Compression Methods for Image Processing Implementation into the Low Capacity Devices

Radim Hercík, Roman Slabý, Zdeněk Macháček

This paper describes a development of systems with limited capacity and limited performance requires a specific approach in the implementation phase especially if the application has high demands which exceeding the possibilities of the used hardware. The paper specifies the compression method used for video signal processing from the camera, which is scanned and detected in the state consumption meter application. Used hardware Atmega1284p microprocessor is not suitable for video signal processing applications according to its parameters, but it can be realized using the proposed compression methods and specific implementation of algorithms. Design and verification of compression methods is important with respect to the best method choosing for detecting the video signal.

Keywords: Compression algorithm, Compressed image, RLE, Atmega 1284p, Flash memory, RAM, C#

1 Introduction

Compression algorithms are important in the case of limited resources and low capacity of data transmission. The task of compression is to reduce the bit rate or the bit size of the data matrix. Presented data compression methods are implemented for the application of image processing in microcontrollers, where must be stored the image patterns and a set of templates on the limited memory space. These stored data are usable for later image processing and correlation calculation. There is necessary to minimize the availability of memory space, because of the system dynamic range conservation, and because of the sufficient space for data structures processing.

There is permanently removed part from the original information, thus lower image quality in the loss-making compression methods. The loss-less compression methods are less effective, but there is possible to make complete reconstruction of the original information. Presented compression methods are implemented and testing in a microcontroller Atmega1284p for image patterns storing and image processing in consumption meter application. Evaluation of the compression methods were performed by statistical analyses.

2 Description of technical solution for implemented compression methods

Implementation of compression methods is realized with Atmega1284p microcontroller, where is crystal frequency 20Mhz. It contains internal 128kB flash memory for program and 16 kB RAM memory. The external flash memory AT45DB with limited capacity 4Mbit is connected via the SPI interface. The memory is organized into 8192 pages of 512 bytes with two input SRAM buffers, which allow simultaneous writing and reading from different parts of the FLASH memory. The Flash memory is operated in slave mode, and the microcontroller is the master device, which controls the communication.

Compression methods are implemented in a device, which is used to state recognition of the consumption meters. Each of consumption meters contains the digits 0 to 9. Because of different fonts of digits, the devices have to contain a standard digit patterns for each of these fonts. Each image digit pattern is defined by the dimensions 25x50 pixels, because of the limited storage space and sufficient quality. The pattern is stored in binary format, which occupies 157 Bytes in uncompressed form. Compression methods implementation leads to significant savings in storage space with regard to the number of images and to the limited size of flash memory [1].

Compressed data of these patterns are decoded and the original image is reconstructed during the recognition process. This image pattern is subsequently correlated with the captured image from the camera. Besides to these patterns, the original color image from the camera is stored in memory AT45DB at a resolution of 640x480 pixels. It takes up to 921,600 bits, which is a quarter the size of whole memory. The memory also contains a process copy of the image and the history of recognized digits. Memory saving for images patterns storage is more important than time saving in process of decompression and reconstruction of the original image.

User application is developed in C# programming language and it is usable for testing the proposed algorithms and effectiveness comparison of the presented compression methods (Fig. 1) [2].

![Figure 1 User application for compression method verification.](image-url)
2.1 Design and implementation of method 1 based on RLE compression algorithm

Compression Method 1 is based on the loss-less compression method RLE (Run Length Encoding). There is used to encoding only one Byte. The method is designed on the reduction of repeated sequences of digits. Each sequence of encoded data is formed by one Byte, where the first part (7 bits) indicates the number of repetitions and the second (1 bit) indicates current value represented by white or black color (Fig. 2).

![Figure 2 Principles of method based on RLE algorithm.](image)

All of the identical pixel elements are calculated to one Byte in the process of pattern coding. In counting process of elements occurrence, there is computed only one row of the image pattern matrix, because of the easier reconstruction of the original pattern.

The compression process flowchart is shown in Fig. 3. The compression process consists of two cycles, where are gradually passed each row and the algorithm compares the value of the current element with the value of the previous element.

The variable number contains the elements number of the same value. There is stored value of the variable number and the value of the previous element to the compressed Byte, when the sequence of the identical elements is completed or the next row is started. Maximal sequence length of identical elements occurrence is defined by 7 bits, which corresponds to the maximal number $D_{\text{max}}$ given by algorithm

$$D_{\text{max}} = 2^7 = 128. \quad (1)$$

There is stored the new Byte on the following position, when exceeds the maximum length sequence of identical elements occurrence. The compression method can be ineffective for inadequate data, because the size of the RLE coded data can also be double in comparison to the non-compressed data.

