict ourselves to DT–CNN templates of the form given in Formula (5–1), the SEs in the decomposition should be subsets of the $3 \times 3$ rhombus. The cost–function that we use is the number of neighborhood subsets in the decomposition, since this number is equivalent to the number of $3 \times 3$ templates in the DT–CNN realization (the number that we want to minimize). From the literature overview given above we conclude that a decomposition into a minimum number of neighborhood subsets can be found for specific type of SEs but an optimal decomposition method for arbitrarily shaped SEs is yet to be found.

### 5.4 Upper–bound of the Morphological Approach

In this section we will present two SE decomposition strategies; the Simple Shifted Tile (SST) decomposition and the Advanced Shifted Tile (AST) decomposition, which is an improved version of the SST decomposition. Both strategies are decomposition strategies that can be used to decompose arbitrarily shaped SEs. The strategies are hybrid, which means that they produce a decomposition in terms of dilation and set union. The reason for presenting these strategies is twofold:

- to show that every SE is decomposable into $3 \times 3$ SEs
to obtain an upper-bound for the number of $3 \times 3$ SEs in a decomposition, so that the morphological approach can be compared with the approach given in Section 5.2

In the SST decomposition, a large-neighborhood SE is decomposed into a union of shifted tiles (neighborhood subsets), as illustrated by the following example. Assume a SE $A$ defined as in Fig. 5.1.

![Figure 5.1: An example of a large-neighborhood $9 \times 9$ SE.](image)

A tile $A_{x,y}$ of a SE $A$ is a copy of the $3 \times 3$ sub-image of $A$ that is centered around $(3x, 3y)$. Clearly our example SE consists of the 9 tiles shown in Fig. 5.2.a. Notice that the center of a tile is in the origin. In order to describe $A$ as a union of its tiles, each tile $A_{x,y}$ needs to be shifted to its original location $(3x, 3y)$ first. To this purpose we use 8 different SEs; one for each direction (see Fig. 5.2.b).

![Figure 5.2: Basic building blocks for the SE given in Fig. 5.1; (a) the sub-tiles and (b) the shifters.](image)

A dilation with such an element means a shift of 1 position in the corresponding direction. For example, dilating an image by $S_r$ means shifting it right one pixel. It is easy to see that $X \oplus A$ can be rewritten as:
where we adopt the notational convention that $X \oplus nA$ denotes a sequence of $n$ successive dilations of image $X$ by the structuring element $A$. Using Table 4.2 we can see that the dilation operator (possibly combined with a union) can be computed by a single DT–CNN layer. This leads to a total of 33 templates of size $3 \times 3$ (one template for each SE in the decomposition).

It turns out that the number of SEs in the SST decomposition is quadratic in the size of the SE. This is demonstrated as follows. Let $Q(r)$ denote the number of SEs in the SST decomposition of a SE of neighborhood $r$. For the sake of simplicity we will only define $Q(r)$ for values $r \geq 1$ for which $(r - 1)/3$ is an integer. First observe that for $r \geq 4$ a radius–$r$ SE contains $8(r - 1)/3$ more tiles (the tiles on the border) than a SE of radius $r - 3$. For each of these tiles $r$ SEs are required; the border tile itself and $r - 1$ SEs for shifting the tile to the appropriate location. Therefore, $Q(r)$ is defined by:

$$Q(r) = \begin{cases} 
1, & \text{for } r = 1 \\
Q(r - 3) + \frac{8}{3}r(r - 1), & \text{for } r \geq 4
\end{cases} \quad (5-9)$$

By induction it can be proved that for $r \geq 4$ the closed form for $Q(r)$ is defined by:

$$Q(r) = \frac{8}{27}r^3 + \frac{8}{9}r^2 - \frac{5}{27} \quad (5-10)$$

Note that the described decomposition is definitely not the most efficient one. It turns out that for $r \geq 7$ several shift operations can be combined using the distributivity of $\cup$ over $\oplus$. For example, the expression:

$$X \oplus A_{1,1} \oplus 3S_{dr} \cup X \oplus A_{2,1} \oplus 3S_r \oplus 3S_{dr} \quad (5-11)$$

which occurs in the SST decomposition of a radius–7 SE can be rewritten as:

$$(X \oplus A_{1,1} \cup X \oplus A_{2,1} \oplus 3S_r) \oplus 3S_{dr} \quad (5-12)$$

