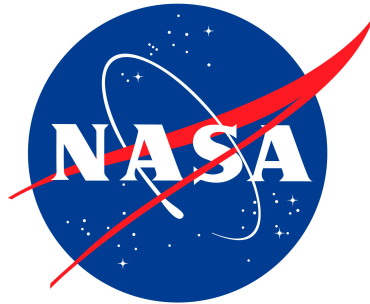


# Marshall Space Flight Center and the open-source radar software revolution

Timothy J. Lang



Brenda Dolan, Brody Fuchs, Paul Hein, Elizabeth Thompson



Scott Collis, Jonathan Helmus



Nick Guy



## **Acknowledgments**

Jason Burks, Steve Nesbitt,  
George Priftis, Brent Roberts  
Funding: NASA, DARPA, USDOJ

# Background

My radar software history ...

Graduate School – RDSS, Reorder, SPRINT, Fortran, and pltgks

Research Scientist – IDL, solo, Reorder, SPRINT

NASA Civil Servant (2012) – Stick with IDL or go Python?

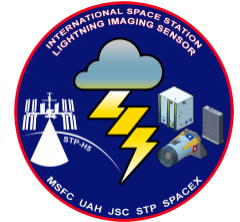
NASA Marshall's Lightning Group is mostly an IDL shop

But folks are receptive to Python and some want to learn!

I made the switch in early 2014

# MSFC Science Goals

1. Understand lightning production in extreme weather events – *GLM*, *ISS-LIS*



2. Validate precipitation estimates from space – *GPM*



3. Study how well scatterometers characterize convective variability – *RapidScat*, *CYGNSS*



Open-source radar software, powered by Python, plays a prominent role in all of these tasks!

# Marshall MRMS Mosaic Python Toolkit (MMM-Py)

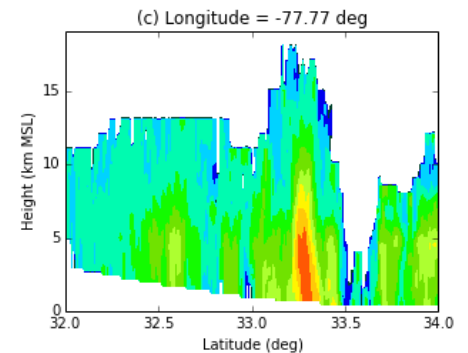
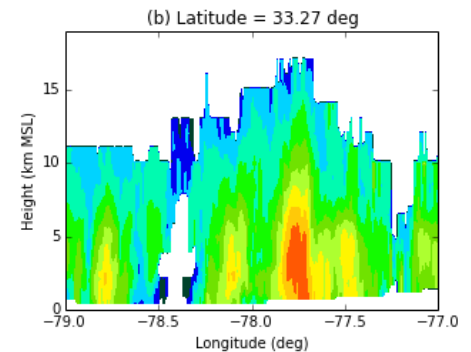
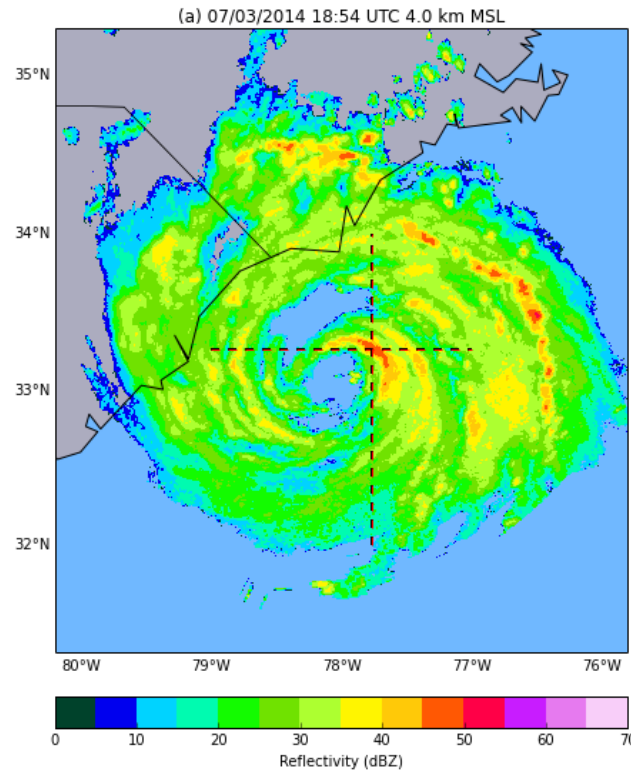
<https://github.com/nasa/MMM-Py>

