Reprint Seventeenth Int. Conf. on Interactive Information and Processing Systems (IIPS) for Meteorology, Oceanography and Hydrology, 81st AMS, Jan. 14-19, 2001, Albuquerque, NM, American Meteorological Society, 2001, pp. 338-342.

9.1 MULTIRESOLUTION TILING FOR INTERACTIVE VIEWING OF LARGE DATASETS

K. Palaniappan and Joshua B. Fraser
Department of Computer Engineering and Computer Science
University of Missouri-Columbia, MO 65211
palani@cecs.missouri.edu, fraser@meru.cecs.missouri.edu

1. INTRODUCTION

The ever-increasing size of spatial datasets in a variety of applications such as remote sensing and scientific visualization requires the development of new data structures and software tools for efficiently managing and manipulating very large datasets. The new generation of sensors, instruments and numerical models combined with enhanced computational capabilities, produce datasets that are many times larger than the physical computer memory available to view that data. Individual simulations in computational fluid dynamics are now 300 GB in size for which highly interactive browsing combined with automatic feature extraction is desired for understanding complex phenomena (Bryson et al. 1999). How does one ensure interactivity when loading a 100 GB dataset into memory can take up to an hour even with high-performance hardware. Another example from remote sensing is to interactively browse a Landsat mosaic of the Unites States (US). A multispectral (6 band) Landsat TM mosaic of the conterminous US at 30 m resolution, requires 428 scenes (single coverage) and results in an image with dimensions of 218,000 x 95,000 pixels that uses 160 gigabytes (GB) of storage space at multiple resolutions (Plesea and Curkendall 2000). This mosaic was created at the NASA Jet Propulsion Laboratory, and is one of the largest seamless single images produced to date. It was shown as part of the AMS 2000 Electronic Theater presentations. Strategies for dealing with large datasets for scientific visualization include sparse traversal and compression (Bryson et al. 1999). Methods for viewing large digital imagery using tiling and discrete wavelet transform (DWT)-based compression are described by Bradley (1998), Hovanes et al (1999) and Diego et al (2000).

Furthermore, extremely large datasets often reside on a remote computer system and hence require integrated support for network access to view and manipulate this data. The objective of this article is to describe a system for interactively viewing large datasets that not only exceed available memory resources but potentially exist only in the secondary storage of a remote system such as a digital library. We will refer to this software as `kolam` (K-tiles for Optimized muLtiresolution Access with coMpression).

2. OPTIMIZATION CRITERIA

A primary goal of developing `kolam` is to provide for *interactive* viewing and manipulation of datasets much larger than the available physical memory. Interactivity or responsive means that a constant refresh rate with minimal latency needs to be maintained while the user manipulates the on-screen data by roaming or zooming. Performance goals for accessing and redrawing the required data, would be to maintain at least 10 Hz update rates with latency less than one tenth of a second. In order to provide interactivity disk-based input and output operations need to be handled efficiently. In particular, `kolam` should minimize local disk I/O as well as minimize network transmission bandwidth for remote datasets. In order to handle the large sized datasets `kolam` supports compression at the tile level or the image level for efficient storage on disk while providing for multiresolution access. Tradeoffs will need to be made in optimizing the following criteria during system design as well as in selecting parameters for preprocessing a given dataset:

- Minimize disk I/O
- Maximize refresh rate
- Minimize network bandwidth
- Maximize file compression

3. LARGE DATASET VIEWING SYSTEM REQUIREMENTS

The `kolam` software tool for large dataset display can be incorporated as part of a comprehensive visualization system such as the Interactive Image SpreadSheet (IISS) tool (Palaniappan et al. 1993; Hasler et al. 1994) or Distributed Image SpreadSheet (DISS) system (Palaniappan et al 2001). In order to accommodate extensibility, scalability and adaptability to different types of datasets `kolam` will be designed and implemented using a modular object oriented approach. The components of `kolam` include data representation with choices for a multiresolution tiling scheme, support for non-integer scaling of data, tile compression codecs, disk file storage format, multithreaded execution and remote access mechanisms. These components are discussed below.