In the case of compression digits characters the modified RLE method is effective because the images with a small number of color changes have got the minimum number of Bytes. This encoding is suitable because of easy data reconstruction. In the method implementation each row is defined by a given number of Bytes, which represent elements values changes [3],[4].

![Figure 3 Flowchart of method based on RLE algorithm.](image)

2.2 Design and implementation of method 2 based on algorithm of bit maps

The compression method 2 is based on algorithm of bit maps and it is the loss-less compression method. The principle of method is based on the storing the length of the identical elements occurrence sequence. The method includes a one start Byte, which contains information of the initial element value of the compressed image (Fig. 4).

![Figure 4 Principles of method based on bit maps](image)

The implementation of the compression method is shown in the flowchart (Fig. 5). The initialization of variables is the first step, where marks the data position in the field, where will be stored the following
compressed Byte. The variable contains the number of identical pixel elements sequence.

\[ D_{\text{max}} = 2^8 = 256 \]  

(2)

Each Byte contains the length of the identical elements sequence and the next Byte contains the length of the following inverse elements sequence. The compression method could be ineffective accordingly to the previous compression method 1 [4].

### 2.3 Design and implementation of method 3 based on algorithm of position detection

Method 3 is based on loss-less compression using the coordinate’s position system of the identical pixel values. The principle of the method is to store the coordinates of the starting position and the final position of the identical elements sequence equal to value (Fig. 6).

\[
\text{Data}[n] = \text{Image}[y,x]-1+A; \\
A = !A \\
\text{Count}++;  \\
\text{Image}[y,x] = A \\
\text{Data}[n] = \text{Count}; \\
n++; \text{Count} = 0;  \\
A = !A
\]

The method allows you to compress images with a maximal number of pixels \( D_{\text{max}} \) given by algorithm

\[ D_{\text{max}} = 2^{16} = 65536 \]  

(3)

here the Byte extension is very simple and universal for the maximal size changes of the encoded images.

### 3 Verification and testing of implemented methods for data compression in capacity limited applications

The presented compress methods were verified and the basic results are presented in the following table 1. The verification was performed in the user application in the text above. There were applied three compress methods for defined digits series of 50x25 pixels size patterns. There were performed measurements and verification for each type of method. In the left column, there is given bytes \( B \) number \( N \) of the compressed image by chosen method. On the top of the table, there is presented reference number \( R \) of bytes \( B \) of the non-compressed original image, which represents the chosen digits pattern [5]. The reference number \( R \) is same value for all individual digits patterns and it is given by presented algorithm

\[
N_{R} = \frac{H}{8},  
\]

(4)

Here \( H \) is the number of all pixels of given image pattern, thus the calculation for this application is equal

\[
N_{R} = \frac{50 \cdot 25}{8} = 156,25 = 157[B],  
\]

(5)

In the right column, there is given bytes reduction \( S \), which compares the compressed image to the non-compressed original image. The reduction \( S \) is given by

\[
S = N - N_{R},  
\]

(6)
The reduction success $P$ is comparison of the compressed image bytes number $N$ to the non-compressed image bytes number $N_R$.

$$P = \frac{N}{N_R} \times 100.$$

(7)

Table 1: Example of results from described compression methods for image digits patterns storing.

