Int. J. Appl. Math. Comput. Sci., 2007, Vol. 17, No. 1, 129–136
DOI: 10.2478/v10006-007-0012-5

amcs

A PROPOSITION OF MOBILE FRACTAL IMAGE DECOMPRESSION

SŁAWOMIR NIKIEL

Institute of Control and Computation Engineering, University of Zielona Góra
ul. Podgórna 50, 65–246 Zielona Góra, Poland
e-mail: S.Nikiel@issi.uz.zgora.pl

Multimedia are becoming one of the most important elements of the user interface with regard to the acceptance of modern mobile devices. The multimodal content that is delivered and available for a wide range of mobile telephony terminals is indispensable to bind users to a system and its services. Currently available mobile devices are equipped with multimedia capabilities and decent processing power and storage area. The most crucial factors are then the bandwidth and costs of media transfer. This is particularly visible in mobile gaming, where textures represent the bulk of binary data to be acquired from the content provider. Image textures have traditionally added visual realism to computer graphics. The realism increases with the resolution of textures. This represents a challenge to the limited bandwidth of mobile-oriented systems. The challenge is even more obvious in mobile gaming, where single image depicts a collection of shots or animation cycles for sprites and a backdrop scenery. In order to increase the efficiency of image and image texture transfer, a fractal based compression scheme is proposed. The main idea is to use an asymmetric server-client architecture. The resource demanding compression process is performed on the server side while the client part decompresses highly packed image data. The method offers a very high compression ratio for pictures representing image textures for natural scenes. It aims to minimize the transmission bandwidth that should speed up the downloading process and minimize the cost and time of data transfer. The paper focuses on the implementation of fractal decompression schemes suitable for most mobile devices, and opens a discussion on fractal image models for limited resource applications.

Keywords: fractals, image processing, fractal image compression, mobile media, texture mapping

1. Introduction

Image textures are commonly used to add visual realism to computer graphics. Multimedia systems exploit several types of images to enhance the content delivered to the user. Images as textures can occupy a considerable amount of memory and bandwidth during the transmission. This is particularly challenging in mobile media systems, where cost and energy efficient solutions are always demanded. Many different techniques have been proposed to reduce both the bandwidth and size of the images. Hardware image compression based on a lossy scheme offers a very high performance (Knittel *et al.*, 1996). The texture is decompressed on-the-fly during the download, but its application area is not platform independent and is limited to a specific implementation. In some applications (visualization and gaming), there is a need for further processing of textures such as MIP mapping and Region Of Interest (ROI) based random access to some areas in the image texture. The execution time for compression should be short, though it is not as crucial as the time of the decompression process. A lot of work has been done in the field of image compression to meet those demands. Research devoted to compression of digital images and image textures is discussed in Section 2.1. In the field of mobile media systems, the limited processing power and narrow bandwidth present additional challenges to developers. The image texture compression presented in this paper was originally targeted for PC and game consoles, but it is in no way limited to those platforms (Stachera and Nikiel, 2004). Sections 2.3 and 2.4 discuss the fractal image coding and decoding that can be implemented on mobile devices: Java-enabled "smart phones." The method utilizes an image compression method to compress texture collages used in mobile applications. The lack of floating point operators and limited memory resources require substantial modifications and limitations to classical fractal decompression schemes and practically make fractal compression useless in such environments. Hence the proposed client-server architecture results. The application prototype and preliminary results are presented in Section 3. In the present work the method has moderate

image quality. As soon as a new MIDP profile is implemented in popular mobile phones, the image quality will be substantially improved. The author has concentrated mainly on PIFS image coding. The paper delivers information for potential developers interested in fractal imaging on mobile platforms.

2. Image Texture Compression

2.1. Previous Work. This section presents recent research related to the image and image texture compression. Generally, image compression algorithms can be organized into three main groups: transform coding methods (including Discrete Cosine and Discrete Wavelet Transform—DCT and DWT), Vector Quantization (VQ) methods and Block Truncation Compression (BTC) (Delp and Mitchell, 1979).