3.1 *Multiresolution Tiling*

One general strategy for supporting interactive viewing of very large datasets is by reorganizing the data such that it is decomposed into tiles and maintained at multiple resolutions. The advantage to tiling is that only those parts of the image that are visible on the screen necessarily have to be in memory. Other tiles not currently in view can still reside in memory, but are available for removal if that memory is needed for

Reprint Seventeenth Int. Conf. on Interactive Information and Processing Systems (IIPS) for Meteorology, Oceanography and Hydrology, 81st AMS, Jan. 14-19, 2001, Albuquerque, NM, American Meteorological Society, 2001, pp. 338-342.

another purpose--such as another tile.

Maintaining multiple resolutions of the data is done in order to reduce I/O, memory usage, and the burden on the graphics hardware. For example, if the data to be viewed is pixel-for-pixel larger than the screen on which it is to be displayed, but the user wishes to view the entire dataset all at once, then the entire dataset must be scaled or culled before it can be displayed. For large datasets this is computationally too expensive to be done interactively. Additionally, if the dataset is larger than physical memory, then the process of scaling and display becomes I/O intensive due to the many disk accesses required to retrieve and scale the data in small pieces. The following methods are proposed for structuring the data such that the data is both tiled and available at multiple resolutions: scale and tile, tile and scale, and several hybrid methods.

3.1.1 *Scale and Tile (SaT) Method*

The scale and tile method first subsamples the data to provide multiple resolutions of the data. Each of these subsampled layers is then broken into tiles. Provided that fixed-size tiles are used, each successively coarser resolution requires fewer tiles than the previous resolution. Potentially, the coarsest resolution may only require a single tile as shown in Figure 1. Dividing the data into fixed-size tiles after scaling is advantageous because it reduces the number of I/O operations necessary to display a particular view of the data. At the coarsest resolution, few tiles are needed to recreate the full image, and consequently few tiles need to be fetched from the data source--whether network or secondary storage. At finer resolutions, only those tiles which are currently visible on the screen need to be read from the data source, once again reducing the number of I/O operations required to display the current view.

Each of these I/O operations entails reading the entire tile, but the majority of secondary storage devices and network protocols perform most efficiently when making a few large block reads rather than many small reads. Further efficiency is achieved by keeping an in-memory cache of most recently viewed tiles, and only reading from the data source when a needed tile has not yet been read into memory, or when a needed tile has expired and been paged out of memory.

This method of tiling is nearly identical to that of the paging method of many operating systems. In fact there are direct correspondences between entities in an operating system memory paging system and this multiresolution tiling system:

- Layer $\Rightarrow$ Segment
- Image $\Rightarrow$ Virtual Memory Space
- Tile $\Rightarrow$ Page
- Pixel Coord $\Rightarrow$ Virtual Memory Address
- Local Cache $\Rightarrow$ Swap Cache

The scale and tile method achieves its performance efficiency at the cost of additional secondary storage usage. The extra storage space results from creating multiple resolutions of the data. For a two-dimensional NxN image, each successive subsampling results in an image one-quarter the size of the previous image which must also be saved to disk:

$$N^2(1+\frac{1}{4}+\frac{1}{4^2}+\cdots+\frac{1}{4^n}) \le \frac{4}{3}N^2$$

This results in a cost of 33% more storage space than the original image. These costs are for storing the raw images to disk without applying any compression. Compressing each tile independently can substantially reduce the storage requirements as discussed below. With 10:1 (lossy) compression, the original image plus all of the resolutions in the hierarchy could be stored in just 13% of the storage requirement for the uncompressed original image.

3.1.2 *Tile and Scale (TaS) Method*

The tile and scale method differs from the SaT method in that it first breaks the data into tiles, and then applies a transform to the data such as a wavelet transform (Chai et al. 1999). Applying the wavelet transform to each tile is done as a part of compressing each tile using a progressive compression method. The benefit of using a progressive compression scheme on a per-tile basis is that multiple resolutions of the data are inherently available without explicitly subsampling the data as shown in Figure 2. So multiple resolutions of the data need not be explicitly saved, and thus some of the storage overhead of the SaT method is avoided.

