

# Interactive Image Browsing with JPEG2000/JPIP

Shaan Patel, Shahar Yuval, and David Lau  
Department of Electrical Engineering, Stanford University

**Abstract**—We present a comparison between three interactive image browsing schemes, Gaussian Pyramid, Laplacian Pyramid, and JPEG2000/JPIP. Using simulations of various browsing trajectories, we compare the image quality as a function of time with a fixed transmission rate. The benefits of JPEG2000/JPIP over the Pyramid schemes are observed. These benefits usually outweigh the complexity of JPIP implementation.

## I. INTRODUCTION

IN current interactive image browsing schemes, a server stores a large image as a grid of coarse tiles at various resolutions, and transmits the tiles that correspond to a client's region of interest (ROI). A preferred transmission scheme does not have significant redundancy, allows fast spatial random access, and optimizes quality of the user browsing experience. Currently, two widely deployed schemes for generating these tiles at various resolutions are Gaussian and Laplacian pyramids. Though these schemes are widely used in applications such as Google Maps, they have glaring disadvantages: Gaussian and Laplacian pyramids store redundant information since they do not fully exploit the dependencies between image resolution levels. Consequently, to explore the entire image at all resolution levels, the client downloads more data than the size of the maximum resolution image. The tiling approach also incurs a large performance penalty when a user moves slightly into a new tile and is required to download the entire tile, and not just the visible section. Another problem with pyramid schemes is that tiles can only be retrieved at a fixed bit-rate. Consequently, they cannot progressively refine image quality, because the entire tile has to be downloaded first. A potentially better approach is to use JPEG2000, in conjunction with its web transmission protocol, JPIP. JPIP avoids transmitting redundant sections and continuously optimizes the user's viewing experience. This paper will analyze and compare the performance of Gaussian pyramid, Laplacian pyramid, and JPIP for different trajectories, different images, and different transmission rates.

### A. Gaussian Pyramid

A Gaussian pyramid is a structure in which a high-resolution image is downsampled to a hierarchy of lower resolution images. Each resolution level is stored independently of the

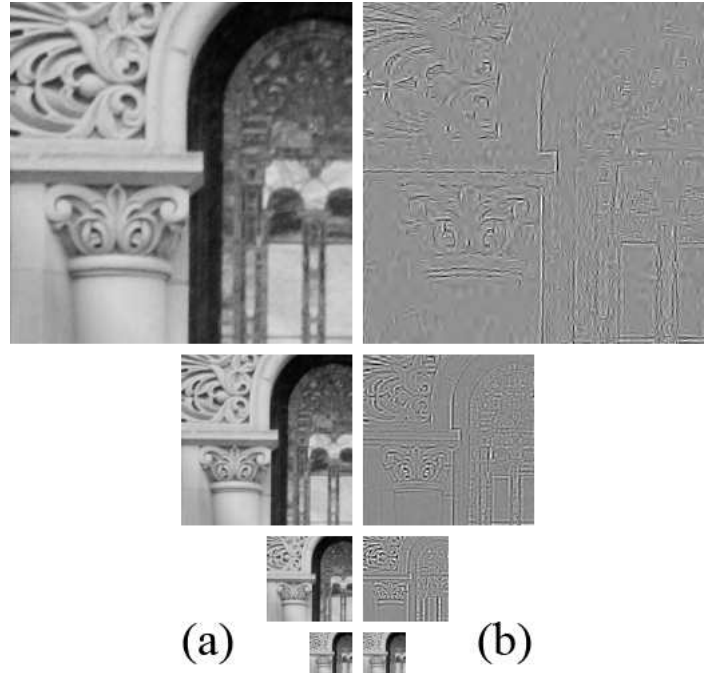


Fig. 1. Example of Gaussian Pyramids images (a) and Laplacian Pyramid images (b). This sample tile is showing a column's decoration and a stained glass window. Note that the lowest resolution images in both Pyramid schemes are the same image.

others; in addition, the image at each resolution level is partitioned into smaller tiles that are stored separately (see Fig. 1). When a user pans or zooms, the server sends every tile that intersects the ROI. If the user zooms, the server sends higher resolution copies of the tiles in the ROI. This causes significant redundancy, because neighboring pixels are highly correlated and they do not need to be retransmitted on zooming [1]. Also, progressive image browsing is not effective, because Gaussian pyramids at a certain resolution cannot provide tiles at intermediate resolutions; consequently they lack resolution scalability.

