- Current image renderers require buffering and again time

Viewing Gigapixel Images

Daniel Garcia-Briseno
Matthew Ostovarpour
Thomas O’Brien
Douglas Peterson

Faculty Mentor: James Palmer

- Professor of Computer Science
- Computer Science and Electrical Engineering Chair

United States Geological Survey (USGS)

Sponsor: Trent Hare

- Graduated from NAU with B.S. in Computer Mathematics and M.S. E in Computer Science.
- GIS-based analysis
- Creation of geospatial tools.

Our Goals

- Refine last year’s work
- Reduce current sampling times
- Develop robust GUI
- Probably should combine this with the our project slide

Last Year’s Project

What they did:

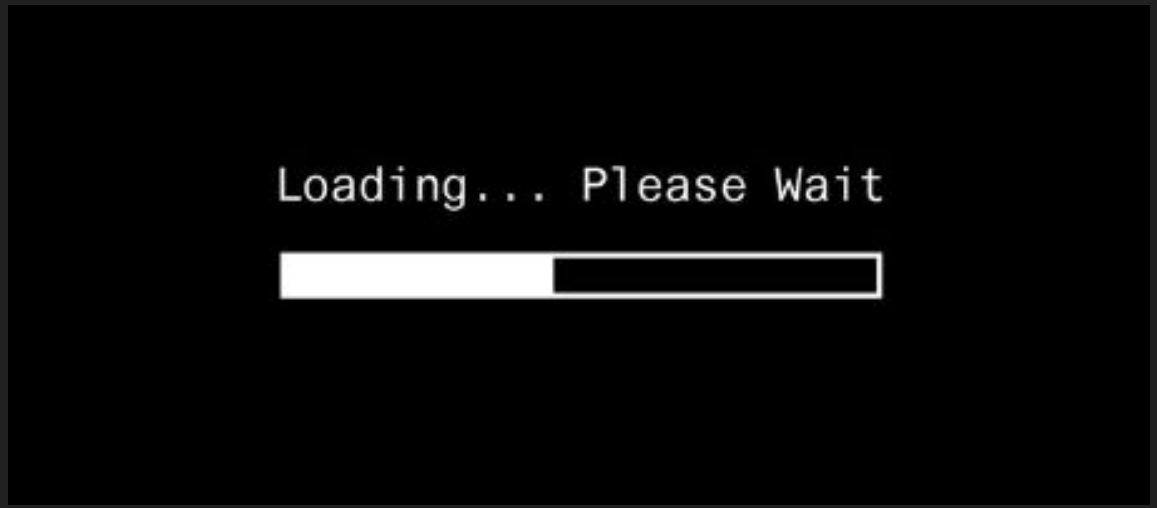
- Implement stochastic sampling
- Create primitive viewer
- Showed increased speed

What they did not:

- Build on Linux

The Problem

- Slow read times
- Panning/Zooming non-processed images
- Pyramids



Requirements Acquisition

Talk about how and from where we got our requirements

We should go into a medium amount of detail for this

I left the pyramid picture here so that we can use it for something else if we need it

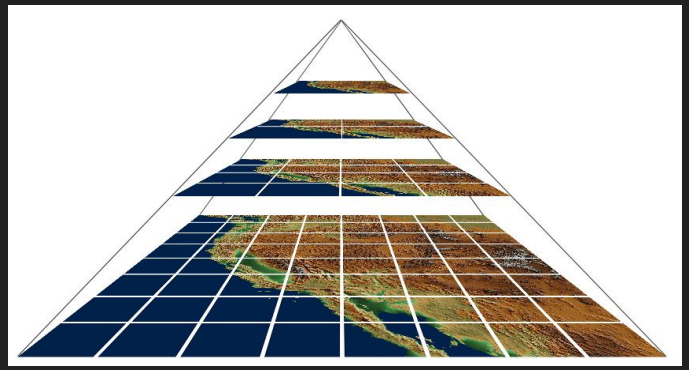


Image Viewers

- Tuiview
- Cartographica for Mac
- ArcGIS
- Geographic Imager plugin for Photoshop
- OpenEV
- QuantumGIS

Our Project

1. Create a stochastic image reader inside GDAL.
2. Create an interactive image viewer to utilize the reader.

Why inside GDAL?