The TaS method, however, suffers from performance penalties with regard to in-memory caching, and excessive disk I/O. In-memory caching is complicated because tile size varies for each resolution. In the most extreme case where the entire dataset is to be displayed, and the dataset is of sufficient size such that each tile of the dataset is represented by a single pixel on-screen, each in-memory tile would be of size 1 pixel by 1 pixel.

In addition, this method incurs the overhead of excessive disk I/O. When displaying a zoomed-out view of the data, every tile of the image not in memory at the correct resolution will have to be accessed from disk. For each of these disk reads, only a few bytes of the tile will be read and displayed. This results in a large number of relatively small reads. For example, a dataset with dimensions 128,000 by 128,000 pixels, divided into tiles of size 512 x 512 pixels would result in 62,500 tiles. This dataset viewed at 128th scale using this method would result in 62,500 disk reads resulting in only one or two bytes per read.

3.1.3 *Hybrid Methods*

A hybrid method combining both methods discussed above could provide the desired performance of the scale and tile method without the extreme storage overhead. This method would use different methods for different resolutions of the data. For fine resolutions,

Reprint Seventeenth Int. Conf. on Interactive Information and Processing Systems (IIPS) for Meteorology, Oceanography and Hydrology, 81st AMS, Jan. 14-19, 2001, Albuquerque, NM, American Meteorological Society, 2001, pp. 338-342.

the tile and scale method would be used, but for coarser resolutions the scale and tile method would be used. The result is a compromise between the two methods resulting in lighter storage requirements and fewer disk operations.

3.2 *Non-integer Zoom*

Non-integer zoom means displaying the data at scales that are not a power of two--for example 3/4 scale. Obviously the data cannot be maintained at all possible resolutions, so the data must be interpolated on the fly in order to provide these "in-between" scales. The solution is to acquire the data at the next higher power of two resolution, or two neighboring scales and use the graphics hardware or software-only method to perform the interpolation.

3.3 *Compression*

Several types of progressive methods have been considered for compressing each tile of the multiresolution image data: DCT-based compression and wavelet-based compression. A popular DCT-based compression method is JPEG (Joint Photographic Experts Group), and the wavelet-based compression methods include SLCCA (Significance-Linked Connected Component Analysis) as described in Chai et al (1999) and Vass et al (1999, 2000) and JPEG-2000. These different methods will be tested although the wavelet-based SLCCA out-performs DCT-based JPEG as well as JPEG 2000 (Diego 2000). JPEG-2000 supports lossy and lossless compression of single-component or multi-component data, region-of-interest variable fidelity coding, progressive image access based on quality or resolution, random access to image regions using tiling, better compression performance using DWT, and good error resilience (Diego 2000).

The SLCCA codec can be adapted for progressive access to multiscale image content, enabling scale-wise decoding of compressed tiles in bit-plane order. As a result of this progressive scheme, each tile that is requested for display would contain image information at several scales not all of which may be needed for display. For this reason the request for a compressed tile would need to specify the desired resolution of that tile using Tile and Scale method. Consequently, decoding the tile can still occur in bit-plane order as is done currently, but with the added constraint of scale by scale decoding or scale ordering placed on the decoding process.

Performance will be a key issue since decoding will be entirely software-based, and the need to access a large number of tiles may lead to unsatisfactory performance, even though the entire compressed file may be much smaller.

3.4 *File Format*

One of the proposed structures in `kolam` for providing multiresolution tiling is a hierarchical one where each level of the hierarchy represents a level of resolution, and each resolution is broken into tiles. Two existing image formats support such a hierarchical structure natively: Hierarchical Data Format (HDF) and Tagged Image File Format (TIFF) (Adobe 1992).

While HDF is commonly used for remote sensing data applications, it is cumbersome and would require extensive modifications to adapt to tiling and compression. Additionally, HDF does not provide a directory of the data contained within the file up front. Rather, an HDF reader must parse most of the entire file in order to inventory the data contained which can significantly affect performance.