This implies that only 4 additional SEs are required for the border tile $A_{2,1}$, while the SST decomposition uses 7 SEs. Here the effort to shift the tile $A_{1,1}$ to its position is reused to shift the tile $A_{2,1}$ to its position. The method that starts with the decomposition in terms of tiles $A_{i,j}$ $|i|, |j| \leq 1$ (Formula (5-8)) and recursively adds a border of tiles where the shifting effort of the old border tiles is reused to shift each of the new border tiles to its position is called the Advanced Shifted Tile (AST) decomposition. Again, the number of SEs in the AST decomposition, say $R(r)$, is de-
rived for values \( r \geq 1 \) for which \( (r - 1)/3 \) is an integer. For each of the \( 8(r - 1)/3 \) border tiles 4 additional SEs are required, which gives the following recursive definition of \( R(r) \):

\[
R(r) = \begin{cases} 
1, & \text{for } r = 1 \\
33, & \text{for } r = 4 \\
R(r - 3) + \frac{32}{3}(r - 1), & \text{for } r \geq 7 
\end{cases}
\]  

(5–13)

and its closed form for \( r \geq 7 \):

\[
R(r) = \frac{16}{9}r^2 + \frac{16}{9}r - \frac{23}{9}
\]  

(5–14)

This shows that the number of SEs, and thus the number of DTCNN templates, is quadratic in \( r \). Instead of the 1,953,125 direct–neighborhood templates in the traditional decomposition of a \( 21 \times 21 \) template (see Section 5.2), now only 193 are required.

### 5.5 Efficient and Optimal Decompositions

The quadratic upper–bound given in the previous section is based on a AST decomposition method. Notice that the AST method is shape–independent, which means that the contents of the SE to be decomposed is not used in the determination of the decomposition. Clearly, the advantage of this shape–independent approach is that the method applies to arbitrarily shaped SEs. However, literature on the decomposition of binary SEs (see Section 5.3) shows that the number of SEs in a shape–oriented decomposition is often much smaller than the number of SEs given by the quadratic upper–bound. In [7], for example, a radius–10 SE is decomposed into only 42 SEs, while the AST decomposition consists of 193 SEs.

In this section we present a decomposition method for decomposing arbitrarily shaped SEs into neighborhood subsets. The proposed method will generally give much better solutions (in terms of the number of SEs) than the AST decomposition. Up until now, a method for the optimal decomposition of arbitrary shaped structuring elements does not exist. Here we present a simple method for the decomposition of arbitrary shapes that produces efficient solutions. For a more sophisticated method that produces sub–optimal decompositions for four–connected structuring elements, the reader is referred to [89]. The reason for presenting the new method here is to give the DT–CNN designer a practical framework that can be used to efficiently decompose arbitrarily shaped SEs into \( 3 \times 3 \) SEs without much knowledge about morphology and the complex algorithms described in the papers [88][107] on which the algorithm in [89] is based. The method described in this section combines elements of published strategies and theories and is complemented with a number of useful look–up tables derived in this thesis.

The proposed algorithm consists of the following steps. Like in [89], the SE at hand is first decomposed into a number of smaller SEs that satisfy a certain property. We use “convexity” as this can easily be verified (far more easy than the property used in [89]), while furthermore convex subsets have relatively simple decompositions. After decomposition, each convex subset is handled using a method that is based on the principles described in [107]. First the convex SE is described using Freeman’s chain code representation [38]. This representation is then used to make a decomposition of the shape in terms of a number of so–called prime factors (\( 3 \times 3 \) SEs). Then
shifting SEs (like in Fig. 5.2.b) are added to guarantee that not only the shape but also the location of the SE is correct. This approach of first making a decomposition of the shape and then compensating for translation is found in almost every paper on template decomposition. Finally, a set of reduction rules is applied to this decomposition in order to reduce the number of SEs.

The proposed decomposition method is illustrated using the example SE given in Fig. 5.1. First the SE is decomposed into a number of (possibly intersecting) convex subsets. An image is called convex if and only if it is an intersection of a number of discrete half-planes each with an angle that is a multiple of 45 degrees. One of the many possible decompositions of our example SE into convex subsets is given in Fig. 5.3. The convex subsets do not have to be disjoint. For our particular choice of a decomposition, sets $D_2$ and $D_3$ have one pixel in common. The reason for adding this pixel to both subsets is to reduce the effort required to decompose sets $D_2$ and $D_3$. As we shall see below, sets that have an identical shape but which are located at a different location have identical intermediate decompositions.

