An Energy-Efficient Image Representation for Secure Mobile Systems

Tim Woo, Catherine Gebotys, and Sagar Naik

Department of Electrical and Computer Engineering
University of Waterloo, Waterloo, Ontario, Canada
`timwoo@alumni.uwaterloo.ca, cgebotys@uwaterloo.ca, knaik@swen.uwaterloo.ca`

Abstract. In a mobile device, two major sources of energy consumption are energy used for computation and energy used for transmission. Computation energy can be reduced by minimizing the time spent on compression. Transmission energy and encryption energy can be reduced by sending a smaller image file that is obtained by compression. Image quality is often sacrificed in the compression process. Therefore, users should have the flexibility to control the image quality to determine whether such a tradeoff is acceptable. This paper proposes an energy efficient image representation system using Binary Space Partitioning (BSP) trees. Our experimental result shows that in most cases, our new tree construction and compression formula use only 40% of the original total energy required for compressing and sending images. BSP tree representation also allows partial encryption, which reduces total energy required for secure transmission by reducing the amount of data that needs to be encrypted. Our experimental results show that BSP tree representation is effective in reducing total energy for secure transmission for computer arts images compared to JPEG.

1 Introduction

The increasing popularity of wireless digital image communication presents a new challenge to image compression and encryption. First, portable devices run on a limited energy source such as batteries. Therefore, energy utilization must be efficient. From a software perspective, there are several strategies for optimizing energy [1]. These strategies focus on minimizing the energy cost for computations and memory accesses. For a wireless device, the total energy consumption consists of energy used for computations, data accesses, and transmission energy. For example, a typical CPU for PDA (whose name is not revealed to protect vendor identity) consumes about 90-150mW in running mode, 36mW in idle mode, and 0.9mW in sleeping mode. The transceiver consumes about 210-540mW for transmitting and 180-240mW for receiving data. Since the power used for computation and transmission is quite significant, this paper focuses on reducing the computation energy and transmission energy. To minimize computation energy, one can reduce the execution time of the compression algorithm, so that less energy is consumed in the switching activities in the processor when computing. To reduce transmission energy, one can reduce the amount of data to be sent, which is achieved by compression.

Although the study of image quality and execution time tradeoff is not a novel idea, it deserves special attention for a wireless device because of its limited computation power, memory and energy source. At the time of writing, a typical PC consists of a 2.8GHz processor, with 512MB RAM. A typical PDA (e.g. PalmOne Tungsten T3) consists of a 400MHz processor, with 64MB RAM, which is only a fraction of the computation power and memory available to a PC. Therefore, it is especially crucial for the image compression and encryption system to be optimized for a wireless PDA.

To provide flexibility and high compression performance, a Binary Space Partitioning (BSP) tree structure can be used to represent an image [2]. BSP tree representation is easy to construct, and it allows convenient pruning to different image quality levels. In a BSP tree, each non-terminal node is associated with a partition line and each leaf node is associated to a particular region of the image. The non-terminal nodes store the line parameters while the leaf nodes store the colour of the associated regions.

We have proposed a new tree construction technique by splitting the image into two equal halves each time instead of finding the optimal line. We have found in our experiments that the original Least-Square-Error (LSE) method in [3] is slow. We have also proposed a new pruning formula for compressing images. In BSP lossy compression, calculation of errors at each node is required to make pruning decisions. Our new formula aims to reduce computation times by reusing previous calculated values in previous iterations. Finally, we have introduced the concept of variable compression rate for different areas of the image using the advantages of BSP.

This paper is organized as follows: Sect. 2 discusses the relevant previous research. Sect. 3 describes the proposed BSP tree construction and pruning formula. Sect. 4 shows the experimental results by varying different parameters. Sect. 5 analyzes the new proposed scheme and discusses how our results compare to previous work, followed by a summary of the advantages and limitations of the new scheme and the concluding remarks in Sect. 6.

2 Related Work

There have been several studies on the effect of varying JPEG parameters on the image quality, latency, and energy. In [4], the authors have used a pre-computed lookup table to estimate compresssion costs. Research in [5] has studied the effect of varying JPEG parameters on the image quality. Although their approach does not require complete decompression to re-scale an image to different quality, it still requires re-encoding of entropy parameters, which consumes significant energy too. In contrast, our BSP tree compression scheme consumes only a small amount of energy when re-scaling because no re-computations are necessary when pruning the tree. JPEG encryption [6] has revealed that there are many image leakages when only part of the JPEG file is encrypted. For example, the image is still recognizable even if most of the AC and DC coefficients are encrypted.

