The International Journal of Virtual Reality, 2006, 5(3):9-15

Multiresolution Graphics on Ubiquitous Displays using Wavelets

Fan Wu, Emmanuel Agu and Matthew Ward

***Abstract*— Large meshes or images cannot be rendered at full resolution on mobile devices such as cell phones, PDAs and laptops since these devices have limited storage, CPU, memory, display, and battery power. Wavelet-based multiresolution analysis can represent meshes and graphics input at multiple Levels of Detail (LODs). We propose UbiWave, a wavelet-based framework for scalable transmission of large meshes and graphics content to heterogeneous ubiquitous computing devices. In UbiWave, a base representation and different levels of wavelet coefficients are pre-generated at the server. An optimal LOD level is transmitted to each mobile client based on its spacification and wireless channel conditions, where the corresponding LOD is reconstructed. To save scarce resources on mobile devices, we render graphics content at the lowest LOD that does not show visual artifacts, called the point of indiscernability (PoI). By rendering content at the PoI instead of the highest resolution, we are able save 61% decode time and 45% energy usage on the client.**

***Index Terms*— multiresolution rendering, ubiquitous graphics, wavelets.**

I. INTRODUCTION

Advances in ubiquitous displays and wireless communications has fueled the emergence of exciting mobile graphics applications including 3D virtual product catalogs, 3D repair manuals, security monitoring systems, 3D maps, telesurgery and mobile games. Mobility makes graphics applications more convenient and increases the productivity of on-site consultants, such as architects who can discuss preliminary 3D designs. Mobile graphics applications also possess desirable qualities of computer graphics such as interactivity and enabling users to visualize imaginary situations.

Current trends in using cameras to capture geometry, material reflectance and other graphics elements means that very high resolution inputs is accessible to render extremely photorealistic scenes. However, since mobile devices and wireless networks have limited resources, scaling of high-resolution content and conversion to a format that is suitable for rendering, is usually required. This scaling and conversion process is currently manual, time-consuming, involves trial-and-error to determine the best resolution for each type of computing device and increases the production costs of many graphics applications.

We propose UbiWave, an end-to-end framework for scalable rendering of graphics content on heterogeneous ubiquitous computing devices. UbiWave uses wavelets as a common representation for all geometry, textures images, material reflectance and other graphics content. Wavelets, which originated from the work of Fourier, are a mathematical tool that can represent input functions at multiple resolutions [12]. Using wavelets, we can render graphics content at scales suitable for tiny devices as small as cell phones and headmounted displays, or as large as large tiled displays. In UbiWave, our philosophy is to save scarce resources by rendering at the lowest resolution that does not show visual artifacts, called the Point of Indiscernability (PoI). By rendering content at the PoI instead of the highest resolution, we are able to save 61% decode time and 45% energy usage on the client.

The rest of the paper is as follows. Section II provides some background on wavelets, Section III describes UbiWave, our system for ubiquitous graphics, Section III-A describes our current UbiWave prototype, Section IV evaluates the performance of our UbiWave current prototype, Section V describes sample resource savings by using UbiWave and Section VI discusses related work, and Section VII is the conclusion and future work.

II. BACKGROUND ON WAVELETS

Overview: Wavelets [12] can *decompose* input functions to yield a coarse (rough) base function, plus a tree of detail coefficients. *Reconstructing* the original function starts from the coarse base function. Its resolution is then successively improved by adding more levels of the detail coefficient tree. In UbiWave, our system for ubiquitous graphics all rendering inputs such as meshes [6, 24, 34], textures [4] and material reflectance properties [19, 31] are converted and distributed as decomposed wavelets (base + coefficient tree) to facilitate scalable rendering on heterogeneous computing devices even when inputs are extremely large captured files. Wavelets have been used in a wide range of applications including graphics and image processing, ray tracing [5], information retrieval [28], FBI fingerprint storage [3], and geographic modeling. Today, published work has shown that almost all aspects of a graphics scene can be decomposed using wavelets including meshes, textures, material and reflectance properties. If captured content is available as decomposed wavelets,

Manuscript Received on September 25, 2006.

Fan Wu, Emmanuel Agu and Matthew Ward are with Computer Science Dept, Worcester Polytechnic Institute, Worcester, MA 01609, USA, E-mail: wuf, emmanuel, matty@cs.wpi.edu

heterogeneous mobile devices can retrieve resolutions suitable for their use. Wavelets achieve aggressive compression, which is also useful for low cellular network bandwidths. Wavelets also support progressive refinement since users can view the increasingly improved intermediate results after coefficients. Finally, using wavelets for graphics content facilitates integration of emerging mobile graphics standards with existing MPEG4 video and JPEG2000 image standards, where wavelets are already used.

Wavelets in computer graphics: Schroeder [32] was one of the first to use wavelets in computer graphics and used wavelets to compress geometric and evaluate global illumination rendering equations.