The most popular transform methods used in image compression are based on DCT functions (Microsoft, 1997). A more efficient image texture compression scheme also based on the DCT was proposed in (Chen and Lee, 2002). The method used adaptive quantization of pixel blocks that resulted in a fixed-length code and offered one of the highest compression rates with very good image quality. However, it proved to be too computationally expensive for mobile implementations. The wavelet model can also be used to analyze and compress signals (Ghulam *et al.*, 2004). Texture compression is a field related to image compression and describes a wealth of data processing techniques. DWT coding exploits the multiresolution image representation (Perebrin, 1999). The texture was converted to the YUV-color model and achieved the compression ratio $Cr = 6 : 1$. It is used in JPEG 2000 image coding. Further reductions of the insignificant coefficients resulted in even higher compression rates and random access to the elements of the image texture (Condissi *et al.*, 2005). The method offers superior quality of textures but is too complex to be implemented on currently available mobile devices.

VQ was proposed to be a compression method that delivers MIP-mapping capability (Beers *et al.*, 1996). The compression scheme achieved a significant efficiency of $Cr = 24 : 1$ at the cost of low image quality due to code word sub-sampling. The interpolative vector quantization method follows the scheme. It has the pyramid representation and two codebooks storing data corresponding to low and high frequency texture elements (Kwon *et al.*, 2000). Generally, VQ based methods suffer from indirect data access during codebook construction that results in additional data caching.

The methods presented above are the mainstream of image texture compression. A sample comparison of compression ratios of YUV based JPEG, JPEG 2000 and fractal image compression implemented in the proposed server-side image compressor (discussed in the next section) is depicted in Table 1 for the test images of Fig. 1. All compression ratios depend on the image size and its content: we can observe different Cr values for comparable image quality (with PSNR oscillating around 31–35 dB).

Table 1. Comparison of selected image compression methods.

Image	FCI (32×32)		JPEG		JPEG 2000	
	Cr	PSNR [dB]	Cr	PSNR [dB]	Cr	PSNR [dB]
Sky	63.31:1	35.68	161.1:1	31.4	542.9:1	36.7
Map	51.18:1	29.8	89.2:1	28.7	126.9:1	30.1
Lenna	74.1:1	26.7	24.1:1	27.8	77.9:1	28.58

*Cr – compression ratio, PSNR – peak signal to noise ratio, FCI (32×32) defines the size of range blocks.

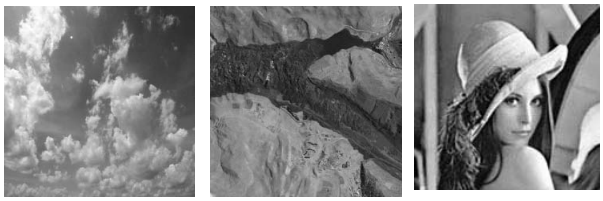


Fig. 1. Test images used for comparison (Sky, Map, Lenna).

2.2. Fractal Image Coding. The process of image coding is based on a set of contractive transformations W such that its fixed point f_w is an approximation to the coded image I. Thus W defines a lossy code for the image I (Baharav *et al.*, 1993; Stachera and Nikiel, 2004). The compression scheme for image textures is based on a block oriented fractal compression scheme for images. The image coding is performed on the server side and then published for mobile clients. The algorithm should allow for the implementation of Region Of Interest (ROI) access. Considering further image decompression on a mobile device, some additional assumptions extending the basic fractal-encoding scheme are defined:

1. $N \times N$ represents the size of the texture ($N = 2^l$).
2. $R \times R$ defines the size of range blocks M_R ($R = 2^n$)—for a mobile application it should be small, e.g., 4×4 pixels.
3. $D \times D$ defines the size of domain blocks M_D to be twice the size of range blocks ($D = 2R$),
4. $D_h = R - D_h$ defines the distance between consecutive domain blocks—then the number of blocks in the domain pool is equal to

$$M_D \equiv \left(\frac{N - D}{D_h} + 1 \right), \tag{1}$$

where each domain block is defined as

$$d_{mj} = I\left(\left(m_j - 1\right) D_n + j\right), \quad m_j = 1, 2, \ldots, M_D, \quad j = 1, 2, \ldots, D. \quad (2)$$

5. $\varphi(\cdot)$ defines the spatial contraction function that averages four adjacent texture elements and then maps the averaged value onto the range block applying one of eight isometries. $\varphi(\cdot)$ is defined as follows:

$$\varphi\left(d_{mi}\right)(j) \equiv \frac{1}{2}\left(d_{mi}\left(2j\right) + d_{mi}\left(2j-1\right)\right), \quad j = 1, 2, \ldots, R, \quad (3)$$

$\varphi(\cdot)$ shrinks domain blocks to the size of range blocks with averaging pairs of $d_m i$ from the domain pool. The resulting range block values are scaled by $s_i = 1/2$ and added to o_i.