## Goal

Simplify the ingest, analysis, and display of NOAA MRMS 3D radar reflectivity mosaics in a Python environment

## Features

- Can read any format MRMS, from 2009 onward
- Easy tile merging and domain subsetting
- Customizable plotting methods
- Save custom mosaics to file
- Demonstration Jupyter notebooks available





# Python Turbulence Detection Algorithm (PyTDA)

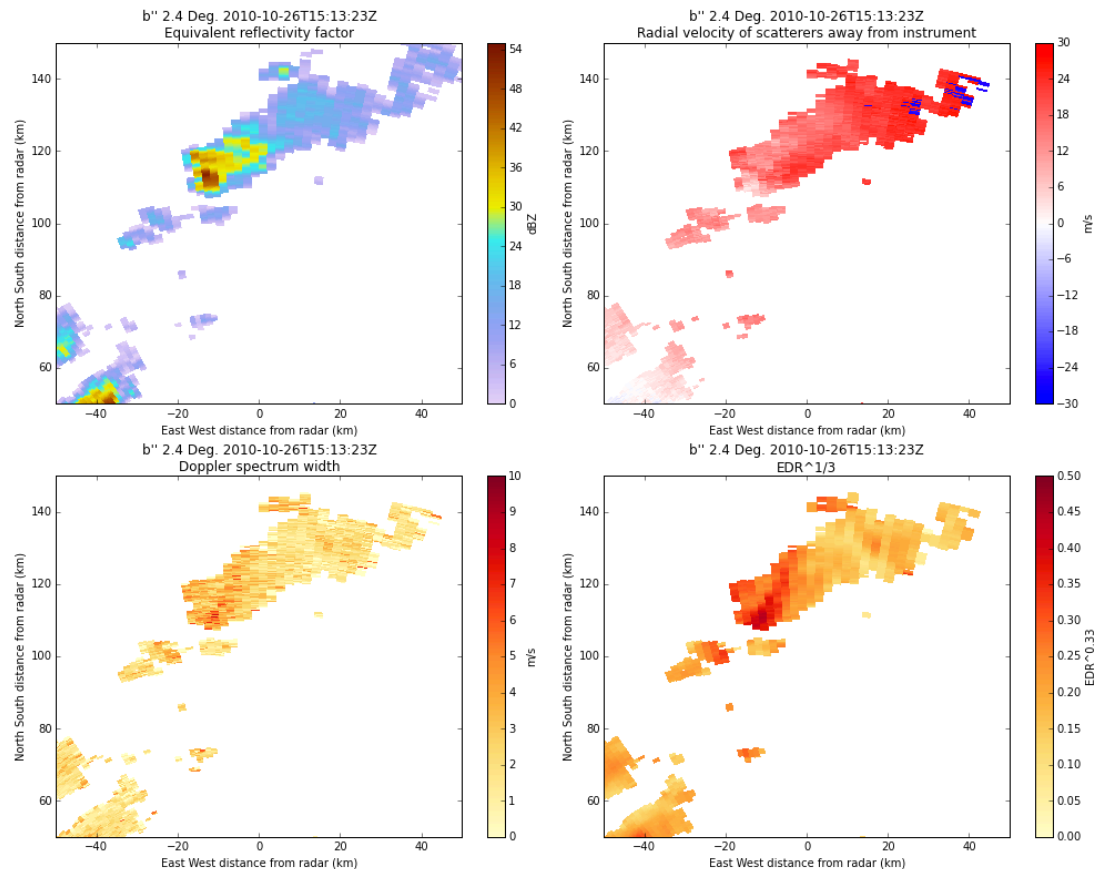
<https://github.com/nasa/PyTDA>

## Goal

Estimate eddy dissipation rate (EDR) from arbitrary Doppler radar sweep or volume