- Accessible through C, C++, and Python
- Support for plenty of image formats (Listed Below)

Aeronav FAA files
AmigoCloud API
ESRI ArcObjects
Arc/Info Binary Coverage
Arc/Info .E00 (ASCII) Coverage
Arc/Info Generate
Atlas BNA
AutoCAD DWG
AutoCAD DXF
CartoDB
Cloudant / CouchDB
CouchDB / GeoCouch
Comma Separated Value (.csv)
OGC CSW (Catalog Service for the Web)
Czech Cadastral Exchange Data Format
DODS/OPeNDAP
EDIGEO
ElasticSearch
ESRI FileGDB
ESRI Personal GeoDatabase
ESRI ArcSDE
ESRI Shapefile
FMEObjects Gateway
GeoJSON
Géoconcept Export
Geomedia .mdb
GeoPackage
GeoRSS

Google Fusion Tables
Google Maps Engine
GML
GMT
GPSBabel
GPX
GRASS Vector Format
GPSTrackMaker (.gtm, .gtz)
Hydrographic Transfer Format
Idrisi Vector (.VCT)
Informix DataBlade
INTREST Data Format
INTERLIS
INGRES
JML
KML
LIBKML
Mapinfo File
Microstation DGN
Memory
MySQL
NAS - ALKIS
Oracle Spatial
ODBC
MS SQL Spatial
Open Document Spreadsheet
OGDI Vectors (VPF, VMAP, DCW)
More Vector formats...

Arc/Info ASCII Grid
ACE2
ADRG/ARC Digitilized Raster Graphics (.gen/.thf)
Arc/Info Binary Grid (.adf)
AIRSAR Polarimetric
Azavea Raster Grid
Magellan BLX Topo (.blx, .xlb)
Bathymetry Attributed Grid (.bag)
Microsoft Windows Device Independent Bitmap (.bmp)
BPG (Better Portable Graphics)
BSB Nautical Chart Format (.kap)
VTP Binary Terrain Format (.bt)
CALS Type I
CEOS (Spot for instance)
DRDC COASP SAR Processor Raster
TerraSAR-X Complex SAR Data Product
Convair PolGASP data
USGS LULC Composite Theme Grid
DirectDraw Surface
Spot DIMAP (metadata.dim)
ELAS DIPEx
DODS / OPeNDAP
First Generation USGS DOQ (.doq)
New Labelled USGS DOQ (.doq)
Military Elevation Data (.dt0, .dt1, .dt2)
Arc/Info Export E00 GRID
ECRG Table Of Contents (TOC.xml)
ERDAS Compressed Wavelets (.ecw)

ESRI .hdr Labelled
Erdas Imagine Raw
NASA ELAS
ENVI .hdr Labelled Raster
Epsilon - Wavelet compressed images
ERMapper (.ers)
Envisat Image Product (.n1)
EOSAT FAST Format
FIT
FITS (.fits)
Fuji BAS Scanner Image
Generic Binary (.hdr Labelled)
GeoPackage
Oracle Spatial GeoRaster
GSat File Format
Graphics Interchange Format (.gif)
WMO GRIB1/GRIB2 (.grb)
GMT Compatible netCDF
GRASS Raster Format
GRASS ASCII Grid
Golden Software ASCII Grid
Golden Software Binary Grid
Golden Software Surfer 7 Binary Grid
GSC Geogrid
Generic Tagged Arrays (.gta)
TIFF / BigTIFF / GeoTIFF (.tif)
More Raster Formats...

Image Downsampling Process

- Averaging
  - Average each pixel in a block
- Decimation
  - Use only the first pixel in each block