TIFF provides a directory structure for the data contained within the file allowing a reader to quickly seek to desired data. Additionally, as the name suggests, TIFF uses a tagged format to identify information about the data contained in the file. Through the addition of new tags, the TIFF format can be readily extended. An example of this is the GeoTIFF format which extends the standard to provide support for projection and navigation information. Finally, TIFF already provides support for tiling and compression natively as well as accommodating new methods of compression within the TIFF specifications.

Extending TIFF would provide a convenient way of storing tiled multiresolution compressed datasets on disk. However, the TIFF standard requires that an organization wishing to extend the format with additional tags must either be issued a private tag identifying the specialized format, or distinguish the extended format from the standard format. For example, GeoTIFF is an extension of the TIFF standard. These additional requirements are only necessary if the format is to be used outside of the organization.

While TIFF should be adopted for the final version this system, for the purpose of prototyping and experimentation, a specialized format can be used for ease of development and modifications.

3.5 *Multithreaded Support*

Due to the latency-sensitive demands of interactively viewing these datasets, support for multiple threads (when multiple processors are available) needs to be designed into any applications or libraries used for decoding or viewing these datasets. In terms of both performance and organization, this problem can be broken into tasks according to the burdens it places on the machine: disk and network I/O, memory usage, and CPU usage. From this notion we get the following scheme for threads:

- Single thread for display
- One or more threads for fetching data
- One or more threads for decoding data

Though display is the most latency-sensitive task of those listed, it is the least demanding of machine resources. Additionally, because the graphics hardware and the libraries used to interface that hardware provide

for only a single point of entry, the display should be singly threaded.

The fetching of data is highly I/O intensive, and is expected to be the bottleneck of the entire system. For this reason, multiple threads will be an integral part of the fetch process. With respect to fetching data, this data may or may not be local to the machine; the data could in fact reside on a remote machine and require a network fetch.

The notion of providing separate threads for decoding data may or may not be beneficial. Experimentation will be required to determine whether or not any gains can be made by separating the fetching and decoding stages. Further investigation is needed to determine whether or not the SLCCA decoding process can be split into multiple threads, and whether or not any gains can be made from this split.

3.6 *Remote Access*

Most of this discussion has focused on the data being local to the system on which it is to be displayed. The data may, however, reside on a remote machine. This system must provide an efficient means of viewing large datasets residing on remote machines. This can be effected by using a simple client server-model and local caching in conjunction with the methods for multiresolution tiling. Two possibilities exist for communicating the data between the system that maintains the data and the viewer: a specialized protocol, or a standard protocol such as HTTP.

Using a specialized protocol would allow for maintaining an open connection with the remote system. An open connection would allow for retrieving multiple tiles without requiring multiple connections, and consequently reducing connect time overhead. However, a specialized protocol would require longer development time to create both the client and the server as well as design and debug the protocol itself.

Using the standard Internet protocol HTTP would create a connection-less system that would suffer connection-time overhead for each tile request. However, this system can be implemented more quickly than the specialized protocol. Additionally the data "server" would actually only exist as a CGI (Common Gateway Interface) program run by the HTTP server. This would result in a more secure system as well as a system that is easier to install.

4. CONCLUSIONS

A new approach for handling the visualization of very large datasets using a combined tiled, multiresolution and compression approach was described. The proposed kolam system is being implemented for working with large multispectral mosaic imagery and is being extended for the visualization of large volumetric data (Vass et al 2000).

5. ACKNOWLEDGEMENTS

This work was supported in part by NASA/GSFC Award NAG-5-3900, NASA MTPE Award NAG-5-6968, NAG-5-6283, NAG-13-99014, and the NSF vBNS Award 9720668.

6. REFERENCES

Adobe Developers Association, 1992: TIFF Specification. TIFF Revision 6.0, 121 pp. [Available from Adobe Systems Incorporated, 1585 Charleston Rd., P.O. Box 7900, Mountain View, CA 94039.] and http://www.remotesensing.org/geotiff for GeoTIFF.