## Features

- Module contains independent functions for processing
- Works seamlessly with Py-ART Radar object
- NTDA-like filtering and quality control of data if desired
- Uses sklearn trees, NumPy function broadcasting, and some Cython to allow near real-time use on a laptop



# Single Doppler Retrieval Toolkit (SingleDop)

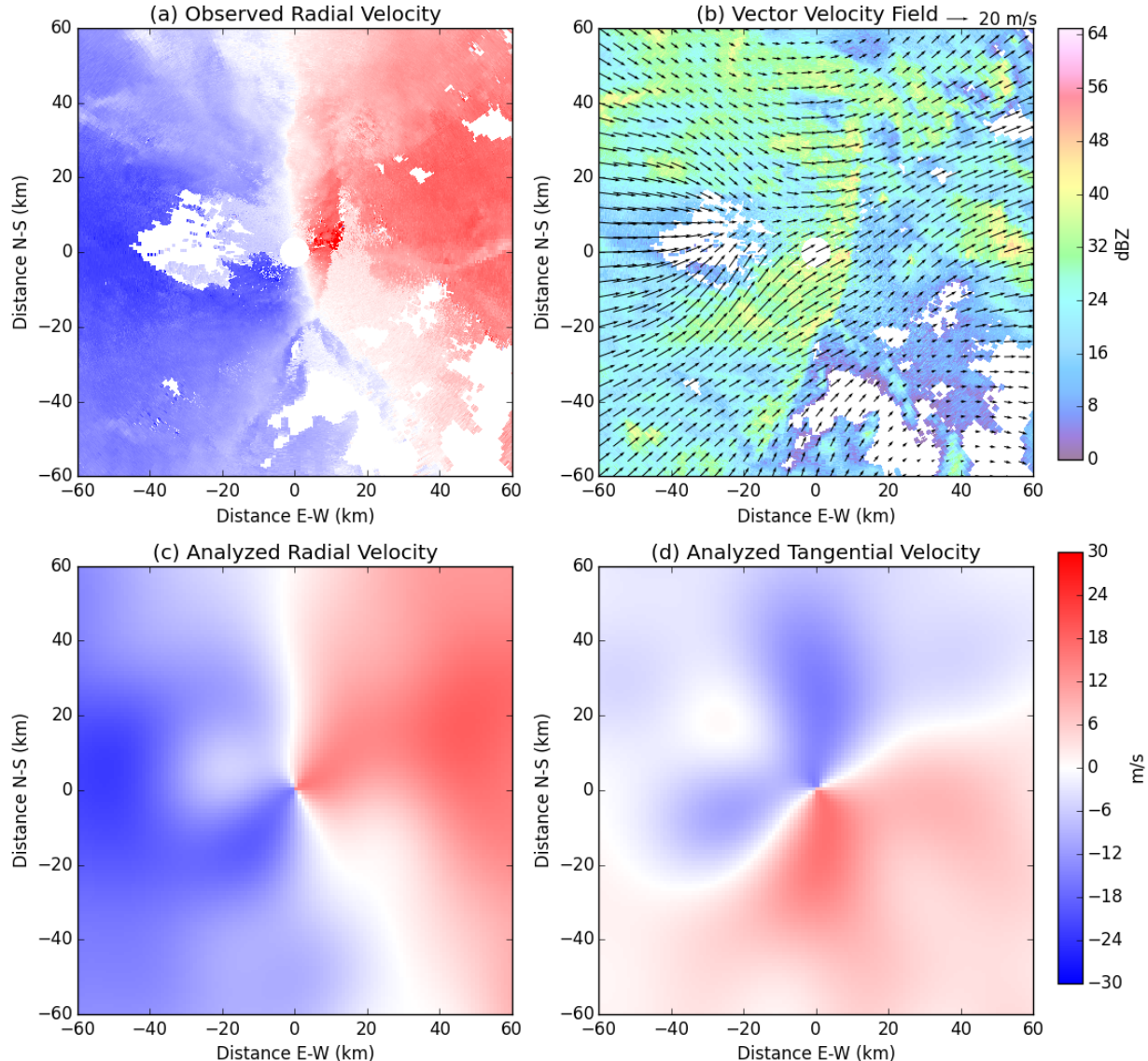
<https://github.com/nasa/SingleDop>

## Goal

Estimate low-level 2D winds from single Doppler radar

## Features

- Xu et al. (2006) method
- Works with real or simulated radar data
- Background field can be specified or determined via VAD
- Uses Py-ART for data input and display
- Output gridded analyses to Python-readable binary (pickle) or netCDF (xray)
- Contour, vector, and mixed multi-panel plots supported



