Urszula Stańczyk
Silesian University of Technology
Institute of Informatics

Synthesis of Boolean functions for image processing

In binary image processing every finite window operator can be expressed as a Boolean function depending on surrounding pixels ([2]). However, such functions are known only for relatively few operations, which means that for the rest of possible image transformations it is necessary to perform some synthesis procedure.

Finding the output value of a transformed pixel requires checking values of points contained in the processing window and depending on these values (and possibly distribution of those values) computing the result, that is deciding whether the output pixel should take value 0 or 1. In this situation pixels in the processing window can be directly interpreted as variables of the logic function and their binary values as assumed logic values.

The size and shape of the processing window depend on the goal of image transformation ([1]). For square processing window (that is most often used) the typical enumeration of variables is given by the figure 1 below. The origin is usually placed at the centre.

x_3	x_2	x_1
x_4	x	x_0
x_5	x_6	x_7

x_{14}	x_{13}	x_{12}	x_{11}	x_{10}
x_{15}	x_3	x_2	x_1	x_9
x_{16}	x_4	x	x_0	x_8
x_{17}	x_5	x_6	x_7	x_{23}
x_{18}	x_{19}	x_{20}	x_{21}	x_{22}

a) window 3 by 3, b) window 5 by 5.

Fig. 1. Enumeration of variables corresponding to a pixel neighbourhood

The first step of synthesis, in general being the statement describing what action the function should take, here becomes establishing the goal of image transformation along with the size and shape of a processing window. This input can assume some different forms: it can be defined by specifying known morphological transformation type and a structuring element, it can be given through the set of masks causing hits, or by natural language statements.

An example of the first form can be given by Hit-or-Miss transformation detecting left lower corners in contours of objects. Such transformation requires the composite structuring element $D = (B, C)$ with $B = \{(0,0),(0,1),(1,0)\}$ and $C = \{(0,-1),(-1,-1),(-1,0)\}$ respectively. In general a BF defining HMT takes the form $F_b(x_1,\ldots,x_N) = x_1 \cdot \ldots \cdot x_M \cdot \overline{x_{M+1}} \cdot \ldots \cdot \overline{x_N}$ for structuring elements $B = \{d_1,\ldots,d_M\}$ and $C = \{d_{M+1},\ldots,d_N\}$. Assuming presented enumeration of variables in the pixel neighbourhood, in this case the function becomes $F_b(x, x_0, x_2, x_4, x_5, x_6) = x\ x_0\ x_2\ \overline{x_4}\ \overline{x_5}\ \overline{x_6}$. If the function is considered as operating on 3 by 3 neighbourhood, then it is degenerate in x_1, x_3 and x_7 variables. If the truth table is specified for this function then minimising process results in obtaining the same logic expression as presented above. In such case omitting vacuous variables brings more compact representation.

In an example of the second form there will be used the concept of connectivity, which for discrete images is usually defined for rectangular grid as happening only in direct horizontal and vertical manner (4-connectivity) or allowing also diagonal neighbours (8-connectivity). This second type can cause problems with interpretation of pixels as either comprising an object that is 8-connected or comprising several disconnected objects that are 4-connected and how this relates to the connectivity of the background. To settle the matter there can be used an approach that for foreground pixels there is defined 4-connectivity, while for the background there is used 8-connectivity. In such case it may be needed to substitute any 8-connectivity that exists in ob-

jects of the considered image. Thus the transformation should check 4-connected neighbourhood and if there is any 8-connectivity case within, it has to be changed into 4-connectivity. The list of patterns that cause the output to equal 1 is given by the following.

$$\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0\,1\,0, & 0\,1\,0, & 0\,1\,1, & 0\,1\,1, & 1\,1\,0, & 1\,1\,0, & 1\,1\,1, & 1\,1\,1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0\,1\,0, & 0\,1\,0, & 0\,1\,1, & 0\,1\,1, & 1\,1\,0, & 1\,1\,0, & 1\,1\,1, & 1\,1\,1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
0\,0\,1, & 1\,0\,0, & 1\,0\,1, & 0\,0\,1, & 0\,0\,1, & 1\,0\,0, & 1\,0\,0, & 1\,0\,1, \quad 1\,0\,1. \\
1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \quad\; 1
\end{array}$$

Secification of patterns if considered in terms of variables and logic functions brings immediate recognition of on-set for the function. Thus the logic synthesis is already performed. However, it is always possible to yield some useful conclusions from analysis of given patterns. The exemplary function is always 1 whenever the central pixel equals 1 or when any number of following four situations happens: either (x_2 and x_4 are 1) or (x_2 and x_0 equal 1) or (x_4 and x_6 are 1) or (x_6 and x_0 equal 1). This observation gives $F_b(x, x_0, x_2, x_4, x_6) = x{+}x_0\ x_2{+}x_0\ x_6{+}x_2\ x_4{+}x_4\ x_6$, which also would be obtained by minimisation process.

Natural language statements describing image transformations constitute the most difficult case for logic synthesis because this form requires a designer's skill to turn such definition into some more precise form. As an example one can consider widely known median operators. For binary case such operator should set the output to the value of the majority of neighbouring pixels. For the window of 3 by 3 the number of pixels is 9 — so the output value should be equal to the state of 5 (or more) surrounding points. It means that if at least 5 out of 9 variables are 1 the output value equals 1 too, otherwise it is 0. The Boolean function that performs such operation written in the SoP minimal form would be equal to summed products of all possible combinations of fives of function variables. Due to Boolean algebra theorems, these terms cover the rest of terms with all possible combinations of 6, 7, 8 and 9 variables of the function. There are 126 combinations of fives of variables, so the function would have that number of terms.

In general case, however, the task specification can be much too complex to obtain the logic function in the simple straightforward manner described. In such situations there are two possible solutions: either generating all possible combinations of input variables and specifying corresponding to them output values, which is in fact the generation of the truth table; or producing a chosen representation type basing on several partial definitions of the transformation that usually are obtainable in much simpler way ([3]).

Since for the processing window of a given shape and containing N pixels there can be defined 2^{2^N} different functions, instead of attempts to synthesise one of them, it is possible to generate all, and then, basing on observations of results of image processing with such definitions of transformations, state what they do to an image. With this approach the process of logic synthesis becomes logic analysis.

References

[1] Robert M. Haralick, Stanley R. Sternberg, Xinhua Zhuang. *Image analysis using Mathematical Morphology.* IEEE Transactions on Pattern Analysis and Machine Intelligence 9, No. 4, July 1987.

[2] Henk J.A.M. Heijmans. *Mathematical Morphology: a modern approach in image processing based on algebra and geometry.* SIAM Review 37, No. 1, March 1995.

[3] William K. Pratt. *Digital Image Processing.* John Wiley & Sons, New York, 1991.