Bradley, J.N., 1998: *Storage and Retrieval of Large Digital Images.* U.S. Patent 5710835, 13 pp.

Bryson, S., D. Kenwright, et al, 1999: "Visually exploring gigabyte data sets in real time". *Communications ACM*, Vol. 42, No. 8, 83-90.

Chai, B.-B., J. Vass, and X. Zhuang, 1999: "Statistically adaptive wavelet image coding". *Visual Information Representation, Communication, and Image Processing,* C.W. Cheng and Y.Q. Zhao, Marcel Dekker, 73-96.

Diego S.-C., T. Ebrahimi, et al., 2000: JPEG 2000 still image coding versus other standards. *Proc. of SPIE*, Vol. 4115.

Hasler, A. F., K. Palaniappan, M. Manyin, J. Dodge, 1994: A high performance interactive image spreadsheet (IISS). *Computers in Physics*, **8**, 325-342.

Hovanes, M.E., J.R. Grizz Deal, and A.H. Rowberg, 1999: Seamless multiresolution display of portable wavelet-compressed images. *Journal of Digital Imaging*, **12**, 109-111.

Palaniappan, K., A. F. Hasler, J. Fraser, M. Manyin, 2001: Network-based visualization using the Distributed Image SpreadSheet (DISS), *17th Int. Conf. on Interactive Information and Processing Systems (IIPS)*, Albuquerque, NM, Amer. Meteor. Soc.

Palaniappan, K., A. F. Hasler, M. Manyin, 1993: Exploratory analysis of satellite data using the Interactive Image Spreadsheet (IISS) environment. *9th Intl. AMS Conf. IIPS for Meteorology, Oceanography, and Hydrology*, Anaheim, CA, AMS, 145-152.

Plesea, L. and D. Curkendall, *Landsat Map of the United States*, http://mapus.jpl.nasa.gov, NASA Jet Propulsion Lab, Personal comm., May 2000.

Vass, J., K. Palaniappan, and X., Zhuang, 2000: Three dimensional wavelet coding of volumetric imagery. *Int. Journal of Robotics and Automation*, **15**, 34-47.

Vass, J., B.-B. Chai, K. Palaniappan, X. Zhuang, 1999: Significance-linked connected component analysis for very low bit-rate wavelet video coding. *IEEE Trans. Circuits and Systems for Video Technology*, **9**, 630-647.

Reprint Seventeenth Int. Conf. on Interactive Information and Processing Systems (IIPS) for Meteorology, Oceanography and Hydrology, 81st AMS, Jan. 14-19, 2001, Albuquerque, NM, American Meteorological Society, 2001, pp. 338-342.