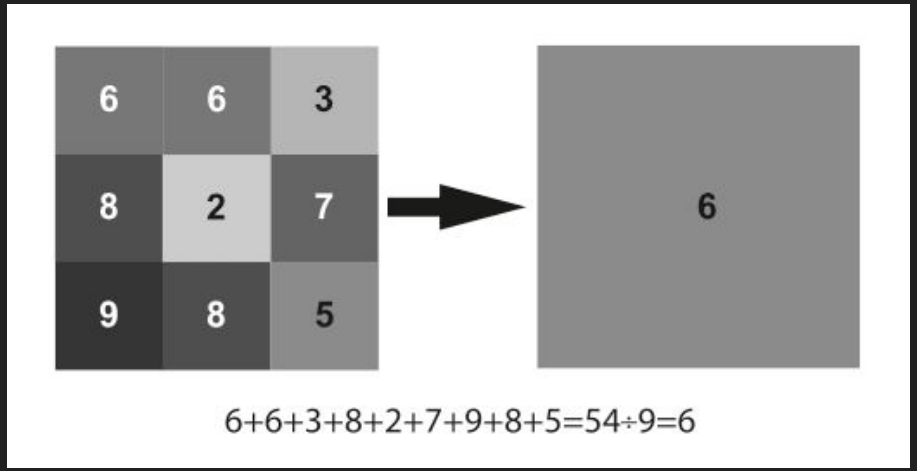
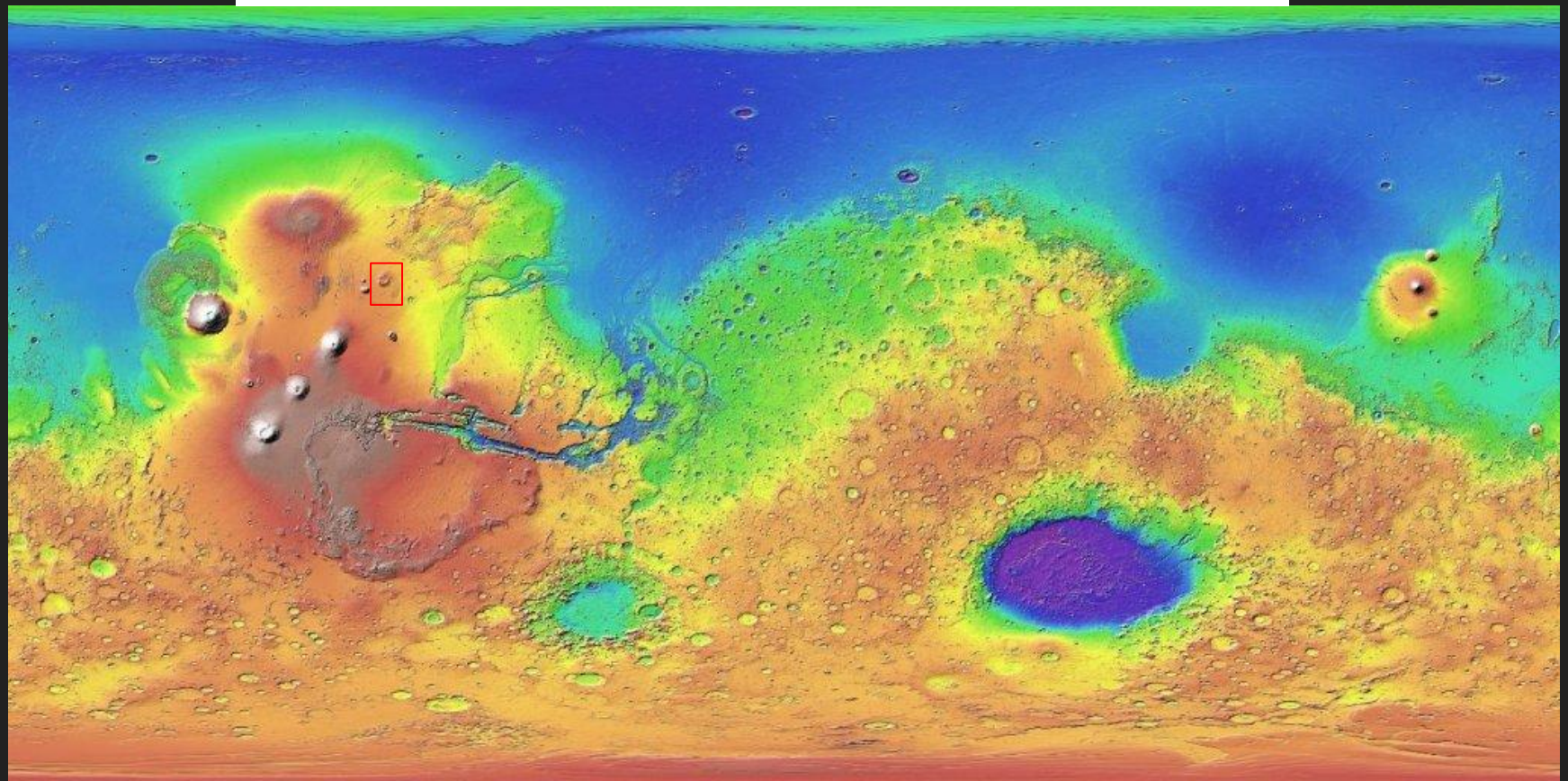