- *Meshes:* Lounsbery proposed wavelet-based 3D compression [6, 24] by applying wavelet transforms to an arbitrary 3D mesh at several detail levels. During wavelet decomposition of meshes, a mesh is subdivided and deformed to make it fit the surface to be approximated. The original high resolution mesh is processed to generate a base connectivity file along with a sequence of smooth and detail coefficients that express the difference between successive levels of detail. Reconstruction starts with the base mesh. As more wavelet coefficients are included, a higher resolution mesh will be rendered. These steps can be repeated at the required resolution levels. A hierarchy of meshes is obtained from the simplest one M0, called base mesh, to the original mesh M ∞ . The wavelet transform of meshes removes a large amount of correlation between neighboring vertices. This hierarchy of meshes at different resolutions is the basis of multiresolution analysis [24]. To make the mesh approximation M^{j-1}as close as possible to the original mesh M^j, the lifting scheme [33] is used. Valette's scheme [34] tries to convert the connectivity simplification to 1:4 subdivision as much as possible. If 4:1 simplification is not possible, other groups of three or two faces are chosen, or some faces are left unchanged. Several methods for performing wavelet transforms on meshes are based on interpolating subdivision schemes such as the Butterfly [8] that defines both interpolating and smoothing parts. The Loop [23] wavelet transform is an approximating scheme that has the advantage that the inverse transform uses Loop subdivision and produces the smoothest surfaces. After wavelet decomposition, adaptive arithmetic coding is often used to compress the size of the transmitted mesh and coefficients. Some alternative vertex-based simplification algorithms that do not use wavelets have also been proposed including vertex decimation [32], edge contraction [14], and valence-based simplification [7].
- *Textures and images:* Techniques to compress images and textures using wavelets have also been proposed. A 2D wavelet transform that can be obtained by a separable decomposition in the horizontal and vertical directions [20]. Image compression based on the Discrete Wavelet Transform (DWT) is used in the JPEG2000 image standard [4]. Wavelet decomposition of textures and images is slightly different from that of meshes. In a preprocessing step, a nonstandard 2-D Haar wavelet decomposition of images is performed, which involves one step of horizontal pair-wise averaging and differencing on the pixel values in each row of the image, followed by applying vertical pair-wise averaging and differencing to each column of the result.
- Material reflectance and BRDFs: Wavelets have been used to represent material reflectance or Bi-directional Reflectance Functions (BRDFs). In [31], reflections were encoded from one incident direction using a spherical wavelet representation, which can represent a slice of the BRDF with several hundreds of coefficients. [19] extended this work and represents 4D reflectance functions using 4D basis wavelet functions stored in a compact wavelet coefficient tree that keeps only the highest coefficients to reconstruct the BRDF and thresholding the rest to zero.

III. UBIWAVE

UbiWave, our system for scalable rendering on heterogeneous computing devices encompasses all stages including storage of graphics content as wavelets, transmission and rendering on the mobile device. This wavelet-based framework enables small mobile devices such as cell phones to render extremely large captured content that can be many gigabytes in size. Fig, 1 is an overview of our proposed approach. Captured content is encoded using wavelets (on the left of figure). When retrieved, the content is dynamically tailored to the resources of a mobile device and wireless network, transmitted wirelessly to the mobile device where it is rendered (right of figure). The realism of rendering on the mobile device can be varied to accommodate mobile device constraints on screen size and battery power. Essentially, small devices such as cell phones on a GPRS cellular data network and laptop on a broadband WiMax network, can render the same scene, access the same rendering inputs from the same content databases, but automatically achieve the best resolutions for their configuration. The framework can be used in several ways. It can be programmed into a software tool that downloaders of scanned content can use offline. In a more ambitious scenario, the quality of rendered images in mobile graphics applications would be varied dynamically based on available resources. For instance, the geometry of rendered objects and the quality of shading of a mobile flight simulator could be gracefully degraded as the device's battery dies.

Novel algorithms in UbiWave To achieve our end-to-end vision in UbiWave, we are developing several novel algorithms. The shaded boxes in Fig. 1 are novel algorithms and techniques in UbiWave that are now summarized.

- *Out-of-core wavelet decomposition of gigantic meshes:* Wavelet decomposition algorithms require that the entire input file is loaded into computer memory. State-of-the-art meshes can be so gigantic that they cannot fit into the memory of most computers. Proposed out-of-core simplification algorithms [27] break gigantic meshes into segments that can be read in, simplified, and written out to

