Efficient Large Image Browser for Embedded Systems[*]

Yuanyuan Liu, Zhiwei He, Haibin Yu, and Jinbiao Liu

Department of electronic information
Hangzhou Dianzi University
No.2 Street, Xiasha Higher education Zone
Hangzhou, China
{liuyuanyuan,zwhe,shoreyhb,ab}@hdu.edu.cn

Abstract. Images obtained by digital cameras nowadays become larger and larger in size. The quality of the obtained images is very excellent, but on the other hand, they are hard to be displayed on such embedded systems as mobile phones, digital photo frames, etc., which have both very limited memory size and very small screen size. In this paper, an efficient large image browser for embedded systems is described. A set of approaches based on pixel resampling technology are proposed to make large images to be displayed effectively. The proposed browser consumes much less memory than many famous image browser softwares. Experimental evaluations of the proposed mechanism indicate that it is efficient and effective for large image display on embedded systems.

Keywords: large image; browser; embedded systems.

1 Introduction

Statistics show that, among all the received information of a human being, more than 70 percent of them are obtained through the sense of sight. So images and videos are two kinds of very important media in people's daily life, more and more people express, share and exchange information through images and videos. According to peoples' increasing requirement and with the incremental development of CMOS sensor and VLSI technology, the pixel resolution of digital cameras and digital vidicons become more and more high, e.g., a digital camera with a 10Mega-pixels CMOS sensor is very common nowadays. The quality of the captured images is accordingly very high, but the memory consumption of such images is also very high, especially when they are browsed by an image viewer. Suppose we have a full color image captured by a 10-mega-pixels digital camera, whose resolution is 3648*2736, then it consumes a memory size of about 30M bytes for the image to be fully displayed. There is of course no problem for today's personal computer to display such an image, because it can have a memory of more than 1G bytes, with many

[*] This work was supported by Foundation of Department of Education of Zhejiang Province with Grant No. Y200907573 and partially supported by Zhejiang Province Nature Science Foundation of China with Grant No. Y1090881.

softwares run on it that can be applied , such as Adobe Photoshop and ACDSee etc. But if the hardware platform changes from PC to embedded systems such as the widely used mobile phones, digital photo frames, handheld PCs, where image browser is also a very important application, problems may occur because the size of the internal memory is very limited in such systems, and the screen is much smaller. For example, the typical embedded system—an operating system supported mobile phone— may have a internal memory size of 32M bytes, a LCD touch screen of the size of 4.3 inches with a resolution of 800*480 pixels. As the operating system run on the mobile phone also needs to consume memory, the memory left for other applications is much more less than 30M bytes. So measures should be taken before the large images can be browsed in such memory limited embedded systems.

Many efforts have been put on image adaptation and related fields from quite different aspects, which cover from the coding process to image semantic analysis.

For instance, the ROI coding scheme and Spatial/SNR scalability in JPEG 2000 [1] have provided a functionality of progressive encoding and display. It is useful for fast database access as well as for delivering different resolutions to terminals with different capabilities.

Jin Li [2] etc. implemented an interactive large image browser base on the JPEG 2000 standard. During the browsing process, the user specifies a region of interest (ROI) with certain spatial and resolution constraint, the browser only downloads the portion of the compressed bit stream that covers the current ROI, and the download is performed in a progressive fashion so that a coarse view of the ROI can be rendered very quickly and then gradually refined as more and more bit stream arrives. In the case of the switch of ROI, e.g., zooming in/out or panning around, the browser uses existing compressed bit stream in cache to quickly render a coarse view of the new ROI, and in the same time, request a new set of compressed bit stream corresponding to the updated view.

The JPEG 2000 based methods requires the images to be encoded according to the JPEG 2000 standard first, but for the images obtained by most of today's digital cameras, the stored image format is just '.jpg' which are encoded according to the JPEG standard. Because the conversion of the image format from JPEG to JPEG 2000 can only be done with a JPEG decoding followed by a JPEG 2000 encoding, there is no way to do the conversion directly. With the same memory limitation problem, apparently it is impossible to convert the image format from JPEG to JPEG 2000 on the embedded system before the images can be browsed. And it is also not convenient to require the users to convert the image format on a PC before it can be browsed on the embedded system. It can be concluded that the JPEG 2000 based methods is not suitable for the browse of large images in an embedded system.

Xing Xie etc. proposed an attention model based image adaptation approach in [3], [4]. Instead of treating an image as a whole, they manipulated each region-of-interest in the image separately, which allowed delivery of the most important region to the client when the screen size is small. In [5], they further extended the image attention model [3] to include a time constraint to calculate an optimal browsing path to maximize the information throughput under limited space and time. The main problem of the attention model based methods is that they consumed an average of more than 8 seconds to browse an image, which is actually a bit longer.

