\[
\begin{array}{l}
\cdot op++ = \text{factor} \cdot c + 0.5; \\
\end{array}
\]
while (ip < iend);
}

else{

c = 0;
a = 0;
do{
if(a)

\[
c += a \cdot (\cdot ip);
\cdot op++ = c + 0.5;
\]

ns = factor - a;
for (j=0; j < ns; j++) \cdot op++ = \cdot ip;
b = \text{factor} - ns - \cdot a;
c = b \cdot (\cdot ip++);
a = 1. - b;
} while (ip < iend);

ip is a pointer to the input array, op is a pointer to the output array, a, b, c, area auxiliary variables, and ns the number of whole pixels mapped to the output.

T.D.S. Duff
T. Pavlidis
AT&T Bell Labs
Murray Hill, NJ


The author replies:

Thank you for your response. The algorithm was indeed developed intuitively and experimentally, and the minds that have worked on the algorithm since its development have been stuck in the realms of algebraic point transforms and projective geometry. The integral calculus didn’t occur to us.

It seems to be a simple, straightforward, and elegant characterization of the algorithm, but if I understand the integral correctly, it should be scaled by \( s \) to achieve the correct output value. The C code appears to be a correct implementation but it’s much more complex than necessary.

The strength and novelty of the algorithm stems from the fact that the loop can be implemented without expansion smoothing with two adders, one multiply, and one accumulate, and that it still generates nonalised pictures.

Karl M. Fant