# Colorado State University Radar Tools (CSU\_RadarTools)

[https://github.com/CSU-Radarmet/CSU\\_RadarTools](https://github.com/CSU-Radarmet/CSU_RadarTools)

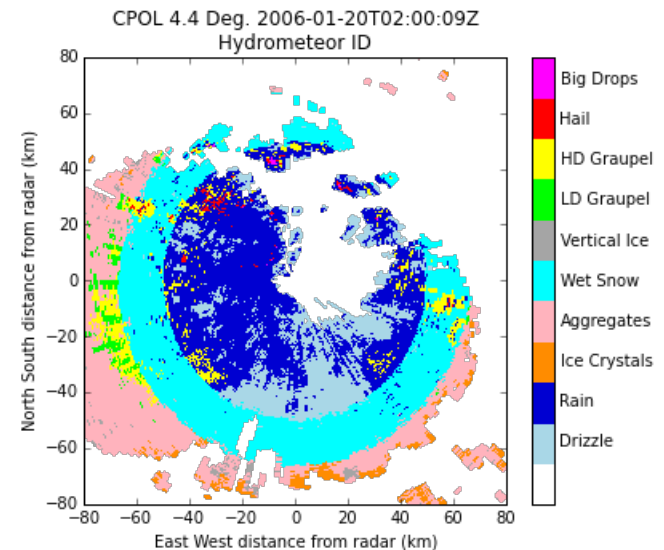
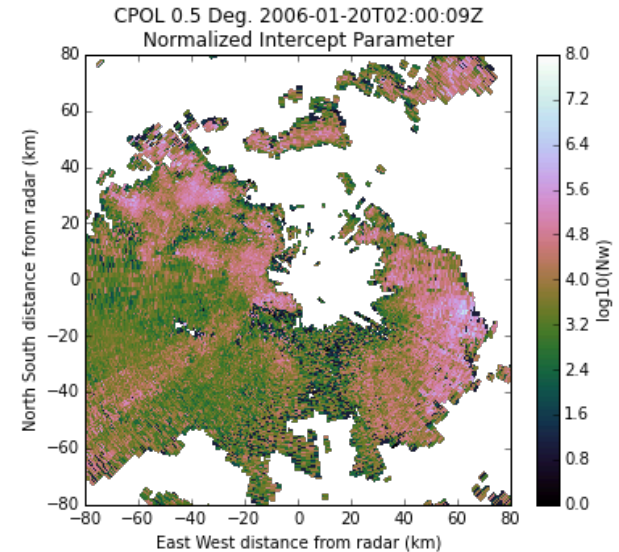


## Goal

Provide routine polarimetric radar analysis and QC tools originally developed/used at CSU

## Features

- Independent of any other software framework
- Works with ndarrays or scalars
- CSU Fuzzy-Logic Hydrometeor ID (X, C, S-band)
- CSU Blended Rainfall Calculations
- CSU Ice and Liquid Water Mass calculations
- CSU Drop-Size Distribution calculations
- CSU KDP calculations (FIR based)
- Miscellaneous QC (insect filter, despeckling, etc.)
- Each works as importable sub-module
- Thorough demo notebook available, shows how to use with SkewT and Py-ART.