Let I be an encoded image. A set R of non-overlapping range cells $R = \{r_1, r_2, \ldots, r_n\}$, $r_i \cap r_j = \emptyset$ that tile I, $I = \bigcup r_i$ is called the range pool. A set D of overlapping domain cells $D = \{d_1, d_2, \ldots, d_m\}$ is called the domain pool $d_i \in I$. A set W is composed of contractive transformations $w_i : d_i \rightarrow r_i$. For a compact representation, W is restricted to the class of transformation given in the form

$$r_i = w_i(d_i) = s_i \varphi(d_i) + o_i, \quad (4)$$

where the set of range blocks R comes from a quad-tree partition of the texture. A mobile application demands the simplification of the problem to a uniform partition of the image texture. Classical PIFS compression methods exploit also hierarchical and Delannuay partition methods (Stachera and Nikiel, 2004). Furthermore, φ is a spatial contraction function which contracts domain cells to the size of range cells (for a mobile prototype, $\varphi = 0.5$ is chosen), s_i is a scaling factor ($s_i \in R$, $|s_i| < 1$), o_i is an offset value, $o_i \in R$ (Fisher, 1995).

In terms of images, s_i controls contrast, o_i controls brightness and φ averages domain cell values. The triplets (φ, s_i, o_i) are called transform parameters.

The encoding problem of the image I is stated as follows:

1. Partition I into non-overlapping range cells $r_i \in R$ that tile I,

$$r_i\left(j\right) = I\left(\left(i-1\right) R + j\right), \quad i = 1, 2, \ldots, M_R, \quad j = 1, 2, \ldots, R. \quad (5)$$

2. For each range cell r_i find domain cells (Eqn. (1)) that are defined in (2).
3. Find transform parameters such that $d(r_i, s_i\varphi(d_i) + o_i)$ is minimized.
4. Save transform parameters (φ, s_i, o_i).

The process of image coding arrives at a set of contractive transformations W such that its fixed point f_w is an approximation of the image I,

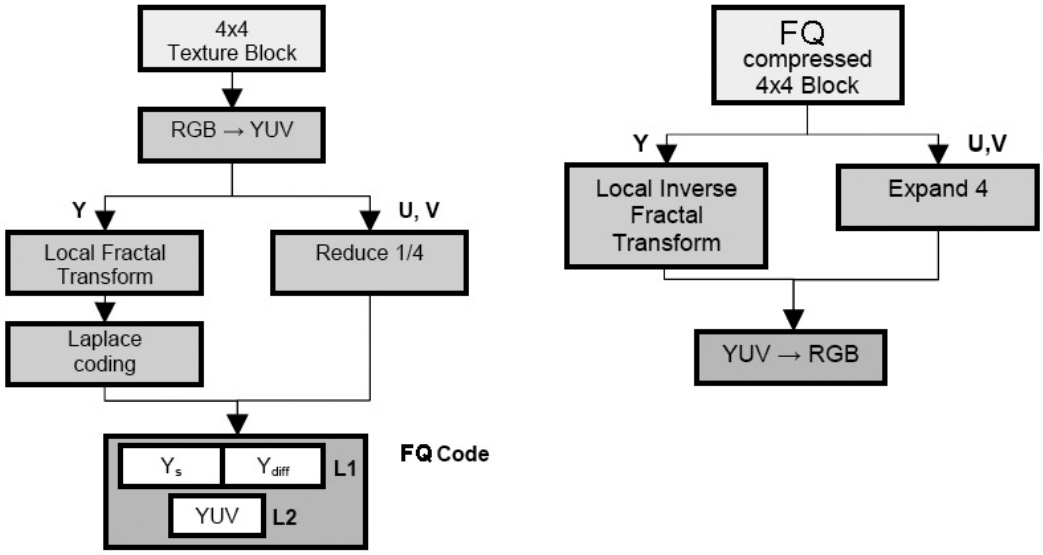
$$f_w = I \cong W\left(I\right). \quad (6)$$

Thus, storing W instead of the original image defines a lossy code for the image I (Baharav *et al.*, 1993). A typical fractal coding scheme does not allow for a local decoding of images. It is not possible to decode only a selected region of a texture without decoding all the domains that are related to it. It is possible to solve the problem of local decompression by restricting the search area to a given range (Stachera and Nikiel, 2004). The search regions are defined by a limited number of quad-tree partitions of the texture less than the number of minimal partitions. The search regions represent nodes at a level of the tree set between the root (an initial texture) and the minimum tree depth. Each region is a square area with the size at least twice the size of range regions. It also defines an independent domain pool. The image compression algorithm compares only ranges with domains that are contained in the same search region. Each search region can be handled independently of the others. It allows ROI access to texture regions and local decompression that is invaluable in texture mapping applications. The original fractal compression scheme implements the YUV-color model. The YUV-color space consists of three channels: a luminance channel Y and two chrominance channels U (hue), V (saturation). The chrominance channels store information about color that can be compressed with a high ratio still delivering little to no visible degradation. As for a better compression for mobile devices, the YUV-color model is proposed. The further hardware interpolation of neighboring pixels (cf. Section 3.1) reduces noise introduced by the lossy compression.