Image credit: USGS


Elevations above 9000 meters
found only on the larger volcanos
–8200 Minimum
21229 Maximum
–7000
–5000
–3000
–1000
0
1000
3000
5000
7000
9000
11000
13000
15000
17000
19000
21000

Downsampling a 1 gigapixel image (1 billion pixels)

32 x 32 Pixels per block = 1024 pixel blocks

976,563 blocks per image

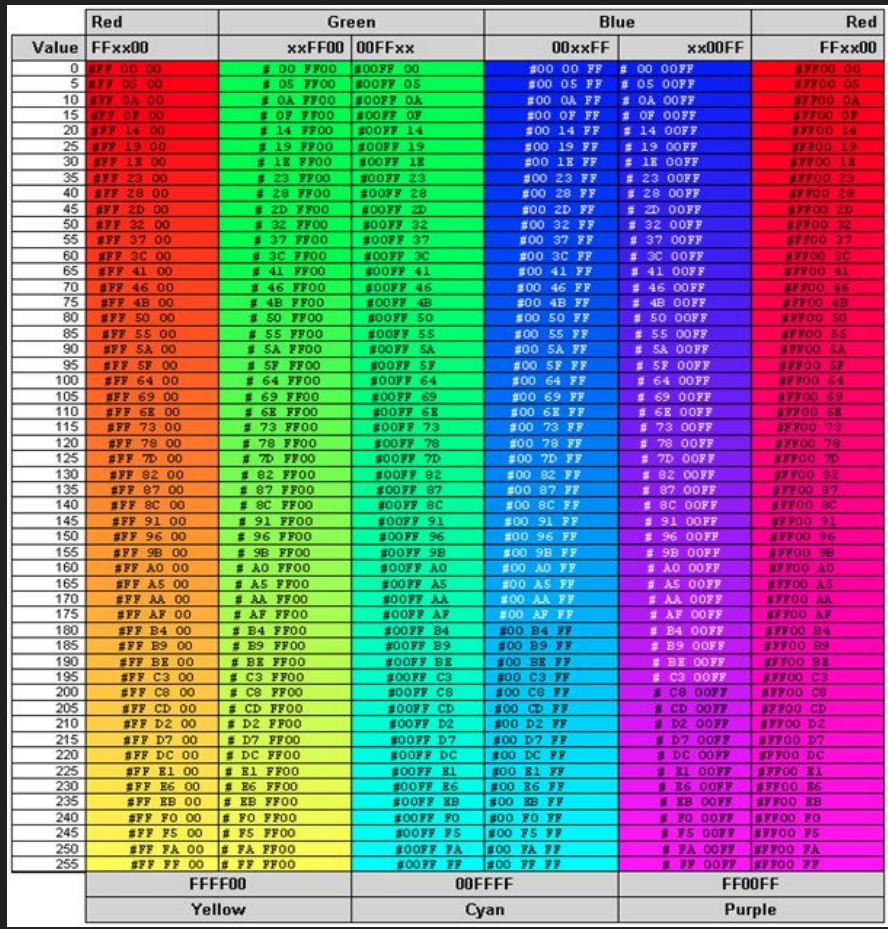
average block value = $\frac{1}{1024}\sum_{i=1}^{1024} pixel_i$

1023 adds + 1 divide

1024 calculations/block * 976,563 blocks = 1,000,000,000 mathematical operations for the image, plus 1,000,000,000 reads for each pixel.

