An AI-enabled Data Refinery for Satellite Imagery at Global Scale

Mike Warren
Co-founder & CTO
mike@descarteslabs.com
@m_warren
My background

Computational Astrophysics...

...and all sorts of Computing
Descartes Labs

- Venture-backed startup spun out of LANL in December 2014
- Machine learning, computer vision, satellite imagery
- Team of >40 physicists, philosophers, mathematicians, software engineers, geographers
- Building a living atlas of the world: persistent, real-time, multi-modal
- Acquire, process, and store imagery (NASA, ESA, commercial)
- Headquartered in Santa Fe, satellite offices in Los Alamos, San Francisco and New York City
In August 2017, we announced our $30M Series B

A data refinery, built to understand our planet
Our geospatial analysis platform

A growing archive of analysis-ready images, with historical records for back-testing models.

Robust pipeline for continuous updating as new images become available.

Multiple satellite image datasets integrated into a single system.

Cloud-based data and services, ready-to-use for data science teams anywhere.

Elastic computing model—scaling compute resources quickly and only when needed.
GeoVisual search

Divide the earth's surface into (billions of) small, overlapping images

Extract a "visual feature vector" from each image using a convolutional neural network

Search for neighbors in this feature space using either direct search or locality-sensitive hashing

*t-SNE visualization of image chips*
## Festivus: our virtual cloud FS

<table>
<thead>
<tr>
<th>Node Type</th>
<th>Nodes</th>
<th>Bandwidth (GB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-vCPU</td>
<td>1</td>
<td>0.43</td>
</tr>
<tr>
<td>4-vCPU</td>
<td>1</td>
<td>0.85</td>
</tr>
<tr>
<td>16-vCPU</td>
<td>1</td>
<td>1.0</td>
</tr>
<tr>
<td>32-vCPU</td>
<td>1</td>
<td>1.44</td>
</tr>
<tr>
<td>16-vCPU</td>
<td>4</td>
<td>4.1</td>
</tr>
<tr>
<td>16-vCPU</td>
<td>16</td>
<td>17.4</td>
</tr>
<tr>
<td>16-vCPU</td>
<td>64</td>
<td>36.3</td>
</tr>
<tr>
<td>16-vCPU</td>
<td>128</td>
<td>70.5</td>
</tr>
<tr>
<td>16-vCPU</td>
<td>512</td>
<td>231.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Blocksize (bytes)</th>
<th>Festivus (MB/s)</th>
<th>gcsfuse (MB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32768</td>
<td>12.5</td>
<td>0.4</td>
</tr>
<tr>
<td>65536</td>
<td>22.6</td>
<td>0.8</td>
</tr>
<tr>
<td>131072</td>
<td>47.3</td>
<td>1.6</td>
</tr>
<tr>
<td>262144</td>
<td>93.0</td>
<td>2.8</td>
</tr>
<tr>
<td>524288</td>
<td>156.8</td>
<td>7.3</td>
</tr>
<tr>
<td>1048576</td>
<td>271.0</td>
<td>13.7</td>
</tr>
<tr>
<td>2097152</td>
<td>472.0</td>
<td>24.8</td>
</tr>
<tr>
<td>4194304</td>
<td>852.3</td>
<td>46.7</td>
</tr>
<tr>
<td>8388608</td>
<td>1046.4</td>
<td>109.5</td>
</tr>
<tr>
<td>16777216</td>
<td>1248.0</td>
<td>200.3</td>
</tr>
<tr>
<td>33554432</td>
<td>1593.3</td>
<td>339.7</td>
</tr>
</tbody>
</table>
Example Problem: How do we find large storage tanks?
## Cost of execution vs development

<table>
<thead>
<tr>
<th>2016 Cost ($/s)</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1.0 \cdot 10^{-8}$</td>
<td>Gigabyte</td>
<td>Cloud storage</td>
</tr>
<tr>
<td>$1.5 \cdot 10^{-8}$</td>
<td>Gigabyte</td>
<td>Persistent magnetic disk</td>
</tr>
<tr>
<td>$6.5 \cdot 10^{-8}$</td>
<td>Gigabyte</td>
<td>Node solid state disk</td>
</tr>
<tr>
<td>$1.6 \cdot 10^{-7}$</td>
<td>Gigaflop/s</td>
<td>LINPACK 64-bit floating point</td>
</tr>
<tr>
<td>$2.5 \cdot 10^{-7}$</td>
<td>Gigabyte</td>
<td>Node memory</td>
</tr>
<tr>
<td>$3.8 \cdot 10^{-5}$</td>
<td>Gigabyte/s</td>
<td>Local network</td>
</tr>
<tr>
<td>$1.0 \cdot 10^{-2}$</td>
<td>Gigabyte/s</td>
<td>to Wide Area Network</td>
</tr>
<tr>
<td>$2.8 \cdot 10^{-2}$</td>
<td>Gigabyte/s</td>
<td>Skilled human labor</td>
</tr>
<tr>
<td>$1.0 \cdot 10^{-1}$</td>
<td>Gigabyte/s</td>
<td>to Public Internet</td>
</tr>
</tbody>
</table>
Try off-the-shelf OpenCV2 Hough Transform (default)
Try off-the-shelf OpenCV2 Hough Transform (best precision)
Try off-the-shelf OpenCV2 Hough Transform (best recall)
Introducing the Differential Template Object Finder (DTOF)

Motivation

– How do we take advantage of the power of Intel Processors with AVX-512 without depending indirectly on a stack of deep learning library developers to do it for us?