![Figure 5.3: A Decomposition of the N-character into convex subsets.](image)

Next, the chain code representation is determined for each convex subset. A chain code is a code that describes the object by enumerating (in counterclockwise direction) the subsequent directions between consecutive pixels on the object boundary (see Fig. 5.4). Notice that an image can be represented by different chain codes depending on the starting point of the enumeration. For convex images the starting point can be chosen such that the chain code is an ascending sequence of directions. This enables a more compact representation for convex images by a 7-tuple that enumerates the multiplicity of each direction $0, \ldots, 7$ (in that particular order) and will be referred to as the chain code representation in the remainder of this dissertation. In the next chapter we will present a number of extremely interesting properties that allow dilations and factorizations of SEs to be expressed as algebraic manipulations on the corresponding chain codes.

Notice that the chain code is a translation-invariant shape representation. Two images that have an identical shape but which are located at a different position are represented by the same chain code. Therefore an image is completely defined by its chain code complemented with its location. The location is usually represented by defining the image’s maximum $x$- and $y$-coordinate. For a nonempty image $A$ these values are defined by:
\[ \begin{align*}
X_{\text{max}}(A) &= \max \{x \mid (x, y) \in A\} \\
Y_{\text{max}}(A) &= \max \{y \mid (x, y) \in A\}
\end{align*} \] (5–15)

Figure 5.4: Freeman's chain code representation for arbitrarily shaped connected images and the compact representation for convex images. For both images \( X_{\text{max}} = 3 \) and \( Y_{\text{max}} = 1 \).

The chain codes of sets \( D_1, \ldots, D_4 \) are given by:

\[ D_1, D_4 = (1, 0, 8, 0, 1, 0, 8, 0), \quad D_2, D_3 = (0, 0, 1, 2, 0, 0, 1, 2) \]

The location of each of these sets is defined by:

\[ \begin{align*}
X_{\text{max}}(D_1) &= -3, \quad Y_{\text{max}}(D_1) = 4 \\
X_{\text{max}}(D_2) &= 0, \quad Y_{\text{max}}(D_2) = 0 \\
X_{\text{max}}(D_3) &= 2, \quad Y_{\text{max}}(D_3) = 3 \\
X_{\text{max}}(D_4) &= 4, \quad Y_{\text{max}}(D_4) = 4
\end{align*} \]

Based on the chain code we make a shape decomposition in terms of the \( 3 \times 3 \) prime factors shown in Fig. 5.5. This particular set of prime factors is generally known as the minimum set of convex \( 3 \times 3 \) sets that can be used to construct (using dilations) any convex image. The transformation from chain code to prime factor decomposition is described in Appendix A.3. Sets \( D_1 \) and \( D_4 \) are convex subsets of type 1 (see Appendix A.3) and have the following prime factor decomposition:

\[ D'_1 = D'_4 = Q_{24} \oplus 8Q_{144} \]

where the apostrophe above the sets indicate that the given decomposition is a decomposition of the shape (ignoring location information) rather than a decomposition of the template itself. Using the notational convention introduced in [87], we have:

\[ D'_1 \sim D_1 \] (5–16)
which denotes that image $D_1$ is identical to image $D'_1$ except for translation. The sets $D_2$ and $D_3$ are also convex subsets of type 1. The prime factor decomposition of these sets is given by:

$$D'_2 = D'_3 = \mathcal{Q}_{144} \oplus 2\mathcal{Q}_{136}$$

$\mathcal{Q}_{24}$ $\mathcal{Q}_{144}$ $\mathcal{Q}_{80}$ $\mathcal{Q}_{136}$ $\mathcal{Q}_{208}$ $\mathcal{Q}_{88}$ $\mathcal{Q}_{200}$

$\mathcal{Q}_{152}$ $\mathcal{Q}_{184}$ $\mathcal{Q}_{89}$ $\mathcal{Q}_{154}$ $\mathcal{Q}_{464}$ $\mathcal{Q}_{186}$

Figure 5.5: The convex prime factors.

Next, the decomposition of the image $D_i$ can be obtained from $D'_i$ by adding zero or more shifting SEs from the set $\{Q_2, Q_8, Q_{32}, Q_{128}\}$. The required number of shifting SEs for each type depends on the difference in location of $D_i$ and $D'_i$ and are determined by:

$$\#Q_2 = \max[0, Y_{\text{max}}(D'_i) - Y_{\text{max}}(D_i)]$$
$$\#Q_8 = \max[0, X_{\text{max}}(D'_i) - X_{\text{max}}(D_i)]$$
$$\#Q_{32} = \max[0, X_{\text{max}}(D_i) - X_{\text{max}}(D'_i)]$$
$$\#Q_{128} = \max[0, Y_{\text{max}}(D_i) - Y_{\text{max}}(D'_i)]$$

where the values $X_{\text{max}}(D'_i)$ and $Y_{\text{max}}(D'_i)$ can be evaluated using Appendix A.2 (that lists the $X_{\text{max}}$ and $Y_{\text{max}}$ value for each convex SE) and the following properties that hold for arbitrary nonempty images $A$ and $B$ (Proposition 3.6 in [87]):

$$X_{\text{max}}(A \oplus B) = X_{\text{max}}(A) + X_{\text{max}}(B)$$
$$Y_{\text{max}}(A \oplus B) = Y_{\text{max}}(A) + Y_{\text{max}}(B)$$