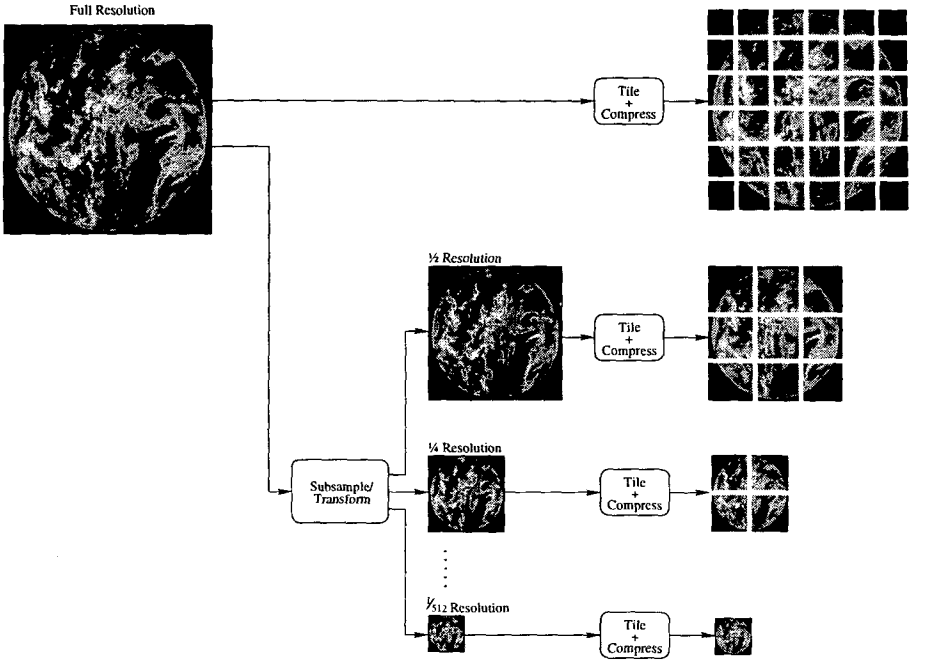

Figure 1: Multiresolution Scale and Tile method

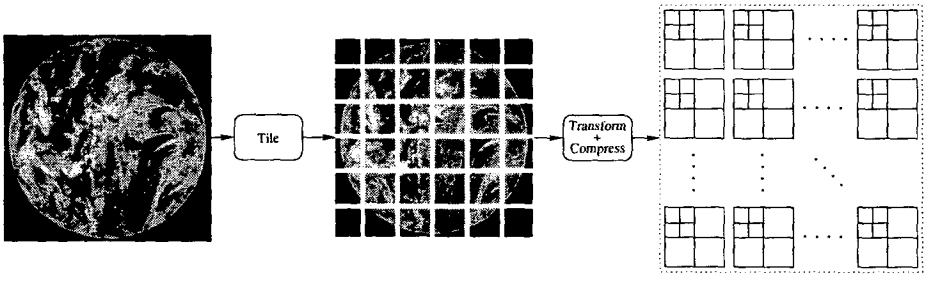

Figure 2: Multiresolution Tile and Scale method

Reprint Seventeenth Int. Conf. on Interactive Information and Processing Systems (IIPS) for Meteorology, Oceanography and Hydrology, 81st AMS, Jan. 14-19, 2001, Albuquerque, NM, American Meteorological Society, 2001, pp. 338-342.