### B. Laplacian Pyramid

The Laplacian pyramid improves on Gaussian pyramid by computing the difference images between the resolution levels in the Gaussian pyramid (see Fig. 1). The minimum resolution level in the pyramid is the same as in the Gaussian case. Once a user zooms, the server sends the difference image used to reconstruct the higher resolution tile from the lower resolution

TABLE I  
COMPRESSION BIT-RATE OF IMAGES IN EXPERIMENT

Zoom Level	Gaussian Images	Laplacian Difference Images
0 (full zoom out)	Standard <sup>a</sup>	Standard <sup>b</sup>
1	Standard	Standard
2	3.5 bpp	2.5 bpp
3 (full resolution)	2.0 bpp	0.5 bpp

Table I. Compression bit-rates that were used in our experiment in bits per pixel (bpp). They were chosen using a training set and selected conservatively so that the reconstructed image at each zoom level would be at least 35dB.

<sup>a</sup> Standard compression means that no bit-rate was set—the JPEG2000 compression program decided which rate to use. This results in lossy compression, usually with a PSNR between 50dB and 54dB.

<sup>b</sup> This image is not a difference image and is exactly the same as the Gaussian image at this zoom level.

tile. This can be advantageous for two reasons: first, by taking differences, the variance of the coefficients in the difference image is significantly reduced, allowing for more efficient compression [1]. Second, a user can view a progressively improving image, as more difference images are fetched. However, there are shortcomings of Laplacian pyramid. If a user panned significantly to a ROI at a high resolution, a Laplacian scheme would need to retrieve the low resolution base image as well as all corresponding difference images to reconstruct at the corresponding resolution. In this case, a Gaussian pyramid scheme would only need to retrieve the image at that corresponding resolution. Since the size of the low-resolution base image plus all the difference images is greater than the size of the high-resolution image, it could be costly for the Laplacian pyramid. However, a Laplacian pyramid scheme would still offer progressive image refinement whereas as a Gaussian pyramid scheme would not.

### C. JPEG2000/JPIP

JPEG2000 image compression provides three main advantages over Gaussian and Laplacian pyramids: spatial random access, progressive image refinement and non-redundant compression. Spatial random access is achieved by dividing the image at each subband into precincts, collections of JPEG2000 codeblocks that describe only a single spatial region; this localizes the amount of data to be transferred. Progressive image refinement is achieved through two mechanisms: quality layers and DWT synthesis stages. Quality layers determine the coarseness of the quantization of each codeblock; as higher quality levels are used, codeblocks are quantized more finely. This feature is known as quality scalability. Since JPEG2000 is a wavelet based codec, lower resolution images can be achieved by omitting the final stages of DWT synthesis levels. A key feature for non-redundant compression is that subsections of the compressed image can be rendered meaningfully without being transcoded. This means that if the image is received and rendered in parts, the size of the final image will be the size of the original image, without any overheads [2]. Thus, as opposed to Laplacian and Gaussian pyramids, downloading the entire image in parts does not introduce redundancy, assuming the Internet header

overheads are negligible [3].

JPIP leverages the features of JPEG2000 to transmit interactive images on the web. The protocol defines an interface for clients to request sections of an image at a certain quality, and resolution from the server. Based on network conditions, rate-distortion estimation, the server can choose whether to serve the client's request, or modify the request to optimize the client's viewing experience. JPIP does so by optimizing the resolution, the quality scalability and the random access knobs provided by JPEG2000, as mentioned above [2]. This is an important advantage over Laplacian and Gaussian pyramids, which do not use additional judgment to modify requests.

## II. EXPERIMENTAL SETUP

The performance of a real JPIP implementation was compared with Matlab simulations of Gaussian and Laplacian pyramids.

### A. Generating Images

To generate Gaussian and Laplacian files the initial filter applied before downsampling was as suggested in [1], with coefficient  $a = 0.58$ . For filtering after upsampling, the 'interp' function in Matlab was used. The pyramids were then broken up into tiles and compressed and decompressed using JPEG2000. Table I shows the compression rates used to encode the tiles. This step was necessary for replicating lossiness when reading the image from the JPEG2000 file during viewing.

The image for use in the JPIP server was compressed with 4 DWT stages, and 30 quality levels. The internal heuristics of the encoder were allowed to determine how to distribute the bit-rates across quality levels, as suggested by [4]. However, due to the network nature of JPIP, an exact bit-rate is irrelevant, because JPIP can optimize the number of quality levels and DWT stages transmitted, depending on environmental conditions.