The overall scheme for fractal image compression, cf. Fig. 2(a), can be described in the following steps:

1. The RGB-color space is converted to the YUV model with each channel compressed separately.
2. A minimum quad-tree depth $q_{\min}$, a maximum quad-tree depth $q_{\max}$ and a search region depth $q_{\text{search}} \in [1, \ldots, q_{\min}]$ are defined.
3. A quad-tree partitioning method and a region search strategy are is used for each component Y, U, V.
4. The header information: the quad-tree depth, the number of transformations for each component, texture resolution, etc., is saved.
5. The quantized transform parameters with quad-tree information for each component using a variable length code are stored.

The independent processing of each YUV-component makes it possible to implement either



(a) compression (b) decompression

Fig. 2. Fractal imaging.

parallel processing during the process of texture compression or multiple threads in the decompression performed on a mobile device. Due to its very high complexity, the entire process of image texture compression is not appropriate for mobile devices.

2.3. Fractal Image Decoding. The decoding scheme for image textures must be adapted to the specific characteristics of mobile systems including limitations of mobile media API. The scheme must deliver a relatively fast and inexpensive method to produce preferably very high compressed images. The main drawback of the presented scheme is a computationally expensive compression algorithm. Hence the coding of image textures must be performed on the server-side of a multi-tier system. The method is characterized by a number of relatively simple decoding algorithms. Several fractal-decoding methods have been developed to optimize the decompression process (Fisher, 1995, Skarbek, 1998). The hierarchical decoding method is one order of magnitude less computationally expensive than the iterative method (assuming $B^2 \gg it$, where it is equal to the number of iterations in the iterative method) (Cisar, 1996). The texture can be decompressed in a finite predetermined number of steps that depend on partitioning the texture, rather than on the texture image itself. The hierarchical method was introduced only for range blocks of a fixed size. The scheme proposed in the paper is a modification of that method, extending the case of quad-tree partitioning (Malah and Sudskover, 1999). One of the most important properties of fractal image representation is the resolution independence of natural images, i.e., the so-called super resolution. In contrast to linear interpolation, which tends to blur the image texture, the fractal decompression method preserves the richness of details even at a resolution higher than the original one. In the implemented fractal compression scheme the chrominance information is averaged (Stachera and Nikiel, 2004). The super resolution property of the fractal decompression makes it possible to decompress the chrominance channels (hue, saturation) at the original resolution without the loss of visual details.

The fractal decoding scheme starts with a fractal compressed image downloaded from a server on a mobile device. Image texture blocks with sizes depending on the search region are used during the compression process. The decompression scheme is based on a hierarchical decompression method. In the first step, the transformations for a given image texture block (range block and domain block sizes) are scaled by a factor of $1/2^{\max}$ (where $2^{\max} \times 2^{\max}$ is the size of the biggest range block) in order to approximate the highest level of the PIFS pyramid.

The transformation is applied at that level only once to approximate the fixed point $f^{1/2^{\max}}$. Then the resolution is doubled. The process is repeated $\log_2(2^{\max})$ times. The image decompression process (Fig. 2(b)) for a given texture block may be described in the following steps, for each RGB component (starting from any non-empty image):

1. The transformation is applied to the top level.
2. The transformation is multiplied by 2.
3. The transformation is applied to the image.
4. Steps 3 and 4 are repeated until the required resolution is achieved.

5. The data are buffered.
6. The image is converted from the YUV to RGB model which is ready for display.

3. Implementation and Experiments

3.1. Mobile Media. The ability to display, manipulate and register digital images is a common element of currently available "smart phones." Most GSM terminals are open to third-party applications designed in Java or C languages and implemented for the Symbian operating system. As a programming environment, the Java language seems to be more versatile. Being a platform-independent and open standard, Java 2 Micro Edition offers a valuable solution for fast growing mobile media market. Modern mobile phones are Java-enabled and implement advanced (Atkinson *et al.*, 2001) Mobile Information Device Profiles. MIDP 2.0 defines a number of packages that handle graphical user interface design, image manipulation, gaming and media processing.