In this paper, a resampling based method is proposed and used to browse large images in embedded systems. The organization of the paper is as follows. Section 1 is an introduction of the whole work, Section 2 gives the framework of the whole system, Section 3 shows the detail description of the proposed methods, Section 4 gives the test results and the conclusion is given in Section 5.

2 The System Framework

For most of today's digital cameras, the captured images are usually stored in the format of "JPG" or "RAW". Those images with a postfix of "JPG" in the file name are encoded according to the JPEG standard, so they should be firstly decoded to obtain the color information at each pixel position before they can be browsed; while the images with a postfix of "RAW" in the file name are not compressed at all, the color information at each pixel can be read directly from the file. Thanks to the official reference implementation of the JPEG decoder, we need not to decode the whole image at one time to obtain the color information of pixels, but each time we can only decode one line to obtain the color information of pixels in that line. This is really very good news for us, otherwise we can not browse the "JPEG" encoded images at all on memory limited embedded systems for the decoding procedure consumes too much memory. With this in hand, we can just treat both of these two types of images with no differences.

For an input image, a complete system framework of the proposed approach to browse it in an embedded system is shown in Fig. 1.

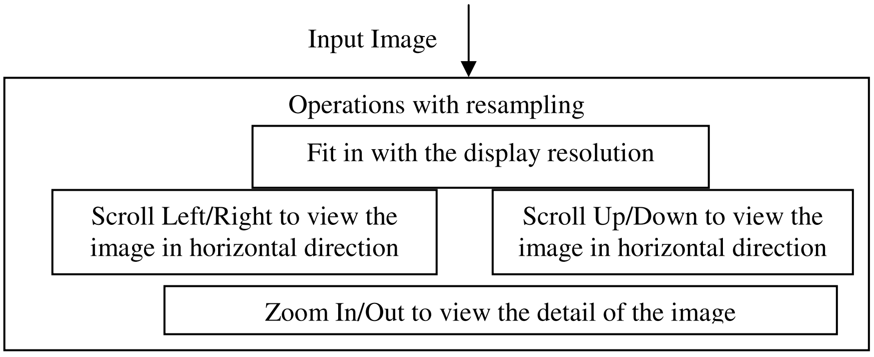
Input Image
Operations with resampling
Fit in with the display resolution
Scroll Left/Right to view the image in horizontal direction
Scroll Up/Down to view the image in horizontal direction
Zoom In/Out to view the detail of the image

Fig. 1. System framework of the resampling based image browsing

The whole framework contains four types of operations: Fit in with the display size, Scroll Left/Right, Scroll Up/Down and Zoom In/Out. Among all the operations, "Fit in with the display size" is a basic one and also a built-in one; it is needed by all other three operations and called automatically whenever needed. "Scroll Left/Right" is used to display the content of the image which is in a left/right position compared to the current position. Similarly, "Scroll Up/Down" is used to display the content of the image which is in an up/down position compared to the current position. The "Zoom In/Out" operation can let the user view the image in a finer/coarser manner.

All the operations, except for the “Fit in” one, are user-interactive. Each time an operation is made, a resamping procedure to the original image is needed.

3 Operations Detail

Suppose the display resolution is $W_s \times H_s$, which means that it can display W_s pixels in the horizontal direction and H_s pixels in the vertical direction. On the other hand, we have an image which has a resolution of $W \times H$ to be browsed. We also suppose that both W and H are much larger than W_s and H_s. Under this condition, the detail of the operations for image browsing is as follows. One thing important which should be mentioned is that all the operations mentioned before are disabled when no image is opened.

3.1 The Fit in Operation

The “Fit-in” operation is automatically activated when a new image is browsed. The process of fitting in with the display resolution is depicted in Fig.2.

The image is opened in a binary mode; the lines are read in one by one and only those lines which is an integer times of the value W / W_s are preserved for further handling. The preserved lines are then resampled every H / H_s pixels. Finally, an image with the resolution of $W_s \times H_s$ is obtained and can be displayed on the screen.

When the image is browsed in this manner, the “Zoom-In” operation is activated for the user to view the details of the image. Other operations are still disabled at this time.

