NUROP CONGRESS PAPER
IMAGE ENCODING BASED ON FINITE AUTOMATA
METHODS FOR REAL-TIME COMMUNICATIONS

YANG KAI¹ AND ONG GHIM HWEE²

School of Computing, National University of Singapore
3 Science Drive 2, Singapore 117543

ABSTRACT
Fractal-based image compression techniques give very efficient decoding time with fairly primitive hardware requirements, which favour real-time communication purpose. We study two such image encoding techniques, namely the generalized finite automata (GFA) for bi-level images and the weighted finite automata (WFA) for greyscale images. An improved image partitioning technique — the binary (or bin-tree) partitioning — is tested on both GFA and WFA encoding methods. The result of using binary partitioning clearly outperforms other traditional partitioning techniques such as the quad-tree. In the implementation of the WFA encoder, we propose a way to separate image components into zones of decreasing significance. This can be used to both decode the image progressively over a congested network, and to pave way for multi-layered error protection over an unreliable networking environment. We are optimistic that the overall performance of our WFA codec is superior to existing methods such as JPEG and GIF, and that it is well poised to suit real-time communication applications.

1 GENERALIZED FINITE AUTOMATA AND BI-LEVEL IMAGE COMPRESSION
In this section, we introduce the Generalized Finite Automata (GFA) as a tool for specification of bi-level images, and we describe an inference algorithm for the construction of an automaton.

A bi-level multi-resolution image is specified by assigning the value 0 (white) or 1 (black) to every node of the infinite quad-tree. If the outgoing edges of each node of the quad-tree are labelled 0, 1, 2, 3 we get a uniquely labelled path to every node; its label is called the address of the node. The address of a node at depth \( k \) is a string of length \( k \) over the alphabet \( \Sigma = \{0,1,2,3\} \). Hence, a bi-level multi-resolution image is specified as a subset of strings over the alphabet \( \Sigma \), namely the collection of the addresses of the nodes assigned value 1 (black).

A digitized image of resolution \( m \times n \) consists of \( m \times n \) pixels each of which takes a Boolean value for bi-level image, or a real value (practically quantized to an integer between 0 and 255) for a greyscale image. Here we will consider square images of the resolution \( 2^k \times 2^k \) \((7 \leq n \leq 11)\). In order to facilitate the application of finite automata to image description, we assign each pixel at \( 2^k \times 2^k \) resolution correspond to a sub-square of size \( 2^\alpha \) of the unit square. We choose \( e \) as the address of the whole unit square. A finite automaton is displayed by its diagram which is a directed

¹ Student — 3rd year B.Comp.(Hons) in Computer Engineering, School of Computing, National University of Singapore
² Supervisor — Ph.D., Associate Professor, School of Computing, National University of Singapore
graph whose nodes are the states, with the initial node indicated by an incoming arrow. An edge labelled from state $i$ to state $j$ indicates that the input causes the transition from state $i$ to state $j$. A word in the input alphabet is accepted by the automaton if it labels a path from the initial state to a final state. For example, the 8x8 chess-board as a multi-resolution image in Figure 1 is described by the regular set $\Sigma^2 \{1, 2\} \Sigma^*$ or by the automaton $A$. Please take note that the regular expression $\Sigma^2 \{1, 2\} \Sigma^*$ is the concatenation of two regular expressions $\Sigma^2$ and $\{1, 2\} \Sigma^*$.

![Figure 1. Finite automaton $A$ defining the 8x8 chess-board](image)

Now, we introduce Generalized Finite Automata (GFA) that specify bi-level images. Each transition of a GFA is labelled by an input symbol (similar to the finite automata in the previous section) and optionally also by one of the 16 transformations that can be obtained by: 90° rotation, flipping and complementation. In the diagram of GFA the transformations are specified by transitions $t_0$ to $t_{15}$, where $t_0$ being the identity transformation can usually be omitted. Their definition is explained in Figure 2 below.

![Figure 2. The 16 Generalized transitions of the GFA (numbered from $t_0$ to $t_{15}$)](image)

An algorithm to find the automaton $A$ that approximates a given bi-level image $\Psi$ is outlined:

```plaintext
>> Input: A bi-level image $\Psi$ addressed by a finitely labelled quadtree, maximum tolerance is error.

1. Assign initial state $q_0$ to the sub-square $e$ (the entire image), and set $q = q_0$.
2. Recursively divide the next unprocessed state $q$ into 4 sub-states $q_0$, $q_1$, $q_2$ and $q_3$. Denote the image in each of the sub-states $q_a$ where $a \in \Sigma$ by $\Psi'$.  
3. If $\Psi'=0$ there is no transition from $q$ with label $a$ (since this sub-state is completely white so we do not bother). Otherwise, the algorithm searches through all created states and all transformations $t_i$ where $i = 0, 1, \ldots, 15$. If state $p$ and transition $t_i$ is found such that $d_i(\Psi', t_i, [\Psi_p]) \leq error$ where $\Psi_p$ is a sub-square image assigned to the state $p$, we create a new edge $(q, a-j, p)$. If
```
there are no such state and transition, we assign a new (unprocessed) state \( r \) to \( \psi' \) and create a new edge \((q, a-0, r)\).

4. Go to step 2 if there is still another unprocessed state, or stop otherwise.

>> Output: GFA \( A \) that describes the bi-level image with maximum tolerance error.