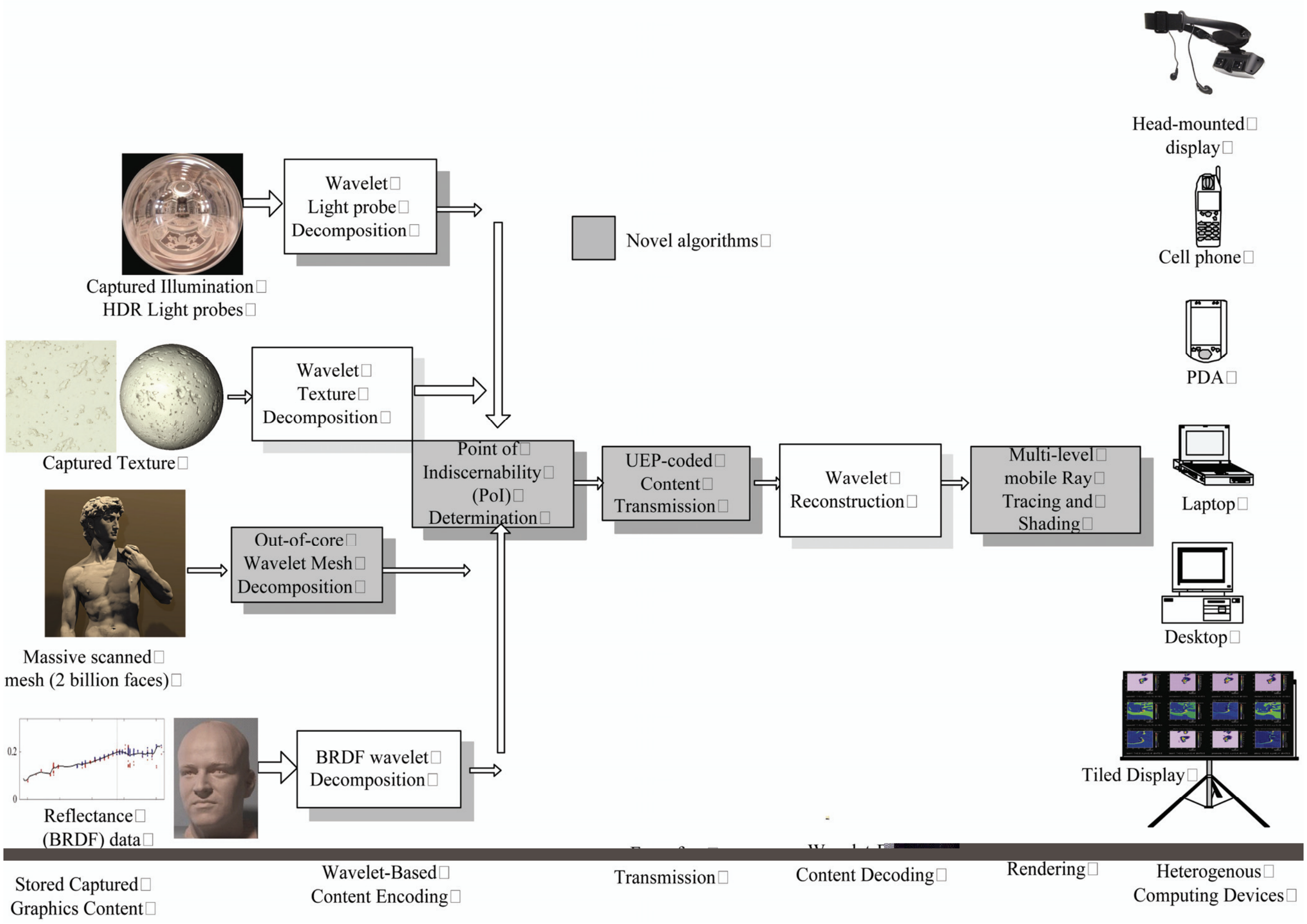

Captured Illumination
HDR Light probes
Wavelet Light probe Decomposition
Captured Texture
Wavelet Texture Decomposition
Massive scanned mesh (2 billion faces)
Out-of-core Wavelet Mesh Decomposition
Reflectance (BRDF) data
BRDF wavelet Decomposition
Novel algorithms
Point of Indiscernability (PoI) Determination
UEP-coded Content Transmission
Wavelet Reconstruction
Multi-level mobile Ray Tracing and Shading
Head-mounted display
Cell phone
PDA
Laptop
Desktop
Tiled Display
Stored Captured Graphics Content
Wavelet-Based Content Encoding
Transmission
Content Decoding
Rendering
Heterogenous Computing Devices

Fig. 1. UbiWave: Our wavelets-based ubiquitous graphics system

disk, but did not use wavelets. We are developing a novel out-of-core algorithm for wavelet decomposition of gigantic meshes.

- *PoI metric for different mobile display sizes:* Wavelets allow us to represent graphics content at a continuum of resolutions. A key question is: *How do we determine the optimal resolution for a particular mobile device?*

  To save scarce mobile resources, we propose rendering at the lowest level-of-detail that does not show visual artifacts, called the *Point of Indispensability (PoI)*.We have developed a metric to compute the PoI based on a mobile display's size.
- *Unequal Error Protection (UEP) for Wireless Transmission of Wavelet Encoded Meshes:* We are also developing an error-coding scheme that adds redundant bits to wavelet-encoded meshes before transmission such that minor errors can be corrected by the receiver. Our scheme is based on Unequal Error Protection (UEP) [1], in which the number of redundant bits allotted to each part of the mesh is proportional to the amount of information it contains: more bits are added to parts with more information. More details on our UEP scheme can be found in [36].
- *Multi-level mobile raytracing and shading*: Rendering algorithms currently do not adapt to the available resources and heterogenity of mobile devices. Ray tracing [35] is a popular graphics rendering algorithms. To make ray tracing more mobile-friendly, we propose 1) Ray tracing at a level of realism that is just adequate for each type of mobile device and 2) Making ray tracing more energy efficient.