# Python Interface to Dual-Pol Radar Algorithms (DualPol)

<https://github.com/nasa/DualPol>

## Goal

Simple workflow for merging  
CSU\_RadarTools with Py-ART

## Features

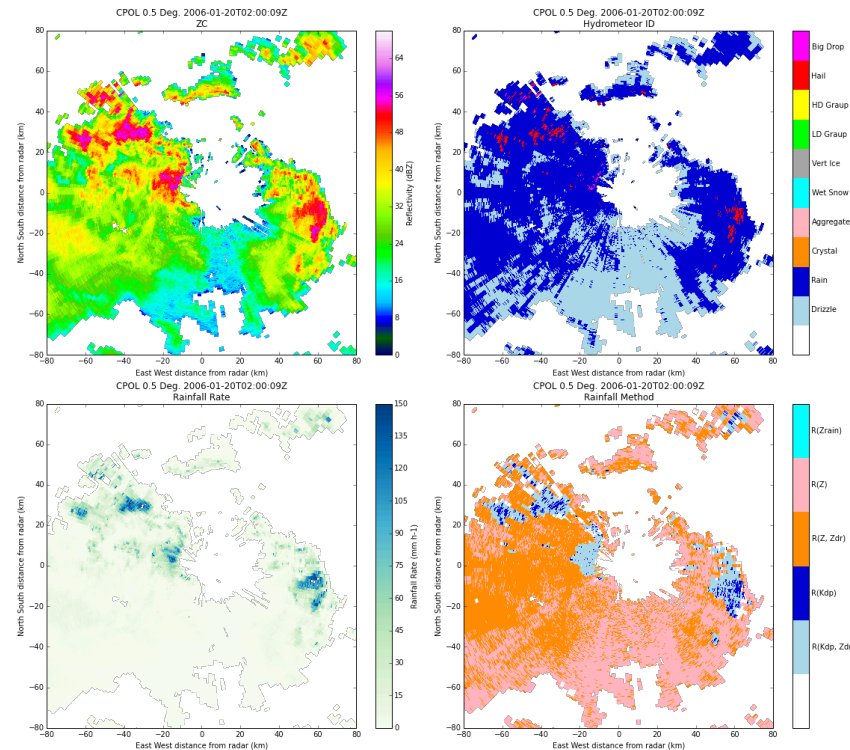
- One-line implementation for dual-pol retrievals and integrating them within Py-ART radar object
- Class that simplifies display of HID & rain method colorbars

```
In [4]: sndfile = '/Users/tjlang/Documents/OVST/CPOL/soundings/snd_Darwin.txt'
retrieve = dualpol.DualPolRetrieval(files[0], dz='ZC', dr='ZD', kd='KD', rh='RH',
use_temp=True, band='C', fhc_method='hybrid',
sounding=sndfile, fhc_T_factor=2,
ice_flag=True, rain_method='hidro')
```

```
In [5]: print retrieve.radar.fields.keys()

[u'DC', u'DZ', 'FH', u'FL', u'HD', u'PF', u'PH', u'RH', 'method', 'NW', u'AD', u'ZD', u'AH',
'rain', u'ZC', u'VR', u'KD', u'SR', 'MI', 'MU', 'MW', 'D0']
```

```
In [11]: display = pyart.graph.RadarDisplay(retrieve.radar)
swp = 0
lim = [-80, 80]
fig = plt.figure(figsize=(16, 14))
ax1 = fig.add_subplot(221)
display.plot_ppi('ZC', swp, vmin=0, vmax=70, cmap='gist_ncar')
display.cbs[0].ax.set_ylabel('Reflectivity (dBZ)')
display.set_limits(xlim=lim, ylim=lim)
ax2 = fig.add_subplot(222)
hidcolor = dualpol.HidColors()
display.plot_ppi('FH', swp, vmin=0, vmax=10, cmap=hidcolor.cmaphid)
display.cbs[1] = hidcolor.adjust_fhc_colorbar_for_pyart(display.cbs[1])
display.set_limits(xlim=lim, ylim=lim)
ax3 = fig.add_subplot(223)
display.plot_ppi('rain', swp, vmin=0, vmax=150, cmap='GnBu')
display.set_limits(xlim=lim, ylim=lim)
ax4 = fig.add_subplot(224)
display.plot_ppi('method', swp, vmin=0, vmax=5, cmap=hidcolor.cmameth)
display.cbs[3] = hidcolor.adjust_meth_colorbar_for_pyart(display.cbs[3])
display.set_limits(xlim=lim, ylim=lim)
plt.tight_layout()
```





