Ordering the Numeric Sequence of Image Pixels at Lossless Compression

V. Smirnov [1], A. Korobeynikov [2]

Dept. Software, Kalashnikov Izhevsk State Technical University, Izhevsk, Russian Federation
E-mail: [1] drolbs@gmail.com, [2] kav33@inbox.ru

Received: 9.11.2015

Abstract. The present paper describes approach to building the ordered sequence of image pixels at lossless compression. This approach comprises the methods of supplementary virtual pixels, cascade fragmentation, and code book. Virtual pixels are added to split image into randomly-sized fragments. The cascade fragmentation method makes it possible to limit the number of stored start values to one. The code book method allows to minimize the search of subimage optimal bypass, the search algorithm complexity attaining $O(N^2)$. The numeric parameters such as the number of bypass options and time required to define optimal bypass are given for 6×6 image.

Keywords: lossless image compression, optimal bypass, code book, cascade fragmentation

INTRODUCTION

Paper [1] discusses building the image pixels bypass path based by the way of minimizing the costs of differential coding of pixels values employing the Little's algorithm. One disadvantage of the method is the long time of building the optimal bypass: approximately 1 minute for 16×16 image [1]. This can be explained by combinatorial explosion of the number of possible bypasses caused by increased image size. Hence, obtaining the optimal bypass for large-sized images is not feasible.

The present paper considers splitting the image into numerous fragments of equally small size. Subsequently, the optimal bypass is found for each fragment. The resulting fragments bypasses are to be joined in order to get the bypass of the complete image.

To implement the proposed method the following problems are to be solved:

1. Building optimal bypass for randomly-sized images which might be not multiple of fragment size.
2. Finding a method of joining image fragments bypasses.
3. The task of building bypass path for image fragment pixels is to be divided into sub-tasks: compiling the code book of all possible bypasses and searching the optimal bypass within their scope.

VIRTUAL PIXELS

Virtual pixels: splitting the image sized $N \times M$ into fragments sized $n \times m$ gives two areas: one containing complete fragments and another made up of incomplete fragments (Fig. 1). The number of pixels in the incomplete fragments area is found by formula:

$$K_{\text{ип}} = \sum_{i=0}^{M-1} \left[N - (N/n) \cdot n \right] + \sum_{j=0}^{N-1-(N-(N/n)n)} \left[M - (M/m) \cdot m \right], \quad (1)$$

where «/» is the integer division operation.

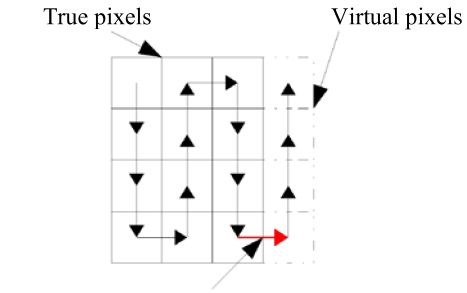

Figure 1. A fragment containing true and virtual pixels

In order to build image bypass in the incomplete fragments area it is required to add such number of virtual pixels to the image (fig. 1) that the following condition is satisfied:

$$\begin{cases} M \% m = 0 \\ N \% n = 0 \end{cases}, \quad (2)$$

where «%» stands for modulo operation.

Virtual pixels are not to be stored and represented in the compressed image, however they are used to build bypass of the fragmented image. Values attached to all virtual pixels are the same and they are many-fold the maximum values of true pixels.