UbiWave benefits and impact

- *Increased productivity:* Captured content will be more accessible to many heterogenous devices with minimal effort, speeding up prototyping of mobile graphics applications. Groups that spend months capturing content would just need a few extra hours to run software that converts captured content to a pre-agreed wavelet representation. Our envisioned framework takes the wavelet-encoded content as input and virtually eliminates manual processes currently required to scale and size graphics content for a target device.
- *Uniform networking:* If all graphics content is converted to wavelets, the wireless network always transmits base representations and trees of coefficients. Networking for graphics can be optimized around this structure. For instance, we propose an error encoding scheme that

improves the resilience of meshes that have been decomposed using wavelets. Potentially, all wavelet-encoded content can use this scheme.

- *Enabling real-time resource adaptations:* Ultimately, we would like to directly connect the mobile devices with repositories of captured content and vary the rendering quality of graphics on mobile devices in real time as resource availability changes.

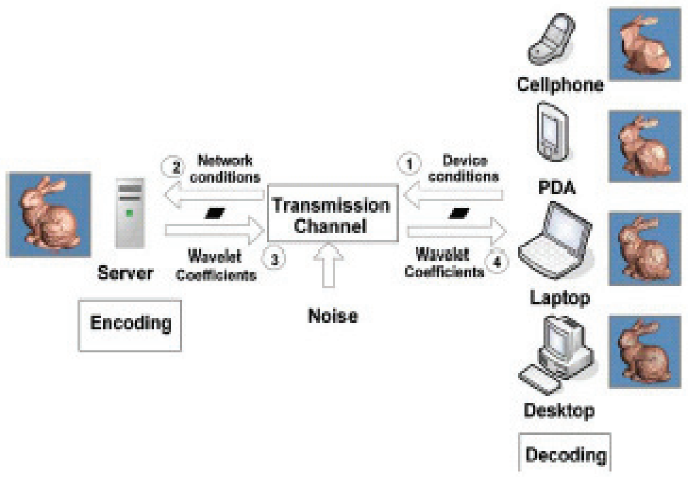


Fig.2. UbiWave system framework

- *Broad impact on related resource-constrained systems:* Our results apply to many resource-constrained graphics systems, including augmented reality systems [29], web-based graphics systems [30], pervasive graphics interfaces [11], as well as scaling up to large tiled displays [16].

A. UbiWave System Prototype

In this section, our current prototype of UbiWave is described. Fig. 2 is an overview of our implemented prototype. Our initial prototype focused on scalable mesh transmission. The meshes encoded using wavelets, transmitted and rendered on a mobile device. Simple lighting and minimal color is used. The system works using the following three steps:

1) *Mesh preprocessing:* The server processes the original high resolution mesh or image to generate the base connectivity file and coefficient files for different levels of detail. Our work with meshes on UbiWave was based on the Loop wavelet transform.
2) *Receiving mobile parameters:* The mobile device transmits certain parameters to the server, so that the server can decide what LOD to transmit to a given mobile device. The transmitted mobile parameters include two parts: mobile device specification and wireless channel conditions. The mobile device specification includes its screen size, CPU, memory and battery power. The wireless channel parameters include measured bandwidth and error rate measured in the area around the mobile device.
3) *Server decision on what wavelet LOD to send:* After server receives the mobile parameters, it decides which level of wavelet coefficients will be sent to the mobile device. Typically, this decision is based on the received mobile device parameters, and can be expressed in the general form:

$$f(\mathit{CPU},\ \mathit{memory},\ \mathit{power},\ \mathit{screen\ size},\ \mathit{bandwidth},\ \mathit{error\ rate}\ldots)=\mathit{level\ of\ coefficients} \qquad (1)$$