(5-17)

For example, since $X_{\text{max}}(D_1) = -3$ and $X_{\text{max}}(D'_1) = 1 \ast 0 + 8 \ast 0 = 0$, three $Q_8$’s are required to realize the correct $x$–position. Similarly $Y_{\text{max}}(D_1) = 4$ and $Y_{\text{max}}(D'_1) = 1 \ast 0 + 8 \ast 1$ imply that four $Q_2$’s are required to realize the correct $y$–position. This results in the following decomposition for $D_1$:

$$D_1 = \mathcal{Q}_{24} \oplus 8\mathcal{Q}_{144} \oplus 3\mathcal{Q}_8 \oplus 4\mathcal{Q}_2$$

The decompositions for $D_2, \ldots, D_4$ are obtained similarly and are given by:

$$D_2 = \mathcal{Q}_{144} \oplus 2\mathcal{Q}_{136} \oplus 3\mathcal{Q}_2$$
$$D_3 = \mathcal{Q}_{144} \oplus 2\mathcal{Q}_{136} \oplus 2\mathcal{Q}_{32}$$
$$D_4 = \mathcal{Q}_{24} \oplus 8\mathcal{Q}_{144} \oplus 4\mathcal{Q}_{32} \oplus 4\mathcal{Q}_2$$
Now Appendix A.4 can be used to reduce the number of SEs in the expression. We will demonstrate this for the decomposition of $D_1$. According to Appendix A.4 we have $Q_{144} \oplus Q_8 = Q_{72}$ and $Q_{144} \oplus Q_2 = Q_{18}$. Applying these reduction rules as often as possible to the decomposition of $D_1$ gives:

$$D_1 = Q_{24} \oplus Q_{144} \oplus 3Q_{72} \oplus 4Q_{18}$$

Then the rules $Q_{144} \oplus Q_{18} = Q_{146}$ and $Q_{72} \oplus Q_{18} = Q_{73}$ are applied as often as possible giving:

$$D_1 = Q_{24} \oplus Q_{146} \oplus 3Q_{73}$$

Finally, the rule $Q_{24} \oplus Q_{146} = Q_{219}$ is applied to give:

$$D_1 = Q_{219} \oplus 3Q_{73}$$

Since Appendix A.4 contains no reduction rule for the pair $[Q_{219}, Q_{73}]$ nor for the pair $[Q_{73}, Q_{73}]$, this expression cannot be reduced further. The reductions for the sets $D_2, \ldots, D_4$ as given in Fig. 5.6 are obtained similarly.

Notice that the SEs and the number of SEs in a final expression (an expression that cannot be reduced further) depends on the sequence of reduction rules that is applied to the initial expression (see Appendix A.4 for an example). In order to check the quality of an obtained solution, the number of SEs can be compared to the number of SEs in an optimal solution. According to [107], the number of SEs in an optimal decomposition of an arbitrary convex image $A$ is given by:

$$N_{opt}(A) = \max \left\{ \left| X_{max}(A) \right|, \left| X_{min}(A) \right|, \left| Y_{max}(A) \right|, \left| Y_{min}(A) \right| \right\}$$  \hspace{1cm} (5-18)

where $X_{min}(A)$ and $Y_{min}(A)$ are defined (like Equation (5–15)) as the minimum $x$– and $y$–coordinate of $A$ respectively. In the following chapter we show how to find a sequence of reduction rules that guarantees an optimal reduction in linear time. For now, we can verify afterwards that the decompositions for $D_1, \ldots, D_4$ as given in Fig. 5.6 are all optimal decompositions using Equation (5–18). This verification is left as an exercise for the reader. For strategies that guarantee an optimal decomposition of the binary convex SE, the reader is referred to [107] and Chapter 6.

The decomposition of the N-character derived with the method described in this section consists of 13 direct–neighborhood templates (see realization below). As expected this is much less that the 33 SEs in the AST decomposition given in Equation (5–8).

