# Demand-driven Image Servers for Combating EM Time to Pixel

Anon

Anon
anon@anon.edu,
WWW home page: http://anon.com/

**Abstract.** Modern electron microscopes acquire image sections at rates of multiple gigabytes per second. Often storing and distributing the dataset is difficult. Rendering requires a registered, stitched, and pyramid decomposed copy of the data, which can take days to compute. Increases in microscope acquisition rates outpace decreases in the time to compute and distribute the data for viewing, and so a different approach is needed. We propose a demand-driven image server for volume viewing which moves towards processing only the requested user display pixels on the fly from the unprocessed dataset. To remove dataset re-storage and re-distribution, clients access the data from the web: the browser viewport details are sent to the server, which replies with just the required pixels. This enables immediate presentation and exploration after data acquisition. We demonstrate the performance of our image server on two large Connectome datasets. We evaluate our infrastructure on a simulated infinitely large volume to validate scalability. We discuss our findings and provide an analysis of requirements for demand-driven rendering of large image volumes.

**Keywords:** Demand-driven rendering, electron microscopy, image server.

## 1 Intro

Paragraph on data bounds: How large is the problem? What kind of data rate do we need as throughput? How fast do existing systems work? How fast are these bounds increasing, and how fast are the bounds and the performance diverging?

"The XXX electron microscope has 12 heads and can read images at 1.21 gigawatts. Even when including time to prepare and load samples, this machine can produce a bazillion image pixels per second. Handling this amount of data is a daunting task for current compute resources."

Paragraphs on pipeline: What does the typical pipeline look like? Which parts take a long time? What impacts upon the time to output pixel?

Paragraph on problems of storage and distribution of data.

Paragraph on problems of aligning and viewing the data.

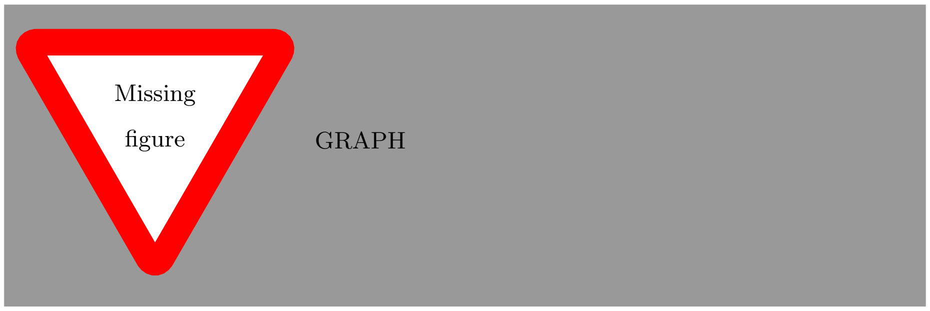

**Fig. 1.** Plot microscope 'time to pixel' for a bunch of different microscopes, and see how it is diverging from compute power.

How to solve? Towards the goal: What progress can we easily make towards this goal? What are low-hanging fruit that most people do badly, and we can attack? Demand-driven rendering.

What we do.

You can add todos like this!

### 1.1 Related Work

Won-ki work on demand-driven volumetric data

Won-ki work on demand-driven image editing for gigapixel imaging - pyramids still kept on disk, and platform is a desktop software application.

**??**

Cite Stackdrop and XTK and Dojo for web-based rendering of medical data.

However, none of these systems account for storage and distribution issues in their approaches. The consideration of storage, distribution, and viewing at the same time must form our approach.

## 2 Implementation

Server cluster with one PC per tile : )

1. Websocket canvas viewer
   (a) add basic interaction
   (b) add simple 3d rendering of current viewport $+-5$ in z using WebGL
2. Adjust openseadragon interface without advanced caching (this is one of the limitations)
3. Adjust slide atlas interface without advanced caching (another limitation due to Ajax)
4. Finish prefetching
5. Add blending while stitching as in
6. Adi: local MLS implementation in OpenCL?

**Fig. 2.** *Top:* The classic EM pipeline representing all steps the data must go through before viewing is possible. *Bottom:* Our approach, which attempts to remove all intermediate processing steps.

### 2.1 Caching Strategy

## 3 Results

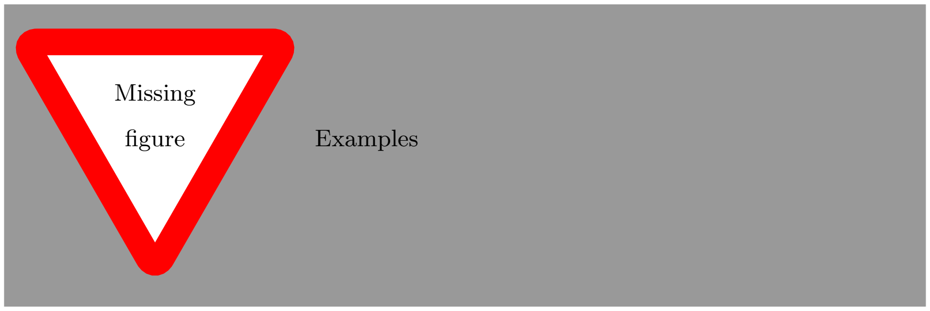

**Fig. 3.** *Left:* System in use with a cool dataset. *Right:* System in use with another cool dataset. Wow.

Examples: What datasets can we use? Alyssa, Josh

Evaluation: using the 1 or 2 datasets (see above) and maybe a simulated infinite datastream. What was the intended evaluation here? Performance?

The slide atlas and openseadragon performance vs. our websocket

## 4 Discussion

Limitations of current approach

Look at current bottlenecks in pipeline; project out bottlenecks, propose solutions to those bottlenecks.

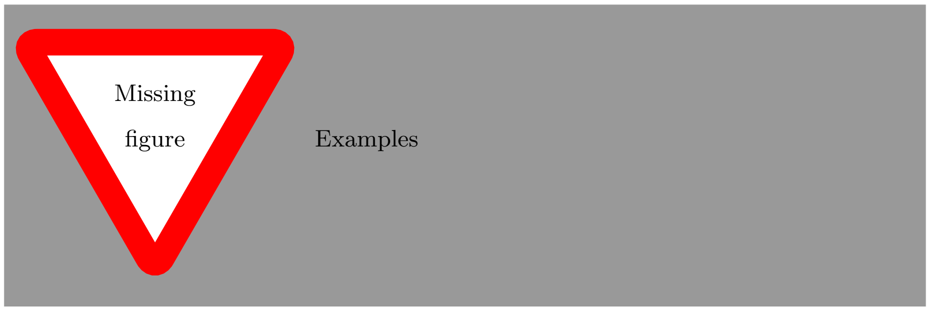

**Table 1.** Quantitative performance results. So good.

## References