Risks

- Quality is lost during downsampling
  - Will the increased speed be worth the quality that is lost?
  - Testing different amounts of samples could help to mediate this risk.
- High Color Variation
  - Result in slow sampling speed
  - Just test for RGB to speed up



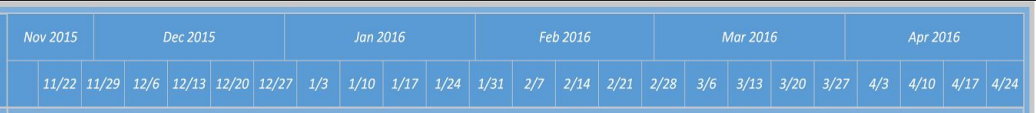
	Red	Green		Blue		Red
Value	FFxx00	xxFF00	00FFxx	00xxFF	xx00FF	FFxx00
0	#FF 00 00	# 00 FF00	#00FF 00	#00 00 FF	# 00 00FF	#FF00 00
5	#FF 05 00	# 05 FF00	#00FF 05	#00 05 FF	# 05 00FF	#FF00 05
10	#FF 0A 00	# 0A FF00	#00FF 0A	#00 0A FF	# 0A 00FF	#FF00 0A
15	#FF 0F 00	# 0F FF00	#00FF 0F	#00 0F FF	# 0F 00FF	#FF00 0F
20	#FF 14 00	# 14 FF00	#00FF 14	#00 14 FF	# 14 00FF	#FF00 14
25	#FF 19 00	# 19 FF00	#00FF 19	#00 19 FF	# 19 00FF	#FF00 19
30	#FF 1E 00	# 1E FF00	#00FF 1E	#00 1E FF	# 1E 00FF	#FF00 1E
35	#FF 23 00	# 23 FF00	#00FF 23	#00 23 FF	# 23 00FF	#FF00 23
40	#FF 28 00	# 28 FF00	#00FF 28	#00 28 FF	# 28 00FF	#FF00 28
45	#FF 2D 00	# 2D FF00	#00FF 2D	#00 2D FF	# 2D 00FF	#FF00 2D
50	#FF 32 00	# 32 FF00	#00FF 32	#00 32 FF	# 32 00FF	#FF00 32
55	#FF 37 00	# 37 FF00	#00FF 37	#00 37 FF	# 37 00FF	#FF00 37
60	#FF 3C 00	# 3C FF00	#00FF 3C	#00 3C FF	# 3C 00FF	#FF00 3C
65	#FF 41 00	# 41 FF00	#00FF 41	#00 41 FF	# 41 00FF	#FF00 41
70	#FF 46 00	# 46 FF00	#00FF 46	#00 46 FF	# 46 00FF	#FF00 46
75	#FF 4B 00	# 4B FF00	#00FF 4B	#00 4B FF	# 4B 00FF	#FF00 4B
80	#FF 50 00	# 50 FF00	#00FF 50	#00 50 FF	# 50 00FF	#FF00 50
85	#FF 55 00	# 55 FF00	#00FF 55	#00 55 FF	# 55 00FF	#FF00 55
90	#FF 5A 00	# 5A FF00	#00FF 5A	#00 5A FF	# 5A 00FF	#FF00 5A
95	#FF 5F 00	# 5F FF00	#00FF 5F	#00 5F FF	# 5F 00FF	#FF00 5F
100	#FF 64 00	# 64 FF00	#00FF 64	#00 64 FF	# 64 00FF	#FF00 64
105	#FF 69 00	# 69 FF00	#00FF 69	#00 69 FF	# 69 00FF	#FF00 69
110	#FF 6E 00	# 6E FF00	#00FF 6E	#00 6E FF	# 6E 00FF	#FF00 6E
115	#FF 73 00	# 73 FF00	#00FF 73	#00 73 FF	# 73 00FF	#FF00 73
120	#FF 78 00	# 78 FF00	#00FF 78	#00 78 FF	# 78 00FF	#FF00 78
125	#FF 7D 00	# 7D FF00	#00FF 7D	#00 7D FF	# 7D 00FF	#FF00 7D
130	#FF 82 00	# 82 FF00	#00FF 82	#00 82 FF	# 82 00FF	#FF00 82
135	#FF 87 00	# 87 FF00	#00FF 87	#00 87 FF	# 87 00FF	#FF00 87
140	#FF 8C 00	# 8C FF00	#00FF 8C	#00 8C FF	# 8C 00FF	#FF00 8C
145	#FF 91 00	# 91 FF00	#00FF 91	#00 91 FF	# 91 00FF	#FF00 91
150	#FF 96 00	# 96 FF00	#00FF 96	#00 96 FF	# 96 00FF	#FF00 96
155	#FF 9B 00	# 9B FF00	#00FF 9B	#00 9B FF	# 9B 00FF	#FF00 9B