A recent standard, JPEG2000, uses wavelet transforms to compress an image. In wavelet compression [7], the image first goes through a discrete wavelet transform (DWT) to generate wavelet coefficients. In [8], the wavelet coefficients are organized in pyramid levels according to importance, with each level adding additional details to the image. The resulting image consists of two parts: the zerotree structure and the wavelet coefficients.

Several energy efficient wavelet image compression schemes have been proposed. In [9],[10], the authors have proposed varying parameters such as the transform level, elimination level (EL), and the quantization level (QL). Again, the authors have used a lookup table that requires re-computation of quantization and entropy coding steps, which consumes fair amount of energy too. In [11], a distributed approach to wavelet compression has been proposed to distribute energy use in a wireless adhoc network evenly at different nodes. In such a system, the task for the wavelet transform, is processed in a distributed fashion to the various nodes in the adhoc network to save energy.

Quadtree compression [12] is similar to BSP except an image is split into four quadrants instead at each iteration. The advantage of quadtree compression is that tree construction is fast and simple. However, since each split creates four quadrants, it may result in a larger file size compared to BSP.

Many of the image compression algorithms are designed to aim for the maximum compression rate. However, these high compression algorithms may be too complex for energy constrained handheld devices. BSP scheme has advantages that other image compression techniques do not have. First, the hierarchical structure of the BSP tree allows convenient pruning of an image. This offers scalability and flexibility by dynamically adjusting the image quality according to the user need and environment to save transmission energy. In addition, since BSP compression operates in spatial domain, it is possible to compress using different thresholds in different regions of the same image. This allows selectively retaining the details in the important areas of an image, while the less important areas can be compressed more.

3 Our Image Representation Scheme

This section presents the proposed scalable energy efficient image representation scheme using BSP trees. We present the specifics of how the BSP tree representation of the highest quality image is constructed, and the specifics of how the energy, and image quality parameters are used in different modules of the system.

Fig. 1 shows the block diagram for our image compression system. The end user specifies his desired security levels and image quality for the various regions of the image (to be used for selective thresholding) to the system. From these user parameters, the test bench module generates the threshold for the compression and the encryption parameters. The compression module then compresses the highest quality image and the resulting image is immediately encrypted using Advanced Encryption Standard (AES) to generate an encrypted image. Only the defining BSP tree needs to be encrypted for security (not the colours defining each leaf node of the BSP tree).

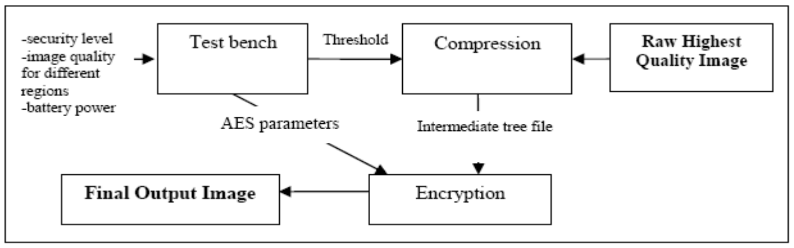


Fig. 1. Our Image Compression and Encryption System

In our proposed binary split method, the algorithm simply bisects the longest dimension of the region of interest in two equal halves. This speeds up the line selection process significantly, since it eliminates the expensive LSE transform computation in [3]. However, the trade-off is that the file size may not be optimal. In our method, for a region R with endpoints p_1 and p_2 and x_1 and x_2 as the x-coordinates (y-coordinates if it is a vertical line) of the two points, the midpoint between x_1 and x_2, i.e. x_C, is simply selected as the split line. The image is partitioned into smaller rectangular regions until the colour of the region is homogenous or until the region cannot be splitted further. Although this splitting method violates the property of optimal alignment of BSP tree partitioning lines with the actual object edges, it is found in our experiments that, in practice, this new method still performs satisfactorily.