To decode the GFA representation of an image, we just need to follow each \((q, a–i, p)\) pair where \( q \) is the current state, \( p \) is the next state zoomed into the quadrant \( a \in \Sigma \) and with transition \( i = 0, 1, \ldots, 15 \). If the current state is known, simply copy the block to the destination space in the decoded image. The decoding procedure is initiated by setting \( q = q_0 \) of \( \epsilon \) (the entire image).

2 Weighted Finite Automata and Greyscale Image Compression

Unlike GFA that process only bi-level images, the Weighted Finite Automaton (WFA) accepts greyscale pictures containing \( 2^m \times 2^m \) pixels where pixel intensity ranges from 0 to 255. Furthermore, a WFA specifying an image does not look for only one of the known states that best matches the sub-state under investigation (as does the GFA); but instead, uses a linear combination of (possibly all) known states to arrive at a much better approximation.

An \( m \)-state weighted finite automaton \( A \), over alphabet \( \Sigma \) is specified by:

1. A row vector \( I^A \in R^{\text{run}} \) (called the initial distribution),
2. A column vector \( F^A \in R^{\text{med}} \) (called the final distribution),
3. Weight matrices \( W^A_a \in R^{\text{med}} \) for all \( a \in \Sigma \).

Suppose that an image \( f \) has been subdivided into a set of non-overlapping homogeneous regions. Starting with the complete image, the current range image is recursively inspected if it can be approximated with a linear combination of arbitrary state images. If a linear combination yields only a poor approximation, then the range image will be subdivided and the recursion continues. A new state must then be appended to the WFA tree in order to establish an approximation of all the new range images. In addition, a new tree edge must be appended to the WFA in order to signify the new partitioning. On the other hand, if an approximation satisfies the given quality threshold, the corresponding transitions are appended to the WFA and the recursion terminates. An outline of the algorithm is given below:

```plaintext
buildWFA(range image \( \psi \))
{
    1 Search for constants \( c_1, K, c_k \), states \( q_i, K, q_k \) and sub-images \( o_i, K, o_k \) such that minimizes \(|\psi-(c_1\psi_{q_1}(o_1)+L+c_k\psi_{q_k}(o_k))|\);
    2 Remember the current number of WFA states by setting \( s = n \);
    3 If (range image \( \psi \) consists of two or more pixels)
        3.1 Subdivide the range image \( \psi \);
        3.2 Call buildWFA(\( \psi_i \)) where \( i \in \Sigma \);
    4 If (subdivision gives better approximation w.r.t. a cost function \( C() \))
        4.1 Set \( n = n + 1 \);
}
```
4.2 Add the $n^{th}$ state with corresponding transitions to the WFA tree;

4.3 Append the images $\psi_i$ for $i \in \Sigma$ to the domain pool;

Else

5.1 Remove states and images created in the recursive subdivision: $n = s$;

\}

One finer detail is worth looking at in the recursive procedure above. Sometimes a new subdivision is more appropriate than an approximation of the current range image. If we only use a threshold value to decide whether an approximation is sufficient, we cannot realize that there might exist a better approximation if this range image were subdivided. Therefore, the inference algorithm is required to compute both alternatives:

1. Subdivide a range image into either four quadrants or two halves, and encode each of them independently; and

2. Approximate a range image with a linear combination from the dictionary.

The results are compared and the better one is used. The following steps must be processed in order to compress a given image $f$:

1. The automaton is initialized with some initial basis;

2. The image $f$ is approximated with a call to buildWFA(image $f$);

3. Finally, the coefficients of the WFA are quantized and stored with an entropy coder.

We introduce a cost function $C()$ which is used to evaluate different approximations of the same range image. Let $d(\psi, \psi')$ be the distance between the original image $\psi$ and the approximation $\psi'$ — calculated in terms of the standard RMSE and let $s(T)$ be the number of bits needed for storing the part $T$ of the WFA that is responsible for the current approximation. Then, the cost function $C()$ is defined as the following: $C(\psi, \psi', T) = q \cdot d(\psi, \psi')^2 + s(T)$. Parameter $q$ is used to control the quality of the encoding process; large values of $q$ will produce better approximations while small values will produce higher compression ratio. Only the computation of the automaton size is more difficult. A sophisticated function has to be constructed such that $s(A_f)$ is close to the number of bits needed for storing the automaton $A_f$.

We implemented binary partitioning as an important enhancement to WFA encoding. We test both horizontal and vertical cutting alternatives and pick the better choice. This inevitably introduces another layer of back-tracking, but the small associated overhead is justified by the visible improvement we observed in compression ratio. An illustration using the Lena image is shown in Figure 3 on the following page.

3 CONCLUSION

For the purpose of real-time image retrieval, an asymmetrical codec is favoured whereby the decoding time is small and the hardware support is primitive. We have examined a number of existing solutions and found that fractal-based compressions are suitable for our application. We further experimented with two finite automaton-related techniques, namely the GFA and WFA codecs, and we developed software to gauge the performance of these two codecs. Binary partitioning, an important enhancement to existing fractal-based partition implementations, is
studies in detail and implemented on both codecs. We observed good performance of GFA on bi-level images and WFA on greyscale images in general. However, problems still exist; the most severe amongst which is the blocking artefacts on boundaries.

Figure 3. Binary Partitioning applied on the Lean image

4 Reference