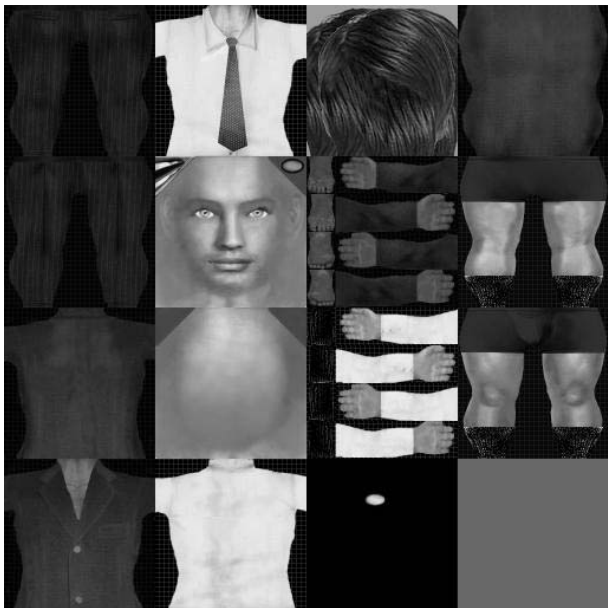


Fig. 3. Sample image used for `TiledLayer` onscreen composition.

`TiledLayer` is the most popular technique for composition of image textures that are acquired from a single digital image (Fig. 3). PNG (Portable Network Graphics) is a native to Java digital image file format that offers run-length encoding schemes. JSR184 capable mobile terminals can use `*.jpeg` images for composite textures. The author assumes that fractal compression will surpass the JPEG compression method when applied to modern terminals with better processing power. Low-level image manipulation is limited to few functions available in J2ME, namely, the extraction of RGB values of a pixel at a given position, operation on a vector of integer values and setting up a pixel at a given position on "canvas" or "game canvas." Color values depend on hardware implementation and the profile of given GSM terminals. A number of neighboring pixels can have the same value assigned even if they are different in the original image. This may cause problems in the application of advanced image processing algorithms, although most of the classical low-level image manipulation methods can be successfully implemented (Nikiel and Moczulski, 2006, Pazio and Cisowski, 2005).

3.2. Application Prototyping. Fractal compression is a highly asymmetric process, where image coding is far more complex and resource demanding than image decoding. Considering that fact, a server-client architecture delivering compressed media for mobile devices is proposed (Fig. 4). The computationally expensive operations of compression are performed on the server, while the simplified decoding scheme is implemented in a Java2ME midlet running on a mobile phone (Fig. 5). Except the limitations mentioned in the previous section, the Java application cannot facilitate pointer operations and must use static declaration of types, which presents problems with code optimization. Fixed-point numbers present another obstacle compared with classical fractal compression implementations. It was necessary to add a floating-point number emulator to perform properly the decoding process. In most mobile phones the operating system limits its memory space for Java applications: one of the tested prototypes was implemented on a MIDP 2.0, CLDC 1.0 SonyEricsson T630 Mobile Phone with approx. 500 kB of RAM available (Fig. 6) (Bury, 2004). During the experiments with this and other prototypes, the compression/decompression parameters were chosen to meet the limited processing power of mobile devices. Only 4 levels of quad-tree partitioning were allowed (MinPart). The number of iterations (niter) was limited to a maximum of 10. Domains were chosen to be just-touching with the size two times the size of range blocks. The decompression process delivered moderate image quality, mostly due to the emulation of floating-point numbers and hardware approximation of neighboring pixels. Another problem has its origin in the limited memory resources of the tested GSM terminals. It resulted in minimal memory usage

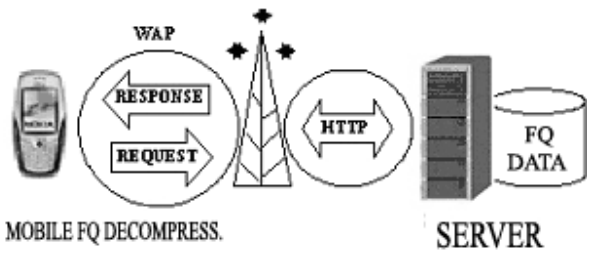


Fig. 4. Client-server architecture for the fractal compression scheme.