Building the optimal bypass of a fragment consisting of virtual and true pixels is special in regard that only the true pixels values are taken into account:

$$min\left(\sum_{i \in V} \sum_{j \in V} c_{ij} \cdot x_{ij} \right) = min_{\partial} \left(\sum_{i \in V_{\partial}} \sum_{j \in V_{\partial}} c_{ij} \cdot x_{ij} \right) + \\ + min_{\text{м}} \left(\sum_{i \in V_{\text{м}}} \sum_{j \in V_{\text{м}}} c_{ij} \cdot x_{ij} \right) + c_{kl} \cdot x_{kl}, \quad (3)$$

where $x_{ij} \in \{0,1\}$ is binary variable representing the (i, j); $x_{ij} = 1$ edge if it belongs to the bypass and $x_{ij} = 1$ edge otherwise; i and j vertices are matrix pixels of image fragment; (i, j) edges are transitions between i and j *vertices*; $c_{ij} = |z_i - z_j|$ weights is the module of values difference for pixels joined by the edge (vertices), $V_{\text{м}}$ and V_{∂} are subsets of vertices of all virtual and true pixels correspondingly; $\min_{\text{м}}$ and $\min_{\partial}$ is the minimum sum of all virtual and true pixels correspondingly; $c_{kl} \cdot x_{kl}$ is the transition between true pixel k and virtual pixel l. Assuming that $c_{kl} \cdot x_{kl} = \infty$, the number of transitions between true and virtual pixels cannot exceed one and all virtual pixels have the same value.

Let us simplify the formula:

$$min\left(\sum_{i\in V}\sum_{j\in V} c_{ij}\cdot x_{ij}\right) = min_{\partial}\left(\sum_{i\in V_{\partial}}\sum_{j\in V_{\partial}} c_{ij}\cdot x_{ij}\right). \quad (4)$$

CASCADE FRAGMENTATION OF IMAGE

Having built the bypass of each fragment we next need to record the value of start pixel and the numeric sequence of bypass delta-code [1]. The number of image fragments (0 level) is found with the formula:

$$K_{\phi} = \left[(N/n) + ORD((N\%n) > 0)\right]\cdot\left[(M/m) + ORD((M\%m) > 0)\right], \quad (5)$$

where «ORD» is the operation of converting the Boolean number into {0, 1}.

The number of start elements is supposed to be equal to the number of fragments K_{Φ}. To avoid storing redundant data the matrix of start elements is generated (level 1). Thus obtained matrix is further split into fragments and optimal bypasses are found for them. The image cascade fragmentation (Fig. 2) goes on until the condition is met:

$$N\cdot M \le n\cdot m, \quad (6)$$

where N and M denote the size of the matrix, consisting of start vertices belonging to the previous fragmentation level.