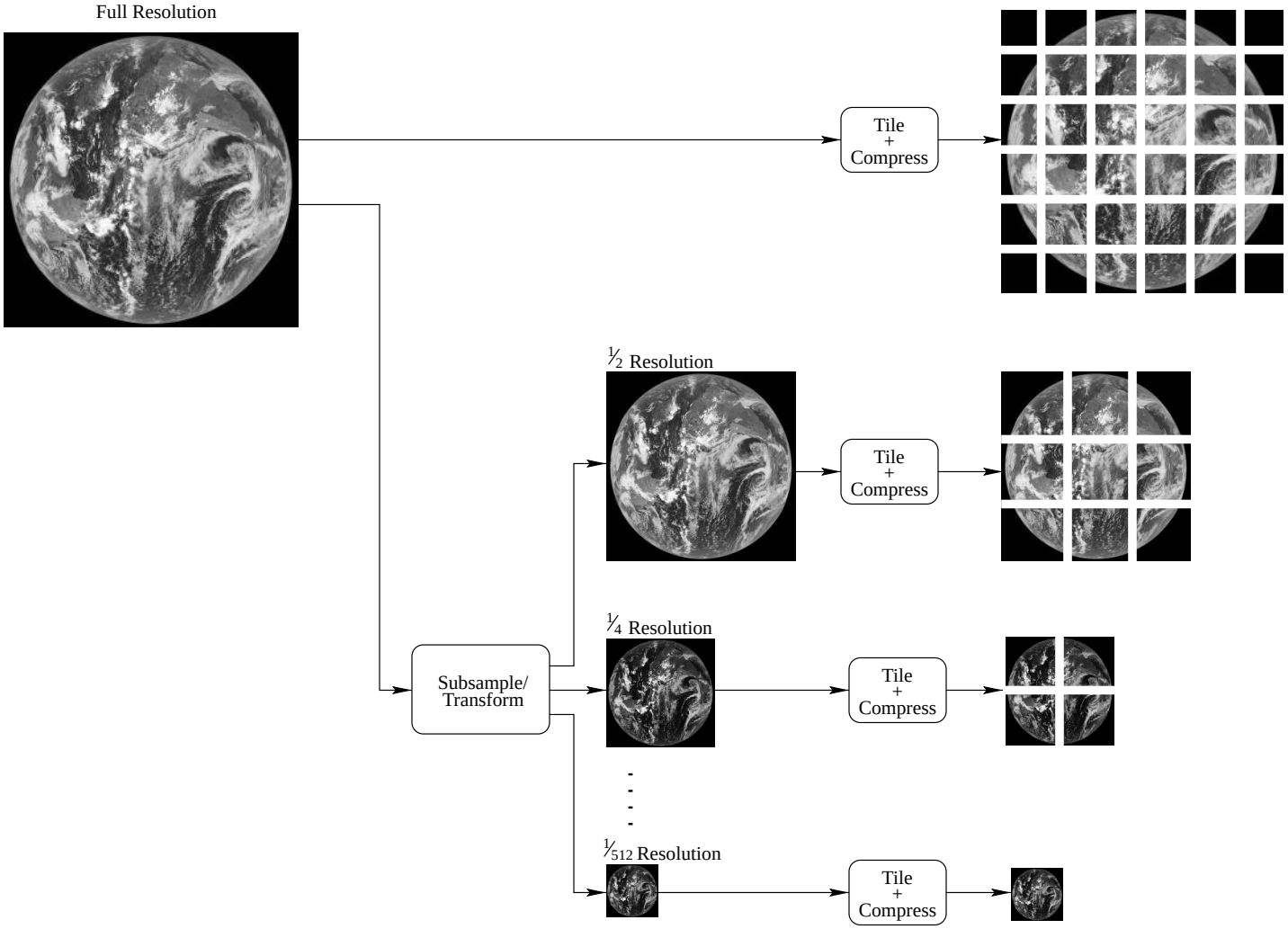

Figure 1: Multiresolution Scale and Tile (SaT) method

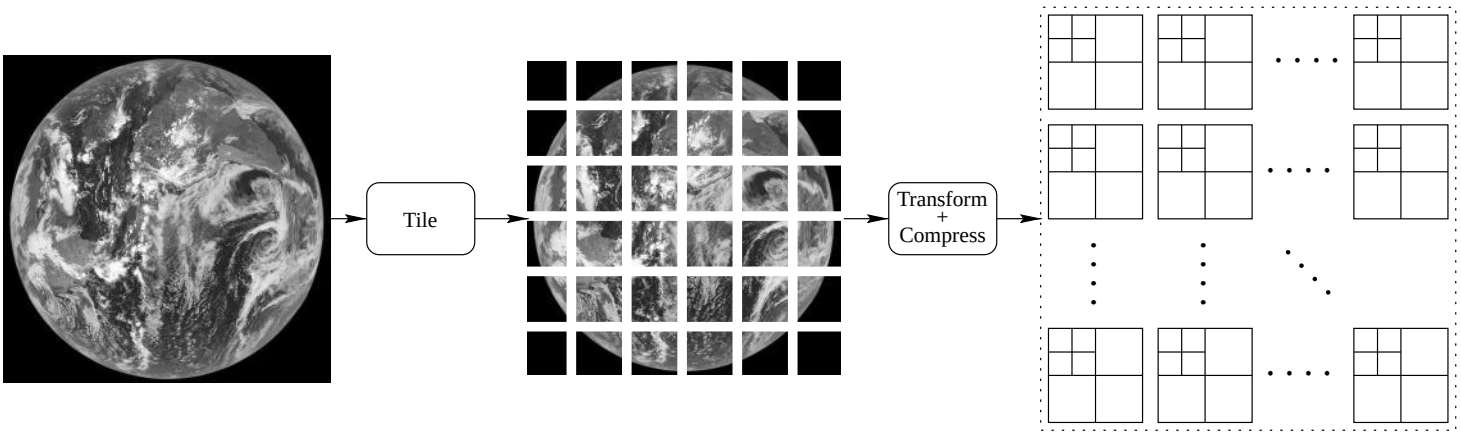

Figure 2: Multiresolution Tile and Scale (TaS) method