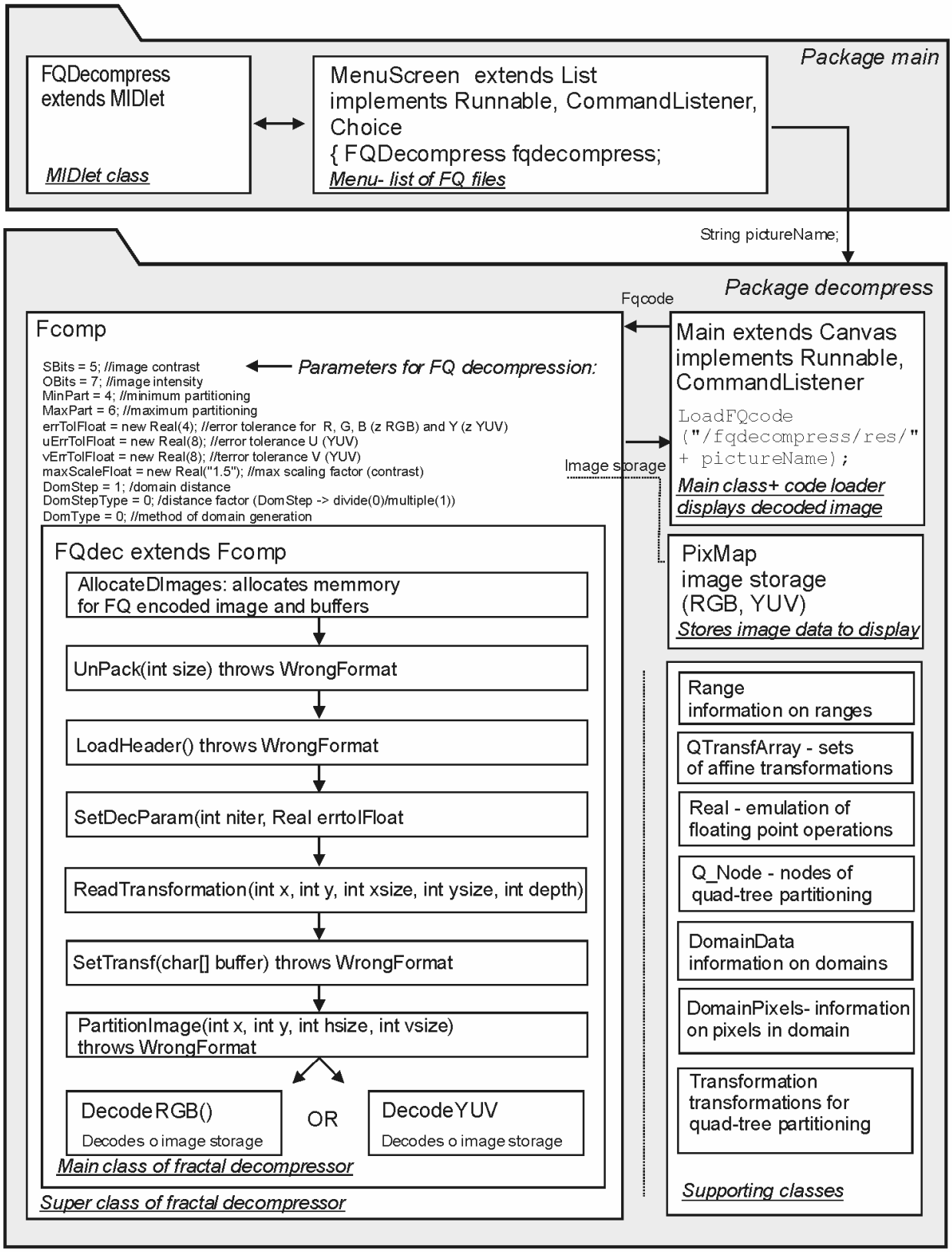


Fig. 5. Structure of a sample prototype of a fractal image decoder implemented in J2ME.