160	#FF A0 00	# A0 FF00	#00FF A0	#00 A0 FF	# A0 00FF	#FF00 A0
165	#FF A5 00	# A5 FF00	#00FF A5	#00 A5 FF	# A5 00FF	#FF00 A5
170	#FF AA 00	# AA FF00	#00FF AA	#00 AA FF	# AA 00FF	#FF00 AA
175	#FF AF 00	# AF FF00	#00FF AF	#00 AF FF	# AF 00FF	#FF00 AF
180	#FF B4 00	# B4 FF00	#00FF B4	#00 B4 FF	# B4 00FF	#FF00 B4
185	#FF B9 00	# B9 FF00	#00FF B9	#00 B9 FF	# B9 00FF	#FF00 B9
190	#FF BE 00	# BE FF00	#00FF BE	#00 BE FF	# BE 00FF	#FF00 BE
195	#FF C3 00	# C3 FF00	#00FF C3	#00 C3 FF	# C3 00FF	#FF00 C3
200	#FF C8 00	# C8 FF00	#00FF C8	#00 C8 FF	# C8 00FF	#FF00 C8
205	#FF CD 00	# CD FF00	#00FF CD	#00 CD FF	# CD 00FF	#FF00 CD
210	#FF D2 00	# D2 FF00	#00FF D2	#00 D2 FF	# D2 00FF	#FF00 D2
215	#FF D7 00	# D7 FF00	#00FF D7	#00 D7 FF	# D7 00FF	#FF00 D7
220	#FF DC 00	# DC FF00	#00FF DC	#00 DC FF	# DC 00FF	#FF00 DC
225	#FF E1 00	# E1 FF00	#00FF E1	#00 E1 FF	# E1 00FF	#FF00 E1
230	#FF E6 00	# E6 FF00	#00FF E6	#00 E6 FF	# E6 00FF	#FF00 E6
235	#FF EB 00	# EB FF00	#00FF EB	#00 EB FF	# EB 00FF	#FF00 EB
240	#FF F0 00	# F0 FF00	#00FF F0	#00 F0 FF	# F0 00FF	#FF00 F0
245	#FF F5 00	# F5 FF00	#00FF F5	#00 F5 FF	# F5 00FF	#FF00 F5
250	#FF FA 00	# FA FF00	#00FF FA	#00 FA FF	# FA 00FF	#FF00 FA
255	#FF FF 00	# FF FF00	#00FF FF	#00 FF FF	# FF 00FF	#FF00 FF
	FFFF00		00FFFF		FF00FF	
	Yellow		Cyan		Purple	

Key Requirements

- Must be faster than current readers
- Must produce a viewable image
  - Image quality must be high enough to view features in the image
- Scalable image quality
  - When zooming in on image, the quality should increase
- Support for multiple image formats

Schedule

ID	Task Name	Start	Finish	Duration
1	Get familiar with GDAL	11/17/2015	11/23/2015	1w
2	Implement primitive stochastic reader	11/24/2015	11/30/2015	1w
3	Study outputs to find most efficient outputs based on viewing area	12/2/2015	12/14/2015	1.8w
4	Implement “smart read”	1/12/2016	2/8/2016	4w
5	Optimize reader	2/9/2016	4/22/2016	10.8w
6	Render GDAL output in bgfx application.	1/12/2016	1/21/2016	1.5w
7	Implement image panning	1/22/2016	2/2/2016	1.5w
8	Implement zoom	2/3/2016	2/12/2016	1.5w
9	Implement “recently viewed” cache	2/15/2016	3/4/2016	3w
10	Create toolbar	3/7/2016	3/16/2016	1.5w
11	Optimization/Optional Features	3/17/2016	4/22/2016	5.4w



Conclusion

- Current image readers are inefficient.
- Stochastic reader has potential to improve productivity.
- GDAL integration will allow it to be used by anyone.