3.2 The Zoom In/Out Operation

In our proposed methodology, the “Zoom-In/Out” operation has very little differences with the “Fit-in” operation except that one more parameter z_r, which stands for “Zoom Ratio”, controls the resampling procedure. The value of z_r is in the range of W / W_s to 1. Each time a “Zoom-In” operation is made, z_r is decreased to one half of its original value; on the other hand, when a “Zoom-Out” operation is made, z_r is increased to two times of its original value. If the “Zoom- Ratio” equals to 1, the “Zoom In” operation is disabled; and when we have a z_r of W / W_s, it is just the “Fit-in” case, the “Zoom Out” operation is disabled. Different from the “Fit-In” operation where we extract every W / W_s lines and pixels, we extract every z_r lines from the original images and every z_r pixels from each of the extracted line from the beginning in the “Zoom-In/Out” operation, the extraction process stops when

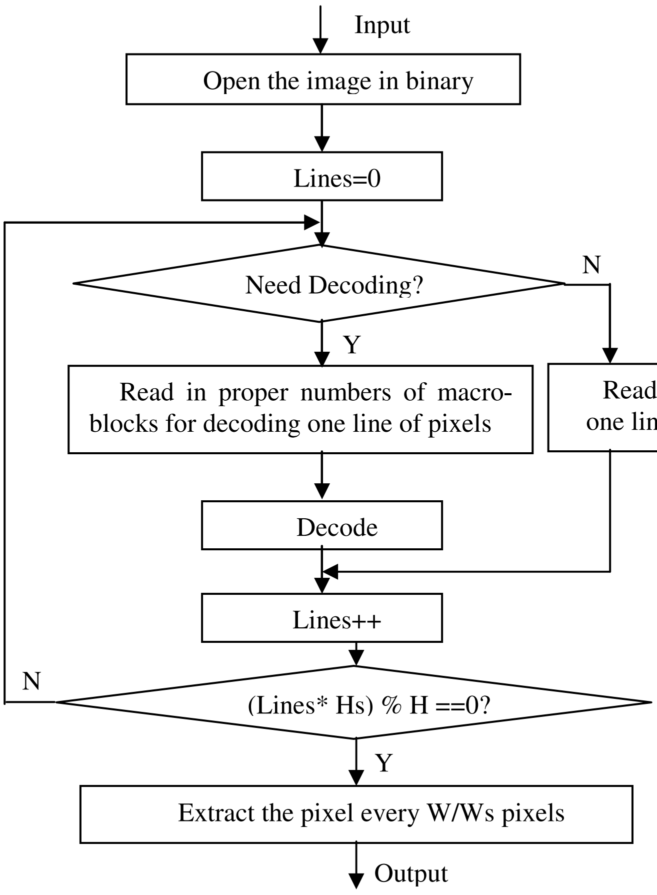

Fig. 2. The Fit-in Process when the image is firstly opened

totally H_s lines are extracted with each line containing W_s pixels. When z_r is smaller than W / W_s, the "Scroll Left/Right" and "Scroll Up/Down" operation are activated.

3.3 The Scroll Left/Right Operation

As we discussed in the preceding subsection, when the "Zoom Ratio" z_r is smaller than W / W_s, there are W / z_r pixels, which is larger than W_s, that can be extracted if we extract the lines every z_r pixels, but the screen can only display W_s pixels in the horizontal direction, so the scroll left/right operation is activated for the user to view those part that are not displayed. The scroll position, which is defined to be s_r, is in the range between 0 and $\frac{W}{z_r} - W_s$. With this s_r in hand, the "Scroll Left/Right" operation is almost the same with the "Zoom In/Out" operation except that the resampling starts from the horizontal position of s_r other than 0, which means the pixels at the position of

$$s_r, s_r + 1 \cdot z_r, s_r + 2 \cdot z_r, ..., s_r + (W_s - 1) \cdot z_r$$

are extracted at this time.

3.4 The Scroll Up/Down Operation

The Scroll UP/Down operation is very similar to the Scroll Left/Right operation. We also define a variable s_d, which is in the range between 0 and $\frac{H}{z_r} - H_s$, to represent the scroll position in the vertical direction; and only the lines at the position of

$$s_d, s_d + 1 \cdot z_r, s_d + 2 \cdot z_r, ..., s_d + (H_s - 1) \cdot z_r$$

are extracted.

3.5 Zoom In/Out after Scroll

Suppose somebody wants to view the detail of the image at the center, he can zoom in the image as large as possible so that he can view the detail and then drag the scroll bar to the position he want, but he can also zoom in the image only to some extent and then drag the scroll bar to the position and then zoom in again if he wants. So a zoom in/out operation after scroll is also possible, and this time we need not only to change the "zoom ratio" z_r but also the scroll position s_r and s_d. Every time a zoom in operation is made, s_r and s_d should be decreased to one half of their original values, on the other hand, when a zoom out operation is made, they should be increased to two times of their original values.