# Python Polarimetric Radar Beam Blockage Calculation (PyBlock)

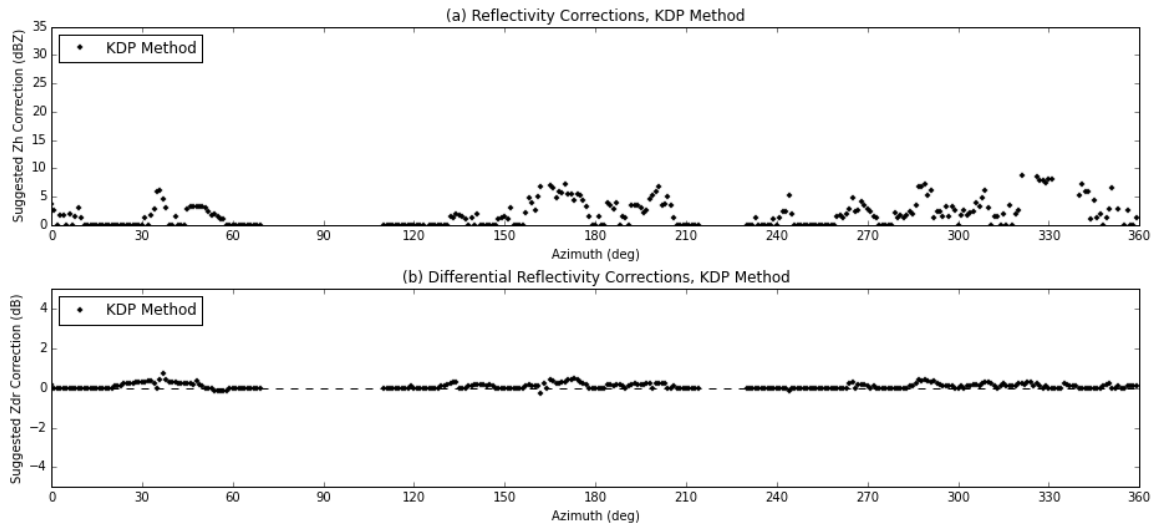
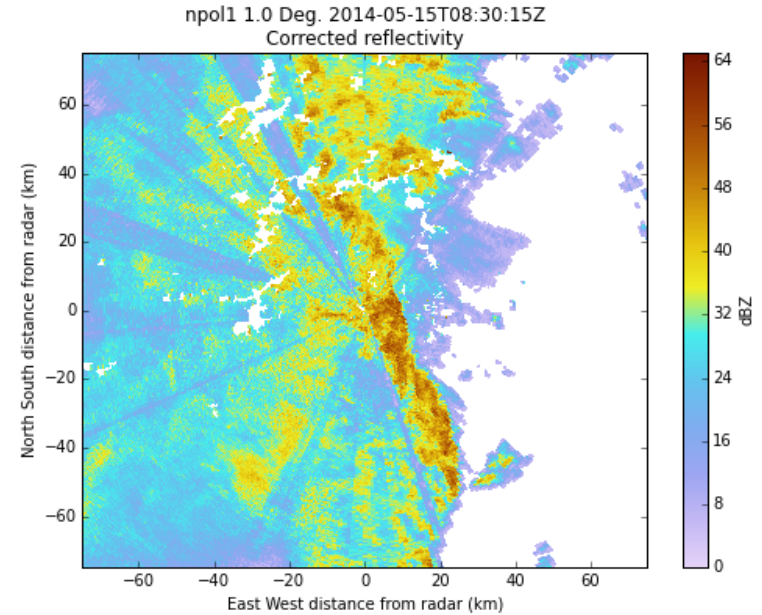
<https://github.com/nasa/PyBlock>