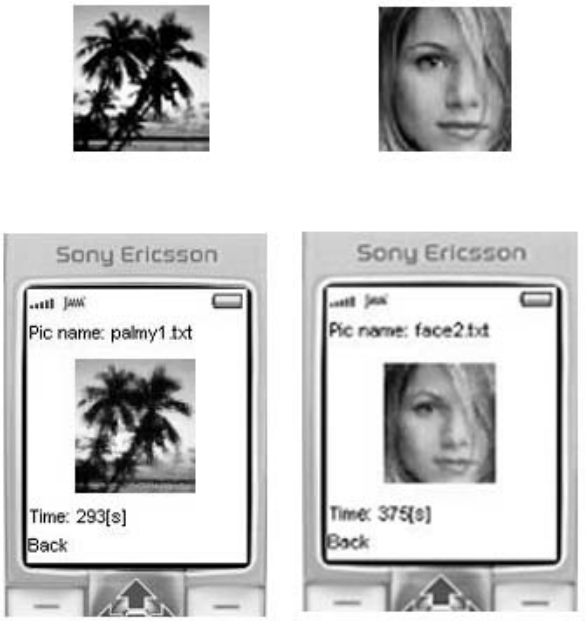


Fig. 6. Decompressed fractal images (bottom) compared to the original picture (top).

required for extensive data buffering implemented in the quad-tree image decomposition fractal decoder, and careful choice of image coding parameters. The presented structure of the decompressing midlet is not optimized. The author currently investigates prototypes developed for the CLDC 1.1 profile and for CDC (Palmtop devices).

Nevertheless, the mobile application prototype was running properly and quite efficiently for small image textures. Table 2 summarizes the results of the experiments. The tests were performed for image resolution ranging from 50×50 pixels up to 500×500 pixels. We have to remember that the display of the GSM terminal is limited to approximately 200×170 pixels and that the average image texture element has a resolution of 20×20 pixels. We can clearly observe better image compression of the Fractal Method (FQ) when compared with BMP and JPEG, although at the cost of decompression time. Savings that are clearly visible in the transmission time (related strictly to the size of the data transferred) are reduced in the non-

Table 2. Decompression results for different image sizes (image Lenna).

Image Resolution (pixels)	Time of compression (Pentium IV)	Image size			Time of decompression (mobile device)
		BMP [kB]	JPG [kB]	FQ [kB]	
50×50	16ms	4	1	1	4 sec.
100×100	16ms	30	20	5	20 sec.
150×150	32ms	67	30	5	50 sec.
500×500	100ms	770	60	6	13 min. 20 sec.

linear increase in the decompression time with the size of the image given in Table 3. The given results are subjective (the tests were performed in Warsaw) and, as can be seen, far below the nominal speeds given in 3G specifications. However, they depict a situation that every user of a mobile system can spot in real life, if not in large cities, at least in remote locations. Assuming the typical size of Java games as 500 kB (including the game code and images/sounds), it takes around two minutes to download the complete code with the GPRS protocol. Comparing the average image texture elements with a resolution of 20×20 pixels, the cost and time of image transfer can then be neglected.

Very long decompression time is observed for larger images. It is due to minimal settings of the Sun WTK emulator and non-optimized software emulation of floating point calculations (for CLDC 1.0 terminals). The currently implemented method suggests the division of texture images into smaller images and separate decoding. The average compression ratio oscillates around $Cr = 30 : 1$ and depends on the image size and content. The rendering time and image quality will be substantially improved for CLDC 1.1 terminals. The method can be hardware accelerated, thus improving the speed of the decoding process, although at the cost of the platform independence.

4. Conclusions

Considering mobile devices such as "smart phones" and portable consoles, it is very important to limit

Table 3. Tested data transfer in mobile networks.

Data type	Warsaw-Chomiczówka		
	Sony Ericsson Z1010 UMTS	Merlin U530 UMTS PC Card – UMTS	Merlin U530 UMTS PC Card – GPRS
text	39.9 KB/s	20.3 KB/s	5.30 KB/s
images/application (`*.jar`) 100 kB	2.5 sec. / 42.6 KB/s	6 sec. / 16.6 KB/s	21 sec. / 4.78 KB/s
average access to WWW	20 kB/s	25 kB/s	4 kB/s

the bandwidth usage in order to reduce both the costs of data transfer and power consumption. Collages of the image texture are a common part of mobile gaming applications. Consequently, with increasing resolutions of mobile displays, the problem of larger texture maps will be more evident. Additionally, mobile devices have much less computational power and memory resources than classical PCs. Therefore, mobile-oriented applications should not be too complex. The paper proposed a client-server architecture for a highly asymmetric fractal compression scheme. The method implemented a computationally expensive compression on the server side. The fractal-coded images are decompressed with a low-complexity decoding scheme. The fractal-based compression utilizes local self-similarity in images of textures and natural scenes. Fractal image textures offer significant savings in both storage and transmission bandwidth. Future work will be focused on improvements of image quality and investigation regarding the hierarchical texture representation that offers direct decompression (Stachera and Rokita, 2006). Hierarchical texture compression is based on the block-wise approach characterized by a low computational complexity. Each block is subject to a local PIFS fractal transform and can be randomly accessed. This, along with new CLDC 1.1 profile-compliant GSM terminals, would facilitate more efficient image texture operations. Fractal compression/decompression could surpass the JPEG standard in mobile applications.