To compress highest quality image file in BSP tree form, we can prune the BSP tree according to the pruning threshold, which is expressed as a fraction relative to the root node's error in our experiments. This process reduces the file size by sacrificing the image quality. There exist some advanced pruning techniques such as [13]. For simplicity, we focus our work using a simple error based pruning criterion.

Before compression, the error associated with each node in the tree must be calculated. We now introduce the concept of pruning. Pruning deletes and merges nodes in a BSP tree if the error is smaller than a user specified threshold. In the best quality image, the leaf nodes store the exact colours of the original image. When pruning this best quality tree, these leaf nodes are merged to form new leaf nodes if the merged node's error is below the pruning threshold, which is expressed as a fraction of the root node's error in our work. After these leaf nodes are merged, the mean colour value of the pixels in the region is used to approximate the new leaf node's colour. First, users will specify a relative threshold divisor value. Then, this value is multiplied by the root node's error value to generate the absolute threshold value that is used for pruning decision.

Traditionally, error for the region is calculated using the total square error. However, in our experiment, it is found that this formula is slow. Therefore, we have proposed a new error calculation formula for image pruning. Our formula calculates error in a bottom-up manner and speeds up computation by reusing previously calculated error values. First, at each iteration i, two parameters, σ_1 and σ_2, are defined:

$$\begin{aligned} \sigma_1 &= \begin{cases} |m_L - m_i| & \text{if left child of the current node is a leaf;} \\ \sigma_L & \quad \text{otherwise} \end{cases} \\ \sigma_2 &= \begin{cases} |m_R - m_i| & \text{if right child of the current node is a leaf;} \\ \sigma_R & \quad \text{otherwise} \end{cases} \end{aligned} \tag{1}$$

where m_L and m_R are the mean values of the left and right child, respectively, σ_L and σ_R are the previously calculated values of σ for the left and right child, respectively and m_i is the mean value of the current node at iteration i. The current node's standard deviation σ_i is calculated by:

$$\sigma_i = \begin{cases} 0 & \text{if left child of the current node is a leaf;} \\ \sqrt{\frac{A_L \sigma_1^2 + A_R \sigma_2^2}{A_L + A_R}} & \quad \text{otherwise} \end{cases} \tag{2}$$

where A_L and A_R are the area of the left and right child, respectively. Finally, the node's total error E_i is calculated by this formula:

$$E_i = \begin{cases} A_L\,|m_L + \sigma_1 - m_i| + A_R\,|m_i - (m_R - \sigma_2)| & \text{if } m_L \geq m_i; \\ A_R\,|m_R + \sigma_2 - m_i| + A_L\,|m_i - (m_L - \sigma_1)| & \text{otherwise} \end{cases} \tag{3}$$

After the image is compressed by BSP tree algorithm, the image goes to encryption stage. In the encryption module, only the BSP tree part needs to be encrypted. Unlike JPEG, where the entire image must be encrypted for secure transmission, BSP tree allows partial encryption while still keeping the image secure. This saves encryption computation energy cost.

4 Experimental Results

The experimental results of varying different parameters in generating an image will be presented in this section. We will investigate the effect of the tree construction mechanism on the computation energy and compression bit rate. We will also investigate how the error formula used for pruning affects the image quality and compression ratios. The computation energy is estimated by multiplying the execution time by the estimated average CPU power dissipation during execution. Due to limited resources, the execution time is measured on a PC, and the time on PDA is estimated by scaling down by a factor of seven (since a typical PC is 2.8GHz and a typical PDA CPU is 400MHz). Then the result is multiplied by 120mW, the average CPU power consumption during computation. The transmission time is estimated from the file size of the image multiplied by the average data rate of 600kbit/s for a typical 3G system [14]. Then the transmission energy is estimated by multiplying the estimated transmission time by 375mW, the average measured transmission power consumption of the transceiver for our PDA.

4.1 Effect of Tree Construction Mechanism

The tree construction method affects how fast the best quality tree representation of image can be generated. In addition, the shape of the tree affects the pruning characteristics. This section presents the resulting images for the optimal line selection method and the binary split method.

Table 1 shows the computation energy E_{tree} used for tree construction and the transmission energy E_{Tx} for both the original LSE optimal line and our binary split method. It can be seen that our method only uses about 40% of the original total energy required for constructing and sending the image. Most of the savings comes from the savings in the computation energy for tree construction. Although the images generated by our method use slightly more transmission energy because the generated file is slightly larger, it is compensated by the much more significant savings in computation.