### B. Measuring JPIP Trajectory

JPIP trajectories were measured by connecting to the JPIP server from the JPIP image browser (kdu\_show) and storing screenshots of the client's focus window and sampling the amount of bytes downloaded between every screenshot. Samples were taken every 2000 ms, and the server was constrained to transmit data at a maximum transmission rate of either 8 KB/s or 16 KB/s.

### C. Simulating Trajectories

In image browsing, a user will follow a trajectory of movements based on zooms and pans. In this experiment it was assumed that all trajectories were equally likely. In JPIP a real-time simulation can be performed, but in Matlab this is more difficult to perform for the Gaussian and Laplacian pyramids. Instead, a log file generated from the trajectory in

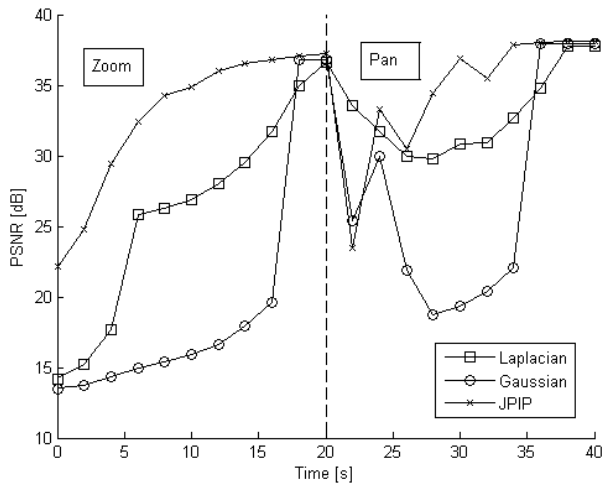


Fig. 2. Plot of PSNR vs Time in the base case run. Memorial Church photo at 8 KB/s, zoom of 25% and a pan 6 tiles. Time 0 is at the first sampling, 2s after the start of the run.

JPIP was read, which gives the pyramid simulations the full size focus window dimensions and the amount bytes downloaded since the last time interval. The Gaussian/Laplacian pyramid simulations have an allowance equal to this amount of bytes downloaded. Since Gaussian/Laplacian pyramid simulations are only allowed to download entire tiles at a time, if the simulation cannot utilize the allowance, the left over bytes are added to next time intervals allowance. However, this allowance is capped at 4000 bytes, because in a real trajectory, the Gaussian/Laplacian pyramid server does not know a priori what the user will request. This emulates the reality that in a network environment, parts of tiles would be transmitted in packets and not as entire tiles; however, this is a simplification. The simulation also keeps track of an unlimited cache to represent the large cache that JPIP will use in its image browsing session. The trajectories are kept relatively short, so it can be assumed that the cache never overflows.

#### D. Constructing Images

To render the focus window Matlab can find relevant cache and download any other needed tiles not found in the cache. If a tile could not be downloaded and a lower resolution tile exists, this tile will be used. Any missing tiles will be replaced with gray space as a blind estimate. After the images are collected, they are concatenated together.

#### E. Test Cases

To compare the three schemes we ran four different test cases:

- 1) *Base Case*: The image used was of Memorial Church, a high detail image (4096x4096). The network data transmission rate was 8 KB/s. For trajectory the user zoomed in to 25% of the maximum size, let the picture settle, and then panned 6 tiles down.
- 2) *Different Data Transmission Rates*: The same image and

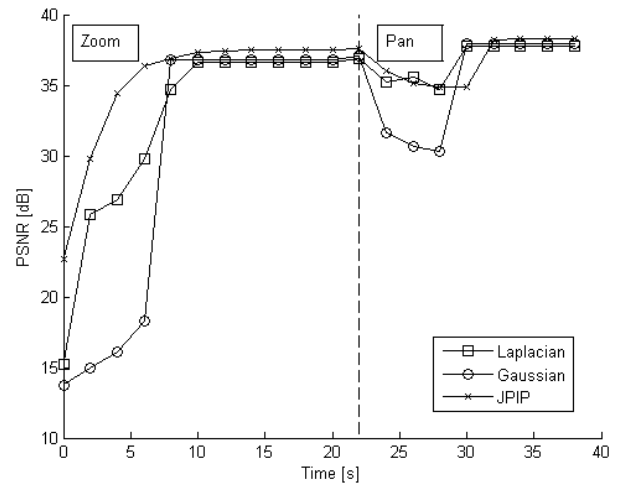


Fig. 3. Plot of PSNR vs Time in the different data transmission rate run. Memorial Church photo at 16 KB/s, zoom of 25% and a pan 6 tiles. Time 0 is at the first sampling, 2s after the start of the run.

trajectory were used. The network data transmission rate was increased to 16 KB/s.