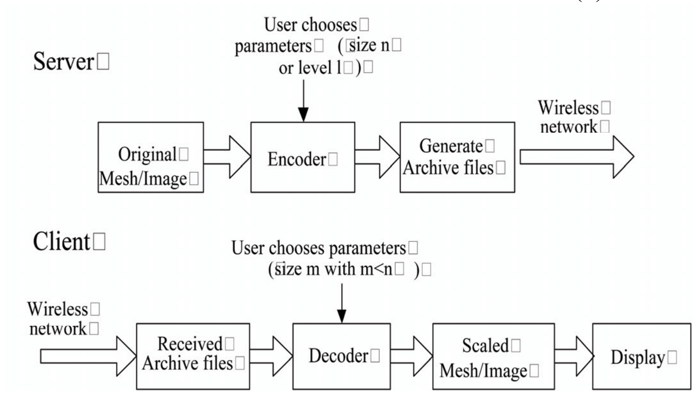


Fig.3. UbiWave prototype implementation

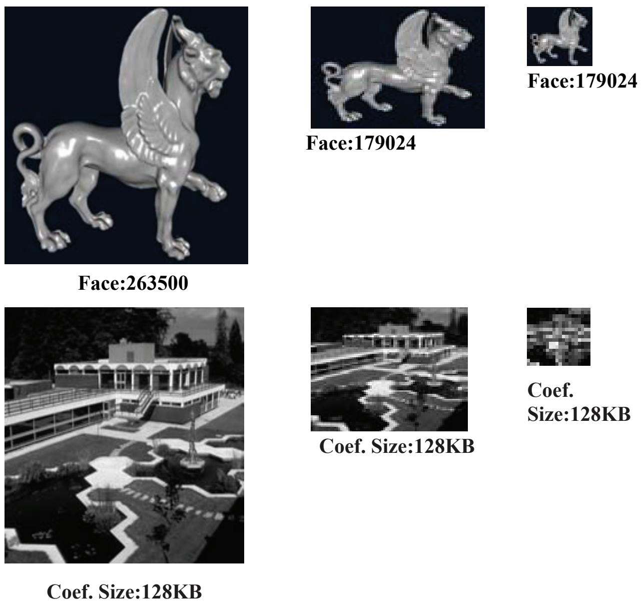


Fig. 4. Sample meshes and images on our prototype

The server only needs to send basic mesh connectivity information and corresponding wavelet coefficients to mobile devices, saving bandwidth and memory. While our complete server decision heuristic is based on multiple factors, we currently use the PoI metric described above, which tries to select the lowest acceptable LOD for any given mobile device based on its screen size.

For images and textures, a 2D Haar wavelet decomposition and biorthogonal filters with filter lengths (9,7) are used. The system figure as shown in Fig. 3. The usage of parameters n or l at the server and parameter m at the client are the same as for the meshes.

At the server, the original mesh (or image) is encoded into different levels of coefficients with the control parameters n or *l*, where *n* is the size of the compressed file and *l* is the level of coefficients. At the client, the user can also choose parameter *m*, where *m* is the size of the compressed file that the client wants to read. If resource availability on the client reduces, it can choose to read from some of the already received coefficient files. Fig, 4 shows sample meshes and images transmitted and rendered to screen sizes whose dimensions are similar to those for laptops, PDAs and cell phones.

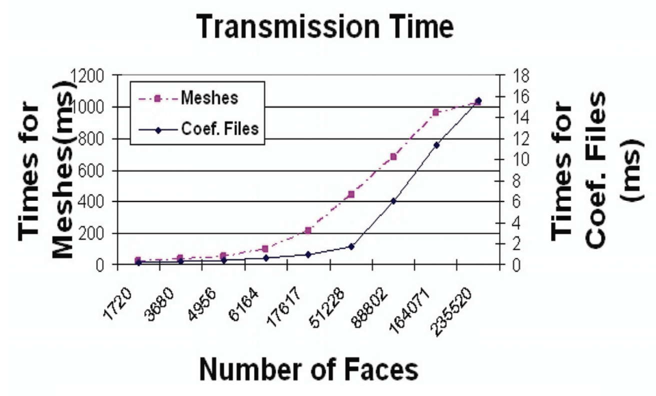


Fig. 5. Transmission Time of Bunny model

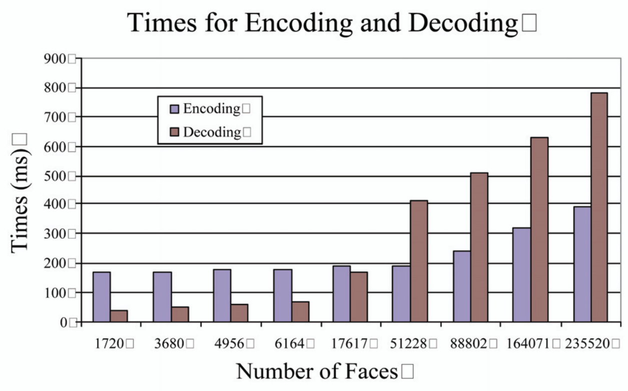


Fig. 6. Encoding and decoding times for bunny mesh

IV. PERFORMANCE EVALUATION

In this section, we describe tests that we conducted using meshes and images to evaluate the performance of UbiWave. One main aim was to ensure that the overheads of converting to and from wavelets do not overshadow the gains. Our general approach was to compare the times for transmitting and decoding wavelet-based meshe and image LODs to times required without using wavelets.

Meshes: We compare times for directly transmitting a given mesh LOD versus the time required to encode, transmit and decode the same mesh using wavelets. Our main result is that even with the overheads of encoding and decoding using wavelets, it is faster to transmit meshes using wavelets than by directly transmitting a mesh of the same LOD. Fig. 5 shows the transmission time for meshes and coefficient files of bunny model with wireless network speed 11Mbps. It can be observed that the time required to transfer time coefficient files is significant less than the transfer time for the actual mesh. This demonstrates that the use of wavelets to encode meshes can save transmission time and network bandwidth.

Fig, 6 shows only the times for encoding and decoding of a bunny mesh model. The server is run on a desktop PC and the client is run on a laptop (CPU: Celeron 1.0GHz, RAM: 256MB). Since we assume that the server does not have limits on system resources and that mesh encoding can be done as a pre-process, the encoding time at the server is not considered as a constraint. However, decoding is done on the ubiquitous computing device whose resources might be limited. Thus, the decode time on the mesh should be minimized as much as possible.

