#2	General Framework for Iterative Modification	ECE 172a	Pamela Cosman	1/12/12

Let b_{ij} be the binary value of a pixel at location (i, j). Let A be a set of neighborhoods (surrounds). We define

$$a_{ij} = \begin{cases} 1 & \text{if the neighborhood of } (i,j) \in A \\ 0 & \text{otherwise} \end{cases}$$

The output pixel that is in location (i, j) is given a value c_{ij}, where c_{ij} is some Boolean function L of a_{ij} and b_{ij}. There are 16 different possible Boolean functions of two binary inputs:

ab	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
00	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
01	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
10	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
11	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Here each column represents a Boolean function while each row represents one of the possible combinations of values for the two inputs a and b. The value at the intersection of a particular row and a particular column is the output produced by the Boolean function when given the input shown on the left.

Some of these 16 functions are not very interesting. Number 0 always produces zeros as output, while Number 15 always produces ones. Numbers 5 and 10 simply reproduce b and its complement $\overline{b}$. So these are an identity operation and a complementing operation on the input image. Another two (numbers 3 and 12) reproduce a and $\overline{a}$. So this is a marking operation: the output pixel simply marks wherever the surrounds in A (or the surrounds not in A) are found.

More interesting are the logical *and* (Number 1) and the logical *or* (Number 7). We denote the *and* operation by $c = a \cdot b$ and note that it will be some kind of erosion or "etching away" operator, since it can only remove ones from the image: $a \cdot b \leq b$

The *or* operator, denoted $a + b$ will implement dilation or growing of objects, since it can only remove zeros from an image: $a + b \geq b$

Some of the remaining Boolean operations, such as $\overline{a} \cdot b$ and $\overline{a} + b$ are of little interest since the same effect can be achieved by using the complement of the set A and employing $a \cdot b$ or $a + b$ instead.

In addition to A and the function L, there remain two things to specify for a general iterative modification scheme:

- The **number of iterations,** n, says how many times we will apply the operator.
- The **number of subfields,** f specifies how many tesselations (tilings, subfields) the image is subdivided into.

We divide the image pixels into subfields, and operate on all pixels in a subfield in parallel, but consider one subfield sequentially after another one. If a subfield contains a pixel X then it should not contain the neighbors of X (whatever neighborhood is used for making the iterative modification). For example, we could use the following 4 subfields:

1	2	1	2	1	2
3	4	3	4	3	4
1	2	1	2	1	2
3	4	3	4	3	4