- 3) *Different Trajectories*: Memorial Church was again used as the image. The network data transmission rate was 8 KB/s. For trajectory the user zoomed in to 25% of the maximum size, and immediately began to pan in a circle around the starting point, aggressively.
- 4) *Different Pictures*: The image used was of Stevens Creek Reservoir, a less detailed image (4096x4096). The network data transmission rate was 8 KB/s. For trajectory the user zoomed in to 25% of the maximum size, let the picture settle, and then panned 6 tiles down.

### III. DISCUSSION OF RESULTS

Fig. 2 shows a plot of PSNR versus time, for the base case. The analysis of the plots is broken into two time intervals: when the user is zooming in and when the user is panning. Initially when the user is zooming, there is an approximately 7 dB gap between the JPIP curve and the two pyramid curves. This can be explained by JPIP's ability to truncate the bit stream, and provide a very good initial view. Now as the user wait for the picture to settle, you see all three schemes' curves increasing until they peak at 20 seconds. The Gaussian pyramid curve increases the slowest; it takes 17 seconds before it can reach 25 dB, since it must retrieve high resolution image tiles immediately. The Laplacian pyramid curve shows steady improvement in visual quality since it can retrieve low-resolution image tiles first, and then move on to higher resolution difference image tiles. During the time interval 6 to 20 seconds, where Laplacian pyramid has retrieved all low resolution tiles and is now retrieving the first set of difference tiles, the separation between the Laplacian and the JPIP curves increases, until JPIP saturates. This is largely due to JPIP/JPEG2000 having one quarter less coefficients to encode for its DWT stage compared to the corresponding difference

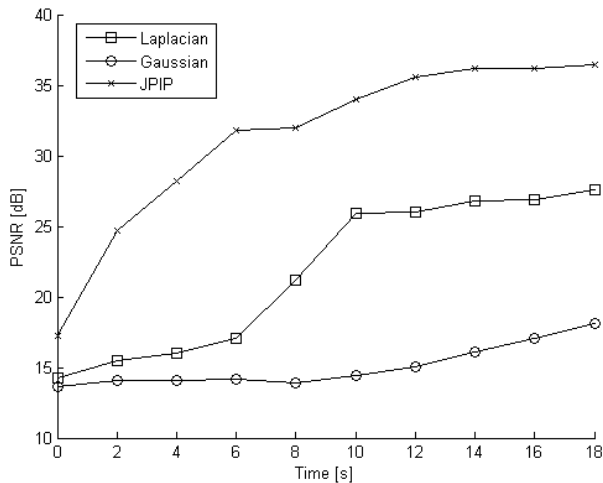


Fig. 4. Plot of PSNR vs Time in the different trajectory run. Memorial Church photo at 8 KB/s, zoom of 25% and a circular motion pan around the original starting point. Time 0 is at the first sampling, 2s after the start of the run.

image of the Laplacian pyramid. In the second time the time interval, where the user pans down 6 tiles, all three curves drop. Gaussian pyramid drops significantly and takes approximately 15 seconds, before it can reach 30 dB again. The reason for this behavior is that Gaussian pyramid must retrieve high-resolution tiles immediately, instead of being able to retrieve low-resolution tiles first like Laplacian pyramid. This feature of Laplacian pyramid allows it to recover more quickly. JPIP recovers quickly since it also can progress from lower resolution imagery to higher resolution imagery. In addition, JPIP can progress from lower to higher image quality due to its ability to optimize responses base on rate-distortion.

To observe if the properties discussed above hold under different circumstances, several other cases were studied. The first case studied was a higher transmission rate, 16 KB/s, as seen in Fig. 3. The curves in this plot are steeper than the curves in Fig. 2 during the zoom time interval; however, the reason for this is that at 16 KB/s, twice as much data can be downloaded in a given amount of time. The important feature to note in Fig. 3 is that the drops in PSNR, when the user pans, are significantly less than the drops seen in Fig. 2. Thus at a higher transmission rate, the benefits of Laplacian pyramid and JPIP become less significant. This makes sense, because if the client is given enough bandwidth it can retrieve almost everything that it needs to construct the focus window.