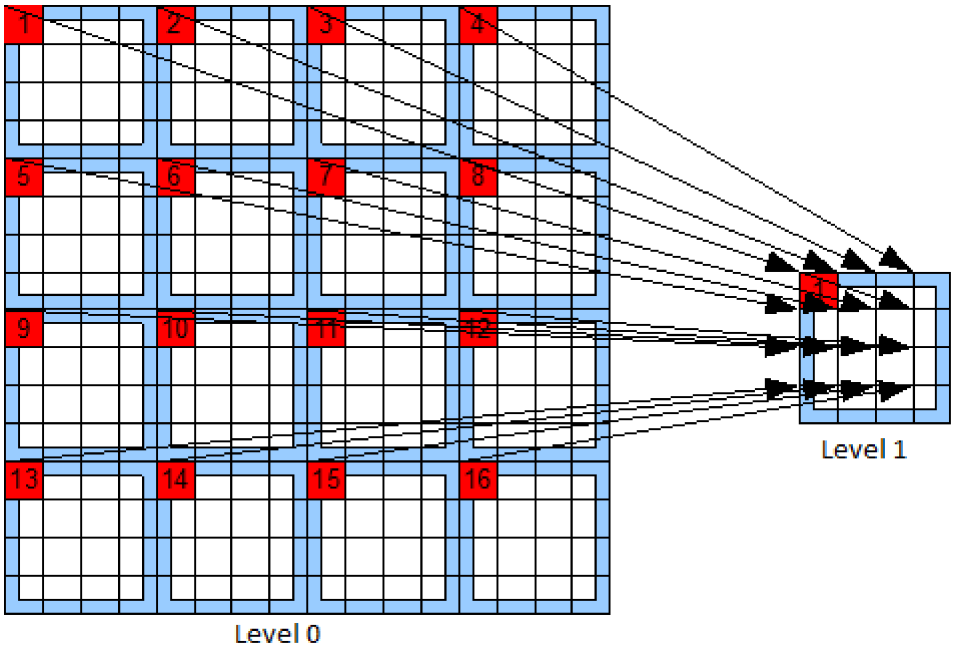

Figure 2. Cascade splitting of 16×16 image into 4×4 fragments

Recording the source image bypass starts with the last fragmentation level. For the last level matrix the value of its start vertex is recorded (the storage capacity of 1 byte is required for the range of pixels values z_i = {0, …, 255}), delta-code for its bypass [1] and also bypass path code. This information provided, the recorded matrix, i.e. the values of all its vertices, can be reconstructed. Thus, there is no need to store the values of fragments start vertices when recording the lower fragmentation levels.

Let us exemplify the recording of the image matrix bypass given in Fig. 2:

$$z_1; D^1{}_1; O^1{}_1; D^0{}_1; O^0{}_1; D^0{}_2; O^0{}_2; ...; D^0{}_{16}; O^0{}_{16};$$

where z_i is the value of start vertex; $D^s{}_i$ and $O^s{}_i$ are delta-code and bypass path code of fragment on s level with start vertex i correspondingly.

IMAGE FRAGMENT BYPASS

The novelty of the optimal image bypass method was checked against the sources [3–6]. The image fragment represented by the pixels values matrix $z_i \in (0,255)$ (Fig. 3) serves as the source data. Pixels matrix bypass can be any possible path beginning at the start cell and passing through all the cells just once. The task of searching optimal bypass is the special case of the Traveling Salesman problem. Similar to the Traveling Salesman problem the start vertex is given (upper left corner), although bypass is not limited to the start vertex return condition (Fig. 5). To solve the problem the weighted graph of the image is built (Fig. 4) [2]:

$$G(V,E) = \{i \in V, j \in V, (i,j), c_{ij} = \infty \mid (i,j) \notin E, c_{ij} \neq \infty \mid (i,j) \in E\}, \quad (7)$$

where i and j are matrix pixels; edges (i,j) are transitions between i and j vertices; weights $c_{ij} = |z_i - z_j|$ is module of values difference for pixels (vertices) joined by an edge. To reduce the calculating costs of the algorithm only the vertices corresponding to the matrix pixels and neighboring them either vertically or horizontally are joined by edges. For the rest (non-neighboring) vertices there are no edges on the graph and $c_{ij} = \infty$.

0	128	140
32	110	201
60	58	255

Figure 3. Sample pixels matrix for 3×3 image

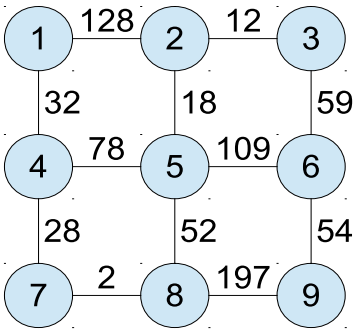

Figure 4. Sample image representation as weighted graph

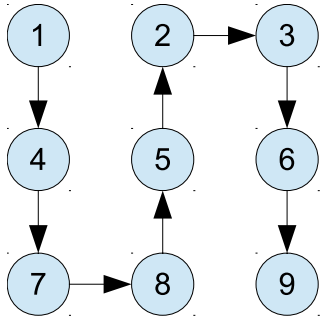

Figure 5. Sample bypass of image pixels

The graph bypass is limited by conditions (8) and (9):

$$|O| = |V| - 1 \quad (8)$$

$$\forall i \in V; \sum_{j \in V} x_{ij} \in \{1,2\} \quad (9)$$

where $x_{ij} \in \{0,1\}$ is the binary variable representing the (i,j); $x_{ij} = 1$ edge, provided the edge belongs to the bypass, and $x_{ij} = 0$ otherwise, $|O|$ stands for the length of the bypass.

