LZ1 Compression of Binary Images using a Simple Rectangle Greedy Matching Technique

Luigi Cinque and Ernesto Grande

Computer Science Department, University of Rome, Italy
Sergio De Agostino

Computer Science Dept., AASU, Savannah, GA 31419, USA

Storer [1] suggested that very fast encoders are possible for lossless image compression by showing a square greedy matching LZ1 heuristic, which can be implemented by a simple hashing scheme and achieves 60 to 70 percent of the compression of JBIG1 on the CCITT test set. In this paper, we show a rectangle greedy matching LZ1 heuristic suitable for a similar fast implementation and achieving 75 to 90 percent of the compression of JBIG1 on the CCITT test set. The compression heuristic presented scans an image $n \times m$ row by row. Therefore if $(i, j)$ is the current position, the last 64K pixels read (the window) are in positions $(k, h)$ such that $k \leq i$. We implemented a simple procedure for finding the largest rectangle with left upper corner $(i, j)$ that matches a rectangle with left upper corner $(k, h)$. Obviously, this procedure can be used for computing the largest monochromatic rectangle in a given position $(i, j)$ as well. If the $4 \times 4$ subarray in position $(i, j)$ is monochromatic or is not hashed to a valid pointer in the window, then we compute the largest monochromatic rectangle in that position. Otherwise, we compute the largest rectangle match in the position provided by a hash table. If the coding does not compress, $(i, j)$ is left as a raw bit. The positions covered by matches are skipped in the linear scan of the image. In practical cases, twelve is the maximum number of bits needed to encode either the width or the length of a rectangle match but often four bits suffice. We use either four or eight or twelve bits to encode one rectangle side. Therefore, nine different kinds of rectangle are defined. A pointer is encoded in the following way: one bit indicating if a raw bit has been left in the current position; the raw bit or one bit indicating if the rectangle is monochromatic; one bit for the color of the rectangle or sixteen bits for the position of the match in the window; three or four bits encoding one of the nine kinds of rectangle; bits for the length; bits for the width. Larger rectangles are less frequent but still relevant for the compression performance. Then, four bits are used to indicate when twelve bits or eight and twelve bits are needed for the length and the width. This way of encoding rectangles plays a relevant role for the compression performance. In fact, it wastes four bits when twelve bits are required for the sides but saves four to twelve bits when four or eight bits suffice.

A major area of investigation that relates to the issue of fast encoding is massively parallel implementation. It is of theoretical and practical interest to consider sublinear algorithms for two-dimensional sliding window compression.

References