Table 1. Tree Construction and Transmission Energy Consumption (Joules) for Various Images. The total energy is the sum of the computation energy required for constructing the tree and the transmission energy for sending the image

	Optimal Line			Binary Split			$\frac{E_{\text{Total}_\text{B}}}{E_{\text{Total}_\text{O}}}$
	E_{tree}	E_{Tx}	$E_{\text{Total}_\text{O}}$	E_{tree}	E_{Tx}	$E_{\text{Total}_\text{B}}$	
peppers (512×512)	11.86	5.23	17.09	1.30	5.58	6.88	0.40
lena_small (256×256)	2.74	1.33	4.07	0.30	1.37	1.67	0.41
tulips (768×512)	17.47	7.96	25.43	1.95	8.46	10.41	0.40
frymire (512×512)	8.85	1.46	10.31	0.80	3.32	4.12	0.40
serrano (512×512)	9.67	1.12	10.79	0.66	2.33	2.99	0.27

4.2 Effect of Error Calculation Formula on Pruning

In this experiment, we will explore how our error pruning formula performs on the optimal line constructed and our binary split constructed tree. We will explore the effects of the tree construction method on the results generated by our pruning formula. The proposed fast error formula in (3) is used to prune optimal-line tree and binary-split tree images at a threshold of 4.00×10^{-4}. The resulting peppers and frymire images are shown in Fig. 2.

With respect to image quality, it is found that our fast error formula works slightly better with binary split generated trees. The optimal line generated

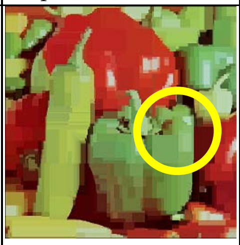
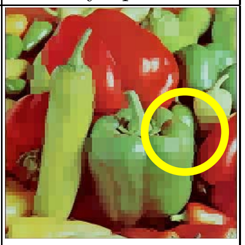
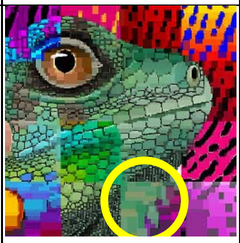
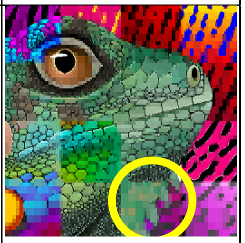
Threshold	Optimal Line Tree	Binary Split Tree
4.00×10^{-4}		
4.00×10^{-4}		

Fig. 2. Peppers and Frymire Images Using Fast Error Pruning. For peppers image, our fast error formula works better images trees constructed with binary split method, since the tree is more balanced. For frymire, our pruning formula works equally well with image trees constructed with either the optimal line or our binary split method

image's quality degrades quickly as the threshold increases. For example, at a threshold of 5×10^{-6} of the root node's error, the bit rate decreases quickly to 3.49bbp at 34.54dB, while the binary split tree has a slower drop in bit rate (8.82bbp) and image quality (37.73dB). The sharp drop in bit rate and image quality for optimal line trees is due to the observation that these trees are usually unbalanced. Therefore, the effect of error propagation is worse for an unbalanced tree where some paths to the root may be very long. As a result, the root's total error calculated using our formula deviates from the actual total square error more. Therefore, our formula performs better for our binary split constructed trees.

The estimated computation energy consumed in computing the errors at each node is shown in Table 2. For both the peppers and frymire images, the computation energy is reduced by almost 90%. Also, the slow total square error (TSE) method works slightly better on binary split tree version than the optimal line tree version of peppers. Again, this is due to the unbalanced tree shape of optimal line trees. Therefore, a lot of the calculations done at the deeper levels need to be performed again. For the frymire image, the behaviour is opposite. The binary split tree version is slower. This is because for the frymire image, the best quality image actually has a lot more nodes than the optimal line version.