To obtain the DT–CNN realization that determines the dilation of an image $X$ with with the N–character we need to decompose the function $F$ defined by:

$$F(X) = X \oplus Q_{219} \oplus 3Q_{73} \cup X \oplus Q_{18} \oplus 2Q_{17} \cup X \oplus Q_{144} \oplus 2Q_{272} \cup X \oplus Q_{438} \oplus 3Q_{292}$$

into DT–CNN primitives (see Section 3.7). For the sake of readability, we first define functions $F_1, \ldots, F_4$ as:

$$F_1(X) = X \oplus 3Q_{73}$$
$$F_2(X, Y) = X \oplus Q_{18} \oplus 2Q_{17} \cup Y \oplus Q_{219}$$
$$F_3(X, Y) = X \oplus Q_{144} \oplus 2Q_{272} \cup Y$$
$$F_4(X, Y) = X \oplus Q_{438} \oplus 3Q_{292} \cup Y$$
Figure 5.6: A content-based decomposition of the $9 \times 9$ N-character.

It’s easy to see the desired function $F$ is equivalent to:

$$F(X) = F_4(X, F_3(X, F_2(X, F_1(X))))$$

Each function $F_1, \ldots, F_4$ can be decomposed into the generic DT–CNN primitives $G_j, H_j$, and $I_{j,k}$ defined by:

$$G_j(U, V) = V \oplus Q_j$$
$$H_k(U, V) = V \oplus Q_j \cup U$$
$$I_{j,k}(U, V) = V \oplus Q_j \cup U \oplus Q_k$$

The decompositions of the sub-functions $F_1, \ldots, F_4$ are given by:

$$F_1(X) = G_{73}([\square, G_{73}([\square, G_{73}([\square, G_{73}([\square, X])])]))$$
$$F_2(X, Y) = I_{18, 219}(Y, G_{17}([\square, G_{17}([\square, X])]))$$
$$F_3(X, Y) = H_{144}(Y, G_{272}([\square, G_{272}([\square, X])]))$$
$$F_4(X, Y) = H_{438}(Y, G_{292}([\square, G_{292}([\square, G_{292}([\square, X])]))))$$

The last step is to determine the templates that realize the DT–CNN primitives in the decomposition of $F$. The templates for the primitives $G_j$ can be derived directly from Table 3.3. The templates for the primitives $H_j$ are determined using Table 4.2 and the parameter swapping rule (Section 4.4), which imply the following template coefficients:

$$\mathcal{A}_{-d} = 1 \quad ; \quad d \in A$$
$$\mathcal{B}_o = 1$$
$$i = \#Q_j \quad (5–19)$$

Finally, the template for the primitive $I_{j,k}$ is found in Table 4.2.
The layout of the system is shown in Fig. 5.7. As expected the number of layers in the DT–CNN realization is 14. The templates $T_1...T_{14}$ that implement the corresponding network layers $L_1...L_{14}$ are defined as follows. Network layers $L_1...L_3$ are realizations of the primitive $G_{73}$. Substituting template $Q_{73}$ in the generic realization of $G_j$ (Table 3.3) gives:

$$T_1 = T_2 = T_3 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, 2 >$$

Similarly, templates $T_4$ and $T_5$ are found by substituting template $Q_{17}$ in the generic realization of $G_j$ giving:

$$T_4 = T_5 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, 1 >$$

Template $T_6$ is found by substituting $Q_{18}$ and $Q_{219}$ in the generic realization of $I_{j,k}$ (Table 4.2):

$$T_6 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}, 7 >$$
The templates that realize function $F_3$ are given by:

$$T_7 = T_8 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, 1 >, \quad T_9 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, 2 >$$

where template $T_9$ is found by substituting $Q_{144}$ into Formula (5–19). Finally, the templates that realize function $F_4$ are:

$$T_{10} = T_{11} = T_{12} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, 2 >, \quad T_{13} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, 6 >$$

Fig. 5.8 shows some experimental results. An input image with only two pixels is presented to the system. As expected, two N-characters appear at the position of the pixels after evaluating all network layers.
5.6 Conclusions

In this chapter it has been shown that morphological decompositions of large-neighborhood functions yield solutions that are far more efficient than the ones found by decomposing these functions in the DT–CNN domain. Traditionally, a large-neighborhood template is decomposed into a number of direct-neighborhood templates that is exponential in the size of the template to be decomposed. Using the AST decomposition method described in this chapter, we have shown the number of direct-neighborhood templates in a morphology-based decomposition is quadratic in the size of the template to be decomposed. Finally, a shape-based morphological decomposition method is presented that finds decompositions with a much smaller number of contemplating templates than the derived quadratic upper bound and in case of convexity an optimal solution can even be found.