Prototyping Mid-Infrared Detector Data Processing Algorithms

Dr. Michael Ressler

Principal Scientist

Jet Propulsion Laboratory, California Institute of Technology

Research reported in this presentation was conducted at the Jet Propulsion Laboratory (JPL), California Institute of Technology, under a contract with the National Aeronautics and Space Administration (NASA).

Infrared Detectors Generate Torrents Of Data

- Megapixel and larger infrared detector arrays in several technologies are available covering the 1 – 30 μm spectral range
- Particularly, in the thermal infrared (> 3 μm), data rates can be quite high due to background emission
- The challenge: in our particular case, each array has
  - 1024x1024 light sensitive pixels divided into 4 outputs
  - “Reference pixels” added to all 4 outputs; mimic "dark" detectors; located on the left and right edges of the image
  - A “reference output” that averages signal from 8 reference pixels; brought out through a 5th output channel
  - A data rate of 1290x1024 pixels in 2.75 seconds – 6.5 GB of lab data in 1 hr, largest to date is 11.8 GB

Data Reordering Requirements

Data is output serially – one frame is a string of 1,320,960 values.

1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5

Simple reformatting as a 1290x1024 2-D array interleaves reference data with optical data – produces "jail-bar" pattern and degrades compression ratio.

1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
1	2	3	4	5	1	2	3	4	5	1	2	3	4	5

Instead, we can strip out the reference data and pack it onto the bottom, producing a 1032x1280 pixel frame.

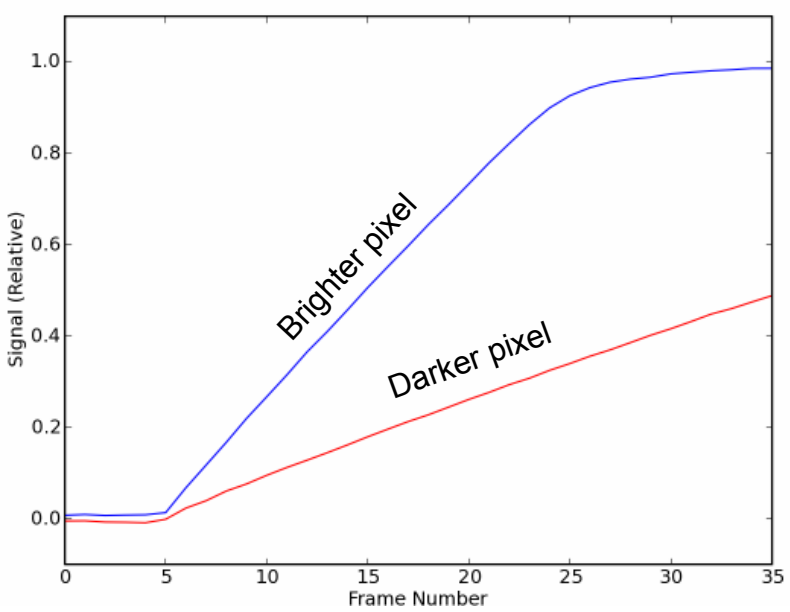
1	2	3	4	1	2	3	4	1	2	3	4
1	2	3	4	1	2	3	4	1	2	3	4
1	2	3	4	1	2	3	4	1	2	3	4
1	2	3	4	1	2	3	4	1	2	3	4
1	2	3	4	1	2	3	4	1	2	3	4
1	2	3	4	1	2	3	4	1	2	3	4
1	2	3	4	1	2	3	4	1	2	3	4
1	2	3	4	1	2	3	4	1	2	3	4
1	2	3	4	1	2	3	4	1	2	3	4
1	2	3	4	1	2	3	4	1	2	3	4
1	2	3	4	1	2	3	4	1	2	3	4
1	2	3	4	1	2	3	4	1	2	3	4
5	5	5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5	5	5

Reordering Code Snippets

```
# Do all the setup stuff, file name checking, options, etc.

# Open a memory map to the data, int32 data type
m=np.memmap(fn,mode='r',dtype='i')

# Grab a frame and reshape it
a=(m[fm_beg:fm_end].copy()).reshape(1024,1290)

# Set up an index to address individual channels
z=np.arange(0,1289,5)+n

# Grab the 5 channels, add offset to index to change
c0=np.take(a,z+0,axis=1) ; … ; c4=np.take(a,z+4,axis=1)

# Reshape each to 3-D to set up interleaving
c1a=c1.reshape(1024,258,1) ; … ; c4a=c4.reshape(1024,258,1)

# A bit of trickery to reinterleave the data
x=np.concatenate((c1a,c2a,c3a,c4a),axis=2).reshape(1024,1032)

# Remember the reference? Reshape to append to the bottom
y=c0.reshape(256,1032)

# Produce the final frame
z=np.concatenate((x,y),axis=0)

# Add it to the output cube and when finished, write to a FITS file
```

Fitting Slopes To All Pixels

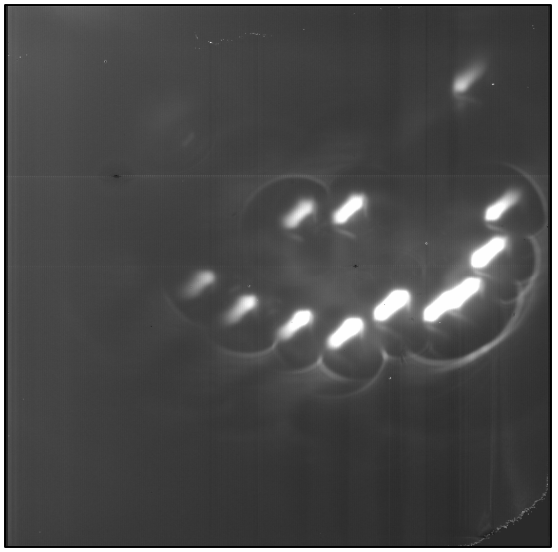
- Pixels integrate signal approximately linearly
- Sample the output of each pixel once per frame time
- Processed signal level (i.e. image) is obtained by fitting a slope to each of the > 1 million pixels




```
# Set up slope factor argument, t = time per frame, n = number of frames
arg=12./t/(n**3-n)

# Create summing array of zeroes
sum=np.zeros((ny,nx),dtype='f')

# Loop through the frames; multiply each frame by weight and then sum
for i in range(n) :
    sum=sum+(i-(n-1.)/2.)*frames[i,:,:].astype('f')

# Multiply by factor and return
return arg*sum
```

The Choice of Python + Numpy

- Needed the ability to explore “how to” process the data, not to just do it
  - C & Fortran – too long a development time
  - IDL – too expensive to run everywhere
- Numpy (and numarray) have a rich set of N-dim features
  - Concatenating, transposing, and reshaping with specific axes can produce complex dataset transforms
  - Trivial extractions allow the examination of individual frames or pixels from data cubes
- All preliminary processing software in python + numpy + matplotlib
  - More rigorous processing algorithms are now in IDL, but are informed by the python code



A weak attempt at a smiley face with an engineering-grade detector