Although the optimal line selection algorithm yields an optimal size image tree, it is too computationally intensive; therefore it is not optimal for energy. On the other hand, the binary split method can construct the BSP tree much faster,

Table 2. Computation Energy Consumption (Joules) for Computing Node Error

	Optimal Line Tree		Binary Split Tree	
	TSE	Our Error Formula	TSE	Our Error Formula
Peppers	4.99	0.50	3.77	0.53
Frymire	1.08	0.14	2.19	0.33

with a slight increase in file size. Moreover, calculating the total square error in the traditional way is too slow. Our proposed a fast error formula yields satisfactory results when pruning images, especially for binary split method constructed images.

4.3 Selective Focus Regions Feature

One of the major advantages of a BSP tree is the feature of selectively choosing important areas of the image to have a higher quality than the less important parts. The ”importance” of a region is supplied as user parameters. For the less important areas, one can choose a higher threshold to compress the image more, while in the more important areas one can choose a lower threshold to keep a higher quality image. For example, in Fig. 3a, the lena image is pruned with the same threshold for the entire picture. The facial features and the background are pruned with the same quality. As a result, the facial features are blurred (highlighted with circles). On the other hand, in Fig. 3b, the facial region is pruned with a lower threshold, while the less important background is pruned with a higher threshold. One can see that Fig. 3b appears to have a higher quality than Fig. 3a because the important facial details are conserved. Selective focus region feature should theoretically work well for quadtrees too, although it has not been suggested in previous research in [12].

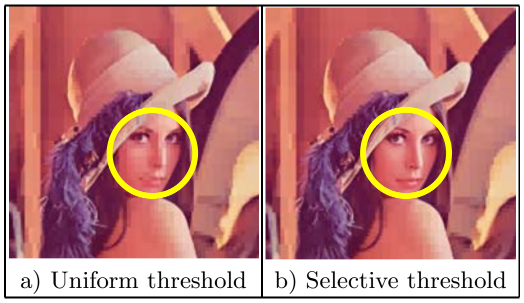


Fig. 3. Lena Image with Selective Threshold Regions. It can be seen that with selective threshold to preserve the facial details, the lena image is visually higher quality than the same image with uniform threshold across the image

5 Discussion

This section presents the analysis for the various characteristics of BSP tree representation. We will analyze the performance of the tree construction methods and pruning formula, image quality with respect to execution time and file size, and the encryption performance with respect to file size and block or key length.

In [3], LSE is calculated for every potential partitioning line. The minimum total number of operations required for constructing a optimal line selection BSP tree for a $N \times N$ image is $MinTotalOps = 2\log_2 N \cdot (3N^2 \cdot Multiplications + 5N^2 \cdot Additions)$. This is only the lower bound of the total number of operations required for the optimal line selection process. Usually, more than one line passes the LSE transform condition. Therefore, the number of actual computations is significantly greater.

In the binary split case, for each split, only one addition and one multiplication is needed to find the midpoint. Thus, only $N^2 - 1$ additions and $N^2 - 1$ multiplications are required. No calculation of total square error is needed. It can be seen that the binary split method is much faster than the optimal line selection method. From the experimental results, it can be seen that the file size increase due to neglecting the optimal line is acceptable. Therefore, the binary split method is a simple yet efficient method to construct the BSP tree. It has the combined benefits of the efficiency of representation of a BSP tree and the efficiency in tree construction of a quadtree.

In [3], TSE needs to be calculated at every level of the tree. This may yield to better pruning decisions, yet it is not efficient since this calculation needs to be repeated again at the higher levels. Our proposed formula reuses the partial error previously calculated from the lower nodes. This speeds up the pruning exponentially. Calculation of the total square error requires N^2. In contrast, our proposed formula in (3) reuses the children's total error and derives the new total error by adding a weighted average of the children node's error and standard deviation. No recalculation of error for the subtree regions is needed.

Our proposed algorithm aims to reduce energy consumption by cutting down the computation energy for constructing the image tree and pruning the tree. In the new binary split tree construction method, the tree construction energy is cut down significantly, at the expense of a slight increase in tree file size and transmission energy. In the fast error calculation method, the computation energy is reduced by reusing results from previous calculations. The tradeoff is a slightly less accurate pruned image quality.

In general, the higher the image quality, the larger the file size is. Table 3 shows the file size, bit rates and the PSNR for the peppers and frymire images compared to JPEG. The BSP tree with the bit rate closest to the best compression (denoted BC) JPEG is chosen for comparison. The optimal tree peppers image is pruned at 10^{-5} threshold using the original TSE formula. The best quality image (denoted BQ) chosen for comparison with the JPEG is also generated by optimal tree method.