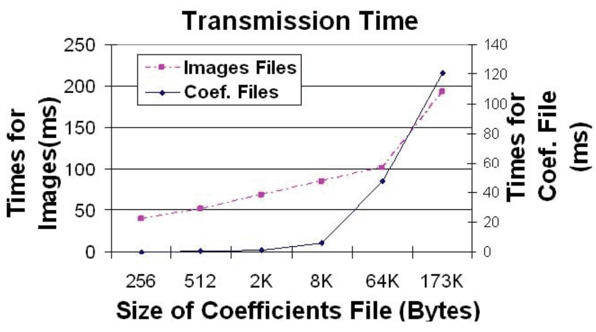


Fig. 7. Transmission Time for images

Images: Fig. 7 shows the transmission time for images and coefficient files with wireless network speed 11Mbps. Again, it can be observed that the time required to transfer the coefficient files is significantly less than the transfer time for the actual images. This demonstrates that the use of wavelets to encode images can save transmission time and network bandwidth.

V. SERVER HEURISTIC FOR SAVING MOBILE RESOURCES

At the server, we can use our PoI metric to reduce resource consumption. The main idea here is that if we can use the PoI metric to accurately determine the PoI for a given mesh or image, that LOD can be sent to the mobile device instead of the original high resolution. In this section, we shall now estimate how much transmission time and battery power is saved by using the PoI.

Battery power capacity in mobile devices has also shown very little growth, especially when compared with the exponential growths of CPU power, memory and disk space. Hence, battery power is typically the most constraining resource on a mobile device. We quantify potential battery energy savings by using a lower resolution mesh. We measure the power consumption for receiving, decoding and rendering a given mesh resolution on the mobile client using our tool PowerSpy [2]. PowerSpy is a software that tracks the energy consumption of MS Windows applications at the thread and I/O device levels. We calculate the total power consumption by summing up the power consumption of the CPU, disk and network cards as $P=P_{CPU}+P_{Disk}+P_{NetworkCard}$.

As an example, from our calculations, when the screen size is 640*720, the PoI of bunny model is the level with 88802 faces, which means there is not much perceptual difference if the number of faces is greater than 88802. If we use 88802 faces instead of the original mesh, the difference in resource usage is saved. Fig. 8(a) shows the saved transmission time, decoder time and power consumption in the mobile device.

Thus, using our perceptual metric, in this example it is possible to save over 60% of the transmission time, 34% of the decoding time and 45% of the total battery power. Similar results for an image are tabulated in Fig. 8(b). For the image in this example, it is possible to save over 60% of the transmission time, 35% of the decoding time and 38% of the total battery power.

Faces Number	88802	225520	Saved
T_{Trans}	6ms	15.6ms	61.5%
$T_{Decoder}$	511ms	782ms	34.7%
Power Consumed	24956mw	45630mw	45.3%

(a) Saved resourses for mesh

Coeff. File Size	64KB	173KB	Saved
T_{Trans}	47.6ms	120.5 ms	60.5%
$T_{Decoder}$	340ms	530ms	35.8%
Power Consumed	7467mw	12156mw	38.6%

(b) Saved resources for image

Fig. 8. Saved resources for a sample mesh and image

The server decision heuristic is as follows. The server calculates our perceptual metric based entirely on the mobile device's screen size as well as the original mesh and image resolutions. The server is then able to decide to send a mesh of a lower resolution (before the PoI) if the perceptual error is less than a threshold.

VI. Related Work

We discuss related work and highlight how our UbiWave is novel.

Novel end-to-end solution: Our envisioned solution starting from stored captured content to rendering on the mobile device is novel. Using the multiresolution properties of wavelets to address the heterogenity of mobile devices, and wireless networks is also novel. Several authors have proposed *alternate techniques to reduce the bandwidth and resource usage of graphics applications* but do not use wavelets. Examples are image-based simplification [37],geometry compression [17,26], and progressive transmission [13]. *Alternate scalable graphics representations* such as point-based rendering [9] have also been proposed. Points support scalable rendering but do not achieve aggressive compression rates. Spherical harmonics [18, 21] is used for shading in low frequency lighting, but cannot be used for high frequency lighting or geometry. Finally, a few *resource-adaptive graphics systems* take alternate approaches. The ARTE system [25] implements polygonal simplification and LOD techniques, but does not use wavelets or mitigate wireless channel errors. Other mobile graphics architectures been proposed in [10].

Novel components of our solution: We are also developing four new algorithms to be used in UbiWave, to complete our end-to-end vision. Out-of-core simplification and UEP had previously been proposed for meshes stored as a vertex list. However, both schemes are specific to the structure of the underlying mesh representation. No prior work on UEP [1] or out-of-core simplification [15, 22] techniques for meshes have been decomposed using wavelets, so we are developing these algorithms. A metric for determining what rendering resolution is best for each type of device is also proposed. This metric combines a) the level of distortion of a mesh caused by simplification, and b) its final look when rendered on a given mobile screen size. In the literature, previous work either focused on a) or b) but did not combine them or consider the effect of different screen sizes. The metric we propose is the sensitive both to mesh resolution and screen size, which is novel. Finally, there is a lot of work on ray tracing but none examines its energy usage or tries to render to the lowest acceptable level of realism.