<table>
<thead>
<tr>
<th>Image</th>
<th>Used method</th>
<th>$N[B]$</th>
<th>$S[B]$</th>
<th>$P[%]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Method 1</td>
<td>158</td>
<td>1</td>
<td>100,6</td>
</tr>
<tr>
<td></td>
<td>Method 2</td>
<td>114</td>
<td>-43</td>
<td>62,3</td>
</tr>
<tr>
<td></td>
<td>Method 3</td>
<td>224</td>
<td>67</td>
<td>129,9</td>
</tr>
<tr>
<td>1</td>
<td>Method 1</td>
<td>107</td>
<td>-50</td>
<td>53,3</td>
</tr>
<tr>
<td></td>
<td>Method 2</td>
<td>98</td>
<td>-59</td>
<td>39,8</td>
</tr>
<tr>
<td></td>
<td>Method 3</td>
<td>192</td>
<td>35</td>
<td>118,7</td>
</tr>
<tr>
<td>2</td>
<td>Method 1</td>
<td>133</td>
<td>-24</td>
<td>82,0</td>
</tr>
<tr>
<td></td>
<td>Method 2</td>
<td>96</td>
<td>-61</td>
<td>36,5</td>
</tr>
<tr>
<td></td>
<td>Method 3</td>
<td>188</td>
<td>31</td>
<td>116,5</td>
</tr>
<tr>
<td>3</td>
<td>Method 1</td>
<td>154</td>
<td>-3</td>
<td>98,1</td>
</tr>
<tr>
<td></td>
<td>Method 2</td>
<td>122</td>
<td>-35</td>
<td>71,3</td>
</tr>
<tr>
<td></td>
<td>Method 3</td>
<td>240</td>
<td>83</td>
<td>134,6</td>
</tr>
<tr>
<td>4</td>
<td>Method 1</td>
<td>145</td>
<td>-12</td>
<td>91,7</td>
</tr>
<tr>
<td></td>
<td>Method 2</td>
<td>102</td>
<td>-55</td>
<td>46,1</td>
</tr>
<tr>
<td></td>
<td>Method 3</td>
<td>200</td>
<td>43</td>
<td>121,5</td>
</tr>
<tr>
<td>5</td>
<td>Method 1</td>
<td>126</td>
<td>-31</td>
<td>75,4</td>
</tr>
<tr>
<td></td>
<td>Method 2</td>
<td>112</td>
<td>-45</td>
<td>59,8</td>
</tr>
<tr>
<td></td>
<td>Method 3</td>
<td>224</td>
<td>67</td>
<td>129,9</td>
</tr>
<tr>
<td>6</td>
<td>Method 1</td>
<td>157</td>
<td>0</td>
<td>100,0</td>
</tr>
<tr>
<td></td>
<td>Method 2</td>
<td>138</td>
<td>-19</td>
<td>86,2</td>
</tr>
<tr>
<td></td>
<td>Method 3</td>
<td>272</td>
<td>115</td>
<td>142,3</td>
</tr>
<tr>
<td>7</td>
<td>Method 1</td>
<td>132</td>
<td>-25</td>
<td>81,1</td>
</tr>
<tr>
<td></td>
<td>Method 2</td>
<td>88</td>
<td>-69</td>
<td>21,6</td>
</tr>
<tr>
<td></td>
<td>Method 3</td>
<td>176</td>
<td>19</td>
<td>110,8</td>
</tr>
<tr>
<td>8</td>
<td>Method 1</td>
<td>167</td>
<td>10</td>
<td>106,0</td>
</tr>
<tr>
<td></td>
<td>Method 2</td>
<td>132</td>
<td>-25</td>
<td>81,1</td>
</tr>
<tr>
<td></td>
<td>Method 3</td>
<td>260</td>
<td>103</td>
<td>139,6</td>
</tr>
<tr>
<td>9</td>
<td>Method 1</td>
<td>144</td>
<td>-13</td>
<td>91,0</td>
</tr>
<tr>
<td></td>
<td>Method 2</td>
<td>112</td>
<td>-45</td>
<td>59,8</td>
</tr>
<tr>
<td></td>
<td>Method 3</td>
<td>220</td>
<td>63</td>
<td>128,6</td>
</tr>
</tbody>
</table>

3 Conclusion

The presented article described the compression methods and implemented algorithms for image processing into the low capacity devices, especially the digit patterns recognition advanced system for consumption meters. The theoretical part of the compression methodology is complemented by practical examples and flow diagrams of conducted measurements of three chosen methods, which are designed for the digit recognition by image processing.

The application as the user interface for parameters setting and verification of compression methods was explained. The results in Table 1 show the rate of compression methods succetivity for each digit pattern type. The red marked digits in Table 1 represent non-successfull compressed digits, respectively the compression method. There are presented additional or saved Bytes $S$ and reduction success $P$ from the number of Bytes evaluation for each compression method type.

From presented results, the method 1 based on RLE compression algorithm is succesfull for all digits except digits with spheric character, the method 2 based on algorithm of bit maps is the most succesfull method, the method 3 based on algorithm of position detection is not aplicable for digits. The comparison of compression methods and their design are very important for the effective system realization in developed low capacity devices.

Acknowledgments

The work was supported by project SP2012 /111 - Data collection and data processing of large distributed systems II.

Grand – aided student R. Hercik, Municipality of Ostrava, Czech Republic

4 References


Radim Hercik
Faculty of Electrical Engineering and Computer Science
VSB – Technical University of Ostrava
17 Listopadu, 708 33 Ostrava–Poruba
Czech Republic
tel.: +420 59 732 3497
e-mail: radim.hercik@vsb.cz

Roman Slaby
Faculty of Electrical Engineering and Computer Science
VSB – Technical University of Ostrava
17 Listopadu, 708 33 Ostrava–Poruba
Czech Republic
tel.: +420 59 732 3497
e-mail: roman.slaby@vsb.cz

Zdenek Machacek
Faculty of Electrical Engineering and Computer Science
VSB – Technical University of Ostrava
17 Listopadu, 708 33 Ostrava–Poruba
Czech Republic
tel.: +420 59 732 3497
e-mail: zdenek.machacek@vsb.cz