– Demonstrate a circle finder for satellite imagery in less than 100 lines of code.

– Write from scratch using Python, with kernels written in C, wrapped in Cython.

– Implement with vector intrinsics, for instant access to AVX (Haswell) and AVX-512 (Skylake) floating point vector instructions.
```python
import descarteslabs as dl
import numpy as np
import dtof

# Cushing, OK
scene = "airbus:oneatlas:v1:8192:0:17:947:2486"
scale = 4
tilesize = 128/scale
center = tilesize/2

img, meta = dl.raster.ndarray([scene], bands=['red','green','blue','alpha'])

fimg1 = (0.2126 * img[:, :, 0] + 0.7152 * img[:, :, 1] + 0.0722 * img[:, :, 2]).astype("float32") / 256
fimg1 = 256 * transform.pyramid_reduce(fimg1, scale).astype("float32")

x = img[0:tilesize, 0:tilesize]
yi, xi = np.where(np.ones_like(x[:, 0]))
rr = np.sqrt((xi-center)**2 + (yi-center)**2, dtype="float32")
```
# gaussian detector width
sigma = 1.0

for y0 in range(0, fimg1.shape[0]-tilesize+center):
    for x0 in range(0, fimg1.shape[1]-tilesize+center):
        out = np.zeros(center, dtype=np.float32)
        xx = fimg1[y0:y0+tilesize,x0:x0+tilesize].flatten()
        dtof.circles(xx, rr, out, sigma)

    delta = out[4:center-1]-out[5:center]
    delta_max_index = np.argmax(delta)
    delta_max = delta[delta_max_index]
    r = delta_max_index + 4
    out_max = out[r]
    fdelta = delta_max/out_max

if fdelta >= 0.12:
    print '%4d %4d %3d %.2f %.2f circle' %
    (scale*(x0+center), scale*(y0+center), scale*r, delta_max, fdelta)
import numpy as np
cimport numpy as np
cimport cython
cdef extern from "dlfind.h":
    void _circles(const float *x, int ii, const float *r, float *out, float sigma) nogil
cpdef circles(np.float32_t[:, :, 1] x, np.float32_t[:, :, 1] r, np.float32_t[:, :, 1] out, double sigma=1.0):
    """Find circles in an image""
    _circles(&x[0], x.shape[0], &r[0], &out[0], sigma)
    return None
Inner loops in C

```c
#include <math.h>

void _circles(const float *x, int xlen, const float *r, float *out, float sigma)
{
    float win = 1.0f/(2.0f*sigma*sigma);
    float winv = -win;

    for (int ir = 4; ir < xlen/2; ir++) {
        float r0 = (float)ir;
        float w = 0.0f;
        float q = 0.0f;
        for (int i = 0; i < xlen; i++) {
            float vx = x[i];
            float dr = r[i];
            dr -= r0;
            dr *= dr;
            dr *= winv;
            float weight = expf(dr);
            w += weight;
            q += weight * vx;
        }
        out[ir] = q / w;
    }
}
```
Inner loops in C (using vector intrinsics)

```c
#include <immintrin.h>

typedef __m256 v8sf; // vector of 8 float (avx)

void
__circles(const float *x, int xlen, const float *r, float *out, float sigma)
{
    float win = 1.0f/(2.0f*sigma*sigma);
    v8sf winv = Vscalar(-win);
    
    for (int ir = 4; ir < xlen/2; ir++) {
        v8sf r0 = Vscalar((float)ir);
        v8sf w = {}
        v8sf q = {}
        for (int i = 0; i < xlen; i += VLEN) {
            v8sf vx = Vload(x, i);
            v8sf dr = Vload(r, i);
            dr -= r0;
            dr *= dr;
            dr *= winv;
            v8sf weight = exp256_ps(dr);
            w += weight;
            q += weight * vx;
        }
        out[ir] = Vsum(q) / Vsum(w);
    }
}
```
DTOF Result:
If you’d like to learn more,

Data-Intensive Supercomputing in the Cloud: Global Analytics for Satellite Imagery

arXiv:1702.03935 [cs.DC]

https://medium.com/descartestech

https://www.descarteslabs.com

We’re hiring:

https://www.descarteslabs.com/jobs

Thank you!