VII. Conclusion and Future Work

We have described UbiWave, our system for scalable distribution and rendering of graphics content using wavelets. We presented results of our current UbiWave prototype which enabled significant resource savings when rendering on ubiquitous computing devices. We are currently integrating our out-of-core algorithm for large mesh simplification and our UEP scheme for forward error correction.

References

[1] G. Al-Regib and Y. Altunbasak. An unequal error protection method for packet loss resilient 3-d mesh transmission, *IEEE INFOCOM*, vol. 2 , *New York City, NY*, pp. 743–752, June 2002.

[2] K Banerjee and E Agu. Powerspy: Fine-grained software energy profiling for mobile devices, *in Proceedings IEEE WirelessCom Conference, Maui, Hawaii,* 2005.

[3] J N Bradley and C M Brislawn. The wavelet/scalar quantization compression standard for digital fingerprint images, *in Proceedings IEEE Int'l Symp*, on *Circuits and Systems (ISCAS)*, 1994.

[4] C. Christopoulos, A. Skodras and T. Ebrahimi. The jpeg2000 still image coding system: an overview, *IEEE Trans. Consumer Electronics*, vol. 46, no. 4, pp. 1103–1127, 2000.

[5] P. Clarberg, W. Jarosz, T. Akenine-Moller and H. Wann Jensen. Ray tracing: Wavelet importance sampling: efficiently evaluating products of complex functions, *ACM Transactions on Graphics (TOG) (SIGGRAPH Proc)*, vol. 24, no. 3, 2005.

[6] T. D.Derose, M. Lounsbery and J. Warren. Multiresolution analysis for surfaces of arbitrary topological type, *ACM Transactions on Graphics (SIGGRAPH Proc)*, vol. 16, no. 3, pp. 34–73, 1997.

[7] O. Devillers, D. Levin and O. Remez. Progressive compression of arbitrary triangular meshes, *IEEE Visualization*, pp. 67–72, 1999.

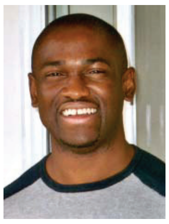
[8] N. Dyn, D. Levin and J. A. Gregory. A butterfly subdivision scheme for surface interpolation with tension control, *ACM Trans. on Graphics*, pp. 160–169, 1990.

[9] F. Duguet and G. Drettakis. Flexible point-based rendering on mobile devices, *IEEE Comp. Graphics and Appl*, pp. 57–63, July/August 2004.

[10] F. Lamberti, C. Zunino, A. Sanna, A. Fiume and M. Maniezzo. An accelerated remote graphics architecture for pdas, *in Proceedings ACM Web3D*, pp. 55–61, 2003.