4 Experimental Results

The proposed operations have been implemented with the software of Embedded Visual C++ 4.0 under the Operating System of Windows CE 5.0. The Graphical User Interface is shown in Fig. 3.5.

The image opened in Fig.3 has a resolution of 23622*11811 pixels, with each pixel occupies 1 bit. The memory consumed by the software is 9982K Bytes. It can be displayed in less than 1 second in an embedded system with a CPU run at 400 MHz and with a memory size of 64MB. The image should be re-read every time a new

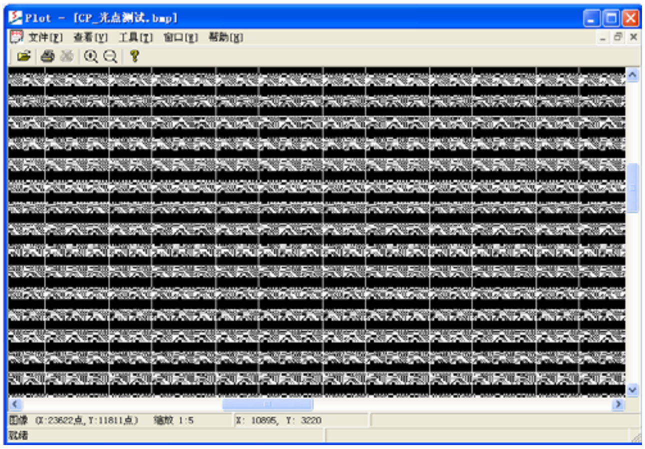
Plot - [CP_光点测试.bmp]
文件(F) 查看(V) 工具(T) 窗口(W) 帮助(H)
图像 (X:23622点, Y:11811点) 缩放 1:5
X: 10895, Y: 3220
就绪

Fig. 3. GUI of the implemented software

operation mentioned above is made, but the re-display time is still less than 1 second. With a comparison, we also display the image in other popular image processing softwares such as ACDSee and Adobe Photoshop. The results are given in Table 1.

Table 1. Comparison to other popular softwares

Software	System	Memory Occupied	Display time
ACDSee 5.0	Windows XP, Pentium D 3.4GHz, 1GB Memory	16MB	3.0 s
PhotoShop 8.0	Windows XP, Pentium D 3.4GHz, 1GB Memory	120MB	<1.0s
The proposed	WinCE 5.0, PXA 270 400MHz, 64MB memory	10 MB	<1.0s

5 Conclusions

In this paper, an efficient and effective method for displaying very large images is proposed. The images are viewed in an interactive manner with several operations such as Zoom In/Out, Scroll Left/Right and Scroll Up/Down supplied. For all the operations, one basic procedure, the resampling, is needed. The starting positions in both the horizontal and the vertical directions, companioned with the zoom ratio are used for the decision how the resampling procedure should be made. Experimental results show that our framework is very suitable for applications on embedded systems.

The drawback of the proposed method is that aliasing may exist when the down-sampling ratio is big, e.g., when the image is firstly opened and fit-in with the screen.

The user can improve this by zooming in the image a bit. Another solution to the aliasing problem is that we can perform some smoothing operations to the extracted lines and pixels. For instance, we can extract several lines at one time and make a linear combination of them to obtain one smoother line, which can decrease the influence of the aliasing effect. But this will increase the memory consumption along with the display time. A tradeoff should be made then.

References

1. Christopoulos, C., Skodras, A., Ebrahimi, T.: The JPEG2000 still image coding system: an overview. IEEE Trans. Consumer Electron. 46(4), 1103–1127 (2000)
2. Li, J., Sun, H.-H.: On Interactive Browsing of Large Images. IEEE Trans. on Multimedia 5(4), 581–590 (2003)
3. Chen, L.Q., Xie, X., Fan, X., Ma, W.Y., Zhang, H.J., Zhou, H.Q.: A visual attention model for adapting images on small displays. ACM Multimedia Syst. J. 9(4), 353–364 (2003)
4. Fan, X., Xie, X., Ma, W.Y., Zhang, H.J., Zhou, H.Q.: Visual attention based image browsing on mobile devices. In: Proc. ICME 2003, Baltimore, MD, vol. I, pp. 53–56 (July 2003)
5. Xie, X., Liu, H., Maand, W.-Y., Zhang, H.-J.: Browsing Large Pictures Under Limited Display Sizes. IEEE Trans. on multimedia 8(4), 707–715 (2006)