Table 3. BSP Tree and JPEG Transmission Energy and PSNR

	JPEG size (bytes)	$E_{\text{Tx}_{\text{JPEG}}}$ (Joules)	PSNR (dB)	BSP size (bytes)	$E_{\text{Tx}_{\text{BSP}}}$ (Joules)	PSNR (dB)
Peppers BC	23308	0.117	32.19	34020	0.170	25.29
Peppers BQ	249787	1.249	44.35	1116072	5.580	56.53
Frymire BC	88012	0.440	21.13	86006	0.433	20.39
Frymire BQ	475277	2.376	47.49	200632	1.014	49.66

The BSP tree with the bit rate closest to the best compression JPEG is chosen for comparison. The optimal tree frymire image pruned at 10^{-4} threshold. The best quality image for comparison with the JPEG is the optimal tree image.

It can be seen that BSP tree compression works better with the frymire image than the peppers image. For the frymire image, the BSP tree compression can compress more than 2 times better at the best quality (BQ) settings in Table 3. Even if we set JPEG at the best compression (BC) settings, BSP tree compression is still able to match the compression ratio of JPEG. Table 4 illustrates the estimated total energy required for encrypting and transmitting the JPEG and BSP images. For JPEG it is assumed that the entire JPEG file is encrypted to avoid security leaks. In the BSP case only the defining BSP tree is encrypted as described in Sect. 3. It can be seen that our BSP approach uses only a fraction of the encryption energy, denoted E_{AES}, compared to JPEG. When sending the frymire image, our BSP approach is better than JPEG, and it uses only 38-87% of the total energy required for encrypting and transmitting an equivalent JPEG image. However, it uses more energy than JPEG for the peppers image. From our experimental results, it appears that BSP trees works better for computer graphics images and JPEG is more suitable for natural photographic images.

Table 4. Estimated BSP Tree and JPEG Secure Transmission Energy (Joules)

	JPEG			BSP Tree			$\frac{E_{\text{Total}_{\text{BSP}}}}{E_{\text{Total}_{\text{JPEG}}}}$
	E_{AES}	E_{Tx}	$E_{\text{Total}_{\text{JPEG}}}$	E_{AES}	E_{Tx}	$E_{\text{Total}_{\text{BSP}}}$	
Peppers BC	0.015	0.117	0.132	0.001	0.170	0.17	129%
Peppers BQ	0.164	1.249	1.413	0.041	5.580	5.621	398%
Frymire BC	0.058	0.440	0.498	0.003	0.430	0.433	87%
Frymire BQ	0.311	2.376	2.689	0.011	1.003	1.014	38%

We have compared the bit rate with other compression systems. For the data based on 8-bit greyscale images, we have multiplied the bit rate by 3 for a fair comparison to the 24-bit colour used in this system. Table 5 shows the comparison to other image compression systems. The values in parentheses are the greyscale values multiplied by 3. Our BSP tree compression system is comparable to other systems' performance for the peppers image. However, for the lena image, other systems work better. Unfortunately, we are unable to find the data for other images from the literature.

Table 5. Compression performance compared to other systems

	Quadtree		JPEG		Fractal Binary Tree		BSP Tree	
	Bit rate	PSNR	Bit rate	PSNR	Bit Rate	PSNR	Bit Rate	PSNR
peppers	0.516	N/A	N/A	N/A	0.60 (1.80)	32.5	1.67	31.37
lena	0.67 (2.01)	30.36	0.125 (0.375)	26.2	0.24 (0.73)	31.0	0.73	26.57
frymire	N/A	N/A	1.813	47.49	N/A	N/A	0.765	49.66

Our proposed BSP compression algorithm is faster than [3] because it reuses calculations at previous nodes. With respect to the compression performance and based upon our experimental results, BSP trees are better suited for computer graphics images than photographic images when compared to JPEG. This is because BSP compresses better when there are larger homogenous regions, which is typical for computer graphics. In contrast, photographic images have smooth transitions between pixels. Therefore, neighbouring pixels are usually slightly different from each other. JPEG is more specialized in handling such cases. Our BSP scheme is also better than quadtrees in most cases because partitioning lines are more flexible, thus less nodes are created to represent the same image.