[11] G. Pingali, C. Pinhanez, A. Levas, R. Kjeldsen, M. Podlaseck, H. Chen and N. Sukaviriya. Steerable interfaces for pervasive computing spaces, in Proc Percom,2003.
[12] A Graps. A friendly guide to wavelets, *IEEE Computational Science and Engineering,* vol. 2, no. 2, Summer 1995.
[13] H. Hoppe. Progressive meshes, *in Proceedings ACM SIGGRAPH*, pp. 189– 198, 1997.
[14] H. Hoppe. Progressive meshes, in Proc ACM SIGGRAPH, pp. 99– 108, 1996.
[15] M. Isenburg and S. Gumhold. Out-of-core compression for gigantic polygon meshes, *in Proceedings ACM SIGGRAPH*, pp. 935–942, 2003.
[16] J. Borchers, M. Ringel, J. Tyler and A. Fox. Stanford interactive workspaces: A framework for physical and graphical user interface prototyping, *IEEE Wireless Communications, special issue on Smart Homes*, December 2002.
[17] J. Rossignac. Edgebreaker: Connectivity compression for triangle meshes, *IEEE Trans. on Vis and Computer Graphics*, pp. 47–61, 1999.
[18] J. Kautz, P. Sloan and J. Snyder. Fast arbitrary brdf shading for low-frequency lighting using spherical harmonics, *in Proceedings Eurographics workshop on Rendering*, pp. 291–296, 2002.
[19] P. Lalonde. Representations and uses of light distribution functions, *Ph.D thesis*, *The University of British Columbia*, 1997.
[20] P. G. Lemarie and Y. Meyer. Ondelettes et bases hilbertiennes, *Rev. Mat. Iberoamericana*, vol. 2, pp. 1–18, 1986.
[21] C. Lindsay and E Agu. Wavelength dependent rendering using spherical harmonics, *in Proceedings Eurographics*, 2005.
[22] P. Lindstorm. Out-of-core construction and visualization of multiresolution surfaces, *In ACM 2003 Symp. Int. 3D Graphics*, pp. 93–102, 239, 2003.
[23] C. Loop. Smooth subdivision surfaces based on triangles, *Master's thesis. Dept. of Math, Univ. of Utah*, 1987.
[24] M. Lounsbery. Multiresolution analysis for surfaces of arbitrary topological type, *PhD. Thesis, University of Washington*, 1994.
[25] I. M. Boier-Martin. Adaptive graphics, *IEEE Comp. Graphics and App,* pp. 6–10, Jan-Feb 2003.
[26] M. Deering. Geometry compression, *in Proceedings ACM SIGGRAPH*, pp. 13–20, 1995.
[27] P. Lindstrom, G. Turk. Fast and memory efficient polygonal simplification, *in Proceedings IEEE Vis*, pp. 279–286, 1998.
[28] L. A. F. Park, K. Ramamohanarao and M. Palaniswami. A novel document retrieval method using the discrete wavelet transform, *Trans. on Graphics (TOG)*, vol. 23, no. 3, 2005.
[29] Azuma R, Baillot Y, Behringer R, Feiner S, Julier S and MacIntyre B. Recent advances in augmented reality, *IEEE Comp. Graphics and Appl.* vol. 21, no. 6, pp. 34–47, Nov/Dec 2001.
[30] R. W. H. Lau, F. Li, T. L. Kunii, B. Guo, B. Zhang, N. Magnenat-Thalmann, S. Kshirsagar, D. Thalmann and M. Gutierrez. Emerging web graphics standards and technologies, *IEEE Comp. Graphics and Appl.*
[31] P. Schroder and W. Sweldens. Spherical wavelets: Efficiently representing functions on the sphere. *in Proceedings ACM SIGGRAPH*, pp. 161–172, 1995.
[32] W. J. Schroeder. Decimation of triangle meshes, *in Proceedings ACM SIGGRAPH*, pp. 65–70, 1992.
[33] W. Sweldens. The lifting scheme: A custom-design construction of biothogonal wavelets, *Applied and Comp. Harmonic Analysis*, vol. 3, pp. 186–200, 1996.
[34] S. Valette and R. Prost. Multiresolution analysis of irregular surface meshes, *IEEE Trans. Visual. Compu. Graphicst*, vol. 10, pp. 113–122, 2004.
[35] T. Whitted. An improved illumination model for shaded display, *Comm. of ACM,* vol. 23, no. 6, pp. 343–349, June 1980.
[36] F Wu and E Agu. Unequal error protection for wavelet-based wireless mesh transmission, *in ACM SIGGRAPH, Boston, MA (research poster)*, 2006.
[37] X. Decoret, F. Durand, F. Sillion and J. Dorsey. Billboard clouds for extreme model simplification, *in Proceedings ACM SIGGRAPH*, 2003.

Fan Wu received the B.S. and M.S. degrees in computer science from Nanjing University of Posts and Telecommunications, Nanjing, China, in 2000 and 2003 respectively. Currently he is a Ph.D. candidate in the Computer Science Department at Worcester Polytechnic Institute. Since August 2003, he has worked with Professor Emmanuel Agu. His research interests include mobile graphics, mobile and ubiquitous computing and 3-D multiresolution graphic compression and transmission.



Emmanuel Agu is currently an assistant professor in the Computer Science Department at Worcester Polytechnic Institute. Dr. Agu received M.S. and Ph.D. degrees in Electrical and Computer Engineering from the University of Massachusetts at Amherst in 1996 and 2001 respectively. His research interests are in mobile graphics, mobile and ubiquitous computing, wireless networking and computer graphics. He has authored or co-authored over 25 publications in these areas. His research has been funded by the NSF and ATI Inc. Dr. Agu has led research into several adaptive architectures for ubiquitous graphics including MADGRAF and UbiWave, as well as middleware components for adaption including PowerSpy and Remote Mesa. He has served as a reviewer for the ACM SIGGRAPH, Eurographics conferences and Eurographics Symposium on Rendering.

Matthew Ward is currently a full professor in the Computer Science Department at Worcester Polytechnic Institute. Dr. Ward received his B.S. degree in Computer Science from Worcester Polytechnic Institute in 1977 and his M.S. and Ph.D. in Computer Science from the University of Connecticut in 1979 and 1981, respectively. He was employed as a Member of the Technical Staff in the Robotics and Computer Systems Research Laboratory at AT&T Bell Laboratories between 1980 and 1984 and as a research scientist at Skantek Corporation until 1986, when he joined the faculty at WPI. His research interests include data and information visualization, visual languages, and scientific data management and analysis. He has authored or co-authored more than 70 publications in these areas, and is actively involved in the development of a text book in the area of data visualization. His research has been funded by government agencies, including NSF, NSA, the Air Force Research Labs, and the DOT, and by industry, including IBM, Sun Microsystems, and SGI. Dr. Ward is the primary architect on several public-domain software packages for multivariate data visualization and exploration, including XmdvTool, MAVIS, and SpiralGlyphics. He serves on the organizing committees for the IEEE Symposium on Information Visualization and IEEE Conference on Visualization, and is an associate editor for IEEE Transactions on Visualization and Computer Graphics.