For the next case, a different trajectory was studied as seen in Fig. 4. Fig. 4 shows the benefits of progressive image refinement. Gaussian pyramid can never retrieve a reasonable image for the focus window as seen by peak PSNR of 18 dB. The Laplacian pyramid's use of resolution scalability, allows it to show some progressive image refinement, since from 6 seconds and onward its curve diverges from the Gaussian PSNR curve. Finally, JPIP's ability to use JPEG2000's flexible spatial random access, quality scalability and resolution scalability, allow it to maintain a 10 dB PSNR

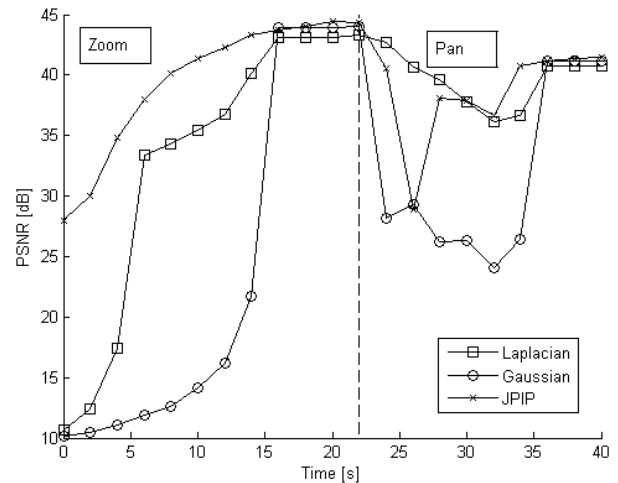


Fig. 5. Plot of PSNR vs Time in the different picture run. Reservoir photo at 8 KB/s, zoom of 25% and a pan 6 tiles. Time 0 is at the first sampling, 2s after the start of the run.

improvement over the Laplacian scheme.

Fig. 5 shows the results for the case of a less detailed image, Stevens Creek Reservoir. The curves are very similar to the curves found in Fig. 2, except the PSNR values are scaled. Since the image is less detailed it should be expected for a given number of bytes downloaded to have better image quality.

#### IV. FUTURE WORK

With the limited time scope of the project, there is still much to be explored. Below are possible experiments or topics that can be investigated:

- 1) Run tests over a network or network simulator. Observe how network overhead and potential loss of packets affect the results.
- 2) Work with larger images such as in Google Maps. Observe if JPIP's benefits still hold under larger scale applications. For instance, JPIP will have to incorporate tiling of very large images.
- 3) Test out longer user experiences with realistic caching methods. This may include investigating various user-behavior modes for improved pre-caching algorithms.

#### V. CONCLUSION

Following the analysis of the test cases, JPIP comes out as an attractive alternative to existing interactive image browsing schemes. The Gaussian pyramid offers simple, but restricted spatial random access through tiling, no progressive image refinement. The Laplacian pyramid can go one step further by offering some degree of progressive image refinement, through resolution scalability. The benefits of JPIP are that it offers quality scalability, resolution scalability, and flexible spatial random access. In most cases, these advantages improve user-browsing experience.

## ACKNOWLEDGMENT

Thanks to Chuo-Ling Chang for his guidance on our project.

## REFERENCES

- [1] P. J. Burt, E. H. Adelson, "The Laplacian Pyramid as a Compact Image Code," *IEEE Transactions on Communication*, vol. 31, pp. 532-540, 1983.
- [2] D. Taubman and R. Prandolini, "Architecture, philosophy and performance of JPIP: Internet protocol standard for JPEG2000," in *Proc. Int. Symp. Visual Communications and Image Processing (VCIP2003)*, vol. 5150, SPIE, Jun. 2003, pp. 791-805.
- [3] M. N. Do, M. Vetterli, "Frame Reconstruction of the Laplacian Pyramid", *Proc. Of IEEE*, Intl. Conf. on Acoustics, Speech, and Signal Processing, Salt Lake City, 2001.
- [4] D. Taubman (2002 August). "Proposal and Implementation of JPIP (Jpeg2000 Internet Protocol) in Kakadu v3.3" [Online]. pp. 31-33. Available: <http://www.kakadusoftware.com/jpip-kakadu-superceded.pdf>

## DIVISION OF WORK

This section indicates main contributors to specific sections. Equal contributions were made to all other parts.

A. *Shaan*

- Designed and implemented simulator for the Gaussian and Laplacian pyramid schemes.

B. *Shahar*

- Interfaced with the JPIP C++ libraries and demo applications, to capture screenshots and bytes downloaded.
- Generated test trajectories.

C. *David*

- Generation of Gaussian and Laplacian files and implemented low level image reconstruction.
- Collecting and stitching of the raw images.