The optimality of the bypass is evaluated as the minimum sum of bypass edges weights:

$$\min\left(\sum_{i \in V}\sum_{j \in V} c_{ij} \cdot x_{ij}\right) \quad (10)$$

Limitations for the graph bypass given in formulae (8) and (9) result in dead-end bypass paths failing to embrace all vertices. Such paths are not the sought ones, however they are considered at the stage of possible bypasses search.

The task should be divided into subtasks:

1. building all possible graph bypasses by conditions (8) and (9);
2. searching the optimal bypass according to criterion (10).

BYPASSES CODE BOOK

The subtask of building possible bypasses depends not on the edges weights c_{ij} but the structure and size of the graph solely. For a given graph all bypasses can be defined once and the obtained results can be stored later in a certain data structure such as the code book (bypass paths list). The use of the code book will allow to eliminate the task of searching all possible bypass options including the calculation of dead-end bypasses; this will save the time of optimal bypass search.

Bypass code book is the data structure having the form of oriented tree and storing the numbered bypasses of the given graph. The root node of the tree is the start vertex of the graph bypass. The internal nodes are the intermediate vertices of the bypassed graph. The number of siblings of non-terminal nodes varies from 1 to 3 and depends on the graph vertex which can have from 2 to 4 incident edges. The bypass is closed by the vertex represented as the terminal node in the tree. Data stored in the node has the following structure:

```
struct tree{
    int ind;
    tree *left, *middle, *right;
    long bRange, eRange;}
```

where *ind* is the number of bypass, *bRange*, *eRange* are numbers of all bypasses falling within *bRange* to *eRange* range and having common vertices up to the given node.

With help of the code book method the problem of the optimal bypass search is confined to evaluating the cost of bypass against all other bypasses stored in the book and subsequently choosing the least costly bypass.

EFFICIENCY OF THE APPLIED METHOD

An image patch has been tested representing a square matrix with pre-set size of 6×6. The time of searching optimal bypass is defined as the difference between two time measurements, one taken at search start, and another upon its completion. Branch and bound method was used for optimal bypass search. The PC in the experiment had the following configuartion: CPU T2080 1.73GHz, RAM DDR PC-3200 1024 Mb. For image sized 6×6 the code book contained 22144 various bypasses. Time of discovering the optimal bypass without the use of code book amounted to 626.766 s and with the pre-compiled book – 0.016 s. Algorithm complexity without the use of code book comes to $O(2^N)$ and with the book – $O(N^2)$.

CONCLUSION

To eliminate the bypasses number combinatory explosion problem the optimal bypass has to be built for the fragments of finite size.

Adding virtual pixels to the image allows splitting the randomly-sized image into fragments of equal size.

To join image fragments bypasses the cascade splitting of image into fragments of uniform size has been offered.

The proposed method of optimal image bypass search applying the code bypass book reduces the search time considerably.

Efficiency tests for the method revealed that compilation of the book represents the most time-consuming task at this.

The time of calculating the optimal bypass can be shortened by using a more efficient method in place of all bypass options search: branch-and-bound method, dynamic programming or parallel computation based on *OpenCL* or *CUDA* technologies.

REFERENCES

1. V.S. Smirnov, A.V. Korobeynikov, "The application of the Little's algorithm for minimizing the costs of coding image bypass at lossless compression," in Information systems in industry and education: Young researchers' collected papers, (Izhevsk, 2010), pp. 165–171.
2. F.A. Novikov, Discrete mathematics for programmers. Students' manual, SPb.: Piter, 2005.
3. D. Vatolin, A. Ratushnyak, M. Smirnov, V. Yukin, Methods of data compression. Archivers make-up, image and video, Moscow: Dialogue-MIFI, 2003.
4. W.B. Pennebaker and J.L. Mitchell, JPEG: Still Image Data Compression Standard, New York : Van Nostrand Reinhold, 1992.
5. D. Taubman and M. Marcellin, JPEG2000 Image Compression Fundamentals, Standards and Practice, New York: Springer Science+Business Media New York, 2002.
6. D. Duce, Portable Network Graphics (PNG) Specification, 8.2 chapter, 2003, unpublished.