References

Atkinson S., Machin A., Graf M., Hageland M., Nashi A., Taylor R., Ayers D., Ray B. and Wiggers Ch. (2001): *Professional Java Mobile Programming*. — Chicago: Wrox Press Ltd., (R. Ashri, Ed.).

Baharav Z., Malah D. and Karnin E. (1993): *Hierarchical interpretation of fractal image coding and its application to fast decoding*. — Proc. Int. Conf. *Digital Signal Processing*, Levkosia, Cyprus, pp. 190–195.

Beers A., Agrawala M. and Chadda N. (1996): *Rendering for compressed textures*. — Proc. Int. Conf. *Computer Graphics and Interactive Techniques, SIGGRAPH*, New Orleans, USA, pp. 373–378.

Bury S. (2004): *Fractal imaging on mobile phones*. — M.Sc. thesis, University of Zielona Góra, Poland (in Polish).

Chen C. and Lee C. (2002): *A JPEG-like texture compression with adaptive quantization for 3D graphics application*. — The Visual Computer, Vol. 18, No. 1, pp. 29–40.

Cisar G. (1996): *On Entropy Coding Fisher's Fractal Quad Tree Code*. — Tech. Rep., Institut fur Informatik, University of Freiburg, Germany.

Condissi N., DiVerdi T. and Hoeller T. (2005): *Real-time rendering with wavelet-compressed multi-dimensional textures on the GPU*. — Tech. Rep. 2005.05, Comput. Sci.,University of California, Santa Barbara.

Delp E. and Mitchell O. (1979): *Image compression using block truncation coding*. — IEEE Trans. Commun., Vol. 2, No. 9, pp. 1335–1342.

Fisher Y. (1995): *Fractal Image Compression: Theory and Application*. — London: Springer.

Ghulam M, Falai C. and Zhangijn H. (2004): *Ternary wavelets and their applications to signal compression*. — Int. J. Appl. Math. Comput. Sci., Vol. 14, No. 2, pp. 233–240.

Knittel G., Shilling A., Kugler A. and Strasser W. (1996): *Hardware for superior texture performance*. — Comput. Graphics, Vol. 20, No. 4, pp. 475–481.

Kwon Y., Park I. and Kyung Ch. (2000): *Pyramid texture compression and decompression using interpolative vector quantization*. — Pro. Int. Conf. *Image Processing*, Vancouver, Canada, Vol. 2., pp. 89–106.

Malah D. and Sutskover I. (1999): *Hierarchical Fast Decoding of Fractal Image Representation Using Quadtree Partitioning*. — Technion, Israel: I.I.T.

Microsoft (1997): *Escalante hardware overview — Talisman*. — Graph. Multimedia Syst., Vol. 18, No. 1, pp. 89–106.

Nikiel S. and Moczulski M. (2006): *Image acquisition and segmentation on mobile devices*. — Proc. Nat. Conf. *Measurement Systems, SP*, Łagów, Poland, pp. 69–70, (in Polish).

Pazio M. and Cisowski K. (2005): *Application of colour image segmentation for localization and extraction text from images*. — Proc. Conf. *Poznań Telecommunication Workshops, PWT*, Ponań, Poland pp. 134–137.

Perebrin A. (1999): *Hierarchical approach to texture compression*. — Proc. Conf. *GRAPHICON*, San Francisco, USA, pp. 195–199.

Skarbek W. (1998): *Rough sets and current trends in computing*. — Proc. 1st Int. Conf. *Rough Sets and Current Trends in Computing, RSCTC*, Warsaw, Poland, pp. 441–453.

Stachera J. and Rokita P. (2006): *GPU-based hierarchical texture decompression*. — Proc. Int. Conf. *Eurographics*, Vienna, Austria, (on DVD).

Stachera J. and Nikiel S. (2004): *Fractal image compression for efficient texture mapping*. — Proc. Int. Conf. *Winter School on Computer Graphics, WSCG* Plzen, Czech Republic, pp. 169–172.

Received: 13 May 2006
Revised: 4 December 2006
Re-revised: 31 January 2007