Finally, this paper has proposed the idea of allowing users to select a different pruning threshold for different regions using a BSP tree. This allows users to conserve the details in the important parts of the image. After all, it is the end users' perception of the image that is ultimately important. Compression algorithms that affect the image globally (e.g. JPEG and wavelets) do not allow such flexibility. We have proposed a binary split scheme in building the BSP tree, which combines the benefits of both worlds: it has the computation efficiency of quadtree in splitting the image, but also has the compact storage property of an optimal line BSP tree.

6 Conclusion

In this paper, we have presented an energy efficient image representation scheme. For mobile systems, a fast compression algorithm is necessary to save computation costs and the energy cost used for transmission. Our binary split tree construction scheme combines the benefits of both quadtree and optimal line BSP tree: it has the execution efficiency of quadtree in splitting the image, but also has the compact storage property of an optimal line BSP tree. Our experiment results show that the file size, hence, transmission energy, only increases slightly when using a binary split scheme. Our BSP tree pruning formula uses less energy than the traditional total square error formula because it reduces computations by reusing results from previous calculations. With respect to compression performance, we have found that BSP tree compresses computer graphics better than photographic images. Since computer graphics have large homogenous regions, our BSP compression algorithm is effective in merging those neighbouring pixels together. Finally, our image representation allows users to choose different pruning thresholds for different areas of the image. By choosing a higher threshold for the less important areas, while maintaining a lower threshold for the important areas, the impact on the perception of image quality is small. At the same time, transmission energy is reduced because the image file size is smaller.

The authors would like to thank A. Sung for his power measurements.

References

1. Naik, S., Wei, D.: Software implementation strategies for power-conscious systems. ACM Mobile Networks and Applications **6** (2001) 291–305
2. Radha, H., Vetterli, M., Leoonardi, R.: Image compression using binary space partitioning trees. IEEE Trans. on Image Processing **5** (1996) 1610–1624
3. Radha, H., Vetterli, M., Leoonardi, R.: Fast piecewise constant approximation of images. SPIE Visual Comm. and Image Processing **1605** (1991) 475–486
4. Chandra, S., Ellis, C.: JPEG compression metric as a quality aware image transcoding. In: Proc. of the 2nd USENIX Symposium on Internet Technologies and Systems, Boulder, CO (1999) 81–92
5. Taylor, C., Dey, S., Panigrahi, D.: Energy/Latency/Image quality tradeoffs in enabling mobile multimedia communication. Software Radio - Technologies and Services (2001) 55–66
6. Kailasanathan, C., Safavi Naini, R.: Compression performance of JPEG encryption scheme. In: IEEE Intl Conf. on Digital Signal Processing. Volume 2. (2002) 1329–1332
7. Shapiro, J.M.: Embedded image coding using zerotrees of wavelet coefficients. IEEE Trans. on Image Processing **41** (1993) 3445–3462
8. Said, A., Pearlman, W.A.: A new, fast, and efficient image codec based on set partitioning in hierarchical trees. IEEE Trans. on Circuits and Systems for Video Technology **6** (1996) 243–250
9. Lee, D., Dey, S.: Adaptive and energy efficient wavelet image compression for mobile multimedia data services. In: Proc. of Intl Conf. on Communications. Volume 4., New York (2002) 2484–2490
10. D.G.Lee, D.Panigrahi, S.Dey: Network-aware image data shaping for low-latency and energy-efficient data services over the Palm wireless network. World Wireless Congress (3G Wireless) (2003)
11. Wu, H., Abouzeid, A.: Energy efficient distributed JPEG2000 image compression in multihop wireless networks. In: Proc. of the 4th Workshop on Applications and Services in Wireless Networks, Boston, MA (2004)
12. Cheng, H., Li, X.: Partial encryption of compressed images and videos. IEEE Trans. on Signal Processing **48** (2000) 2439–2451
13. Chou, P., Lookabaugh, T., Gray, R.: Optimal pruning with applications to tree-structured source coding and modeling. IEEE Trans. on Information Theory **35** (1989) 299–315
14. Eyuboglu, V.: CDMA2000 1xEV-DO delivers 3G Wireless (2002) Network World, Feb 25th 2002, http://www.nwfusion.com/news/tech/2002/0225tech.html.