## Goal

Calculate beam blockage and estimate corrections needed for polarimetric radar data

## Features

- KDP Method from Lang et al. (2009)
- Fully Self-Consistent Method from Giangrande and Ryzhkov (2005)
- Interfaces w/ DualPol, Py-ART, CSU\_RadarTools
- Process one volume or an entire field campaign's worth of radar volumes
- Output results to image or file
- Demonstration notebook



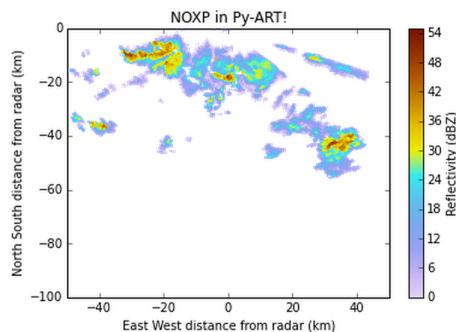
# Collaborations

## Py-ART Contributions

- Color blindness friendlier rainbow color table
- NOXP radar data reader
- Automated dealiasing testing (Poster session #3 today)

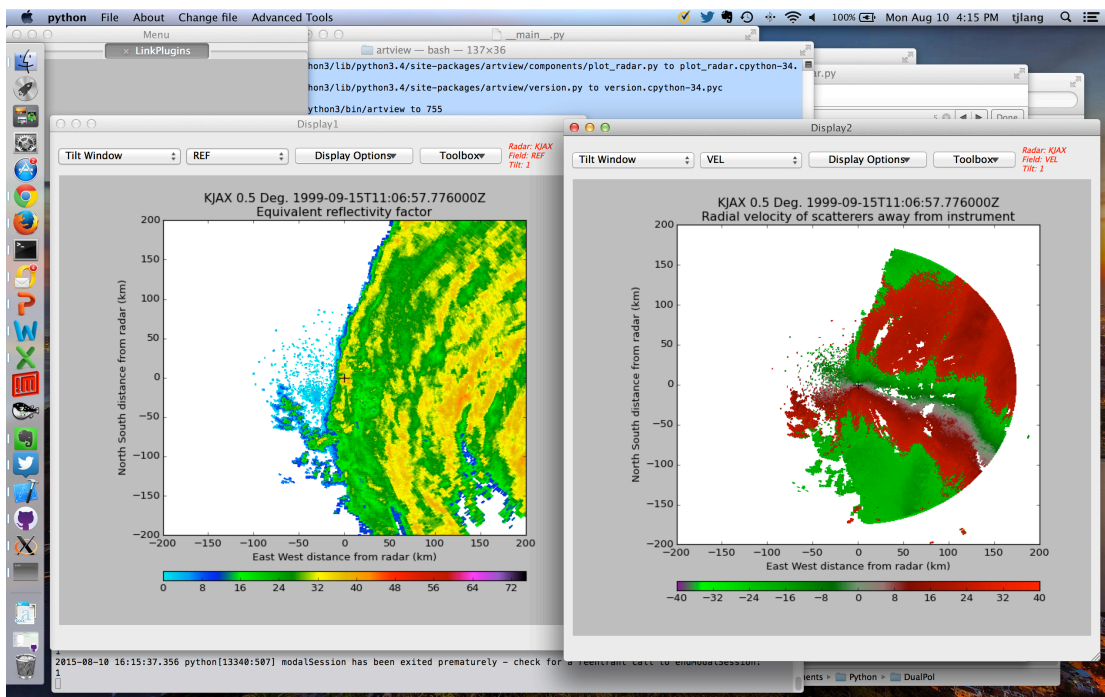


```
In [11]: radar = pyart.aux_io.read_noxp_iphex_nc(files[55], exclude_fields=['m1', 'm2'])
display = pyart.graph.RadarDisplay(radar)
display.plot('reflectivity', vmin=0, vmax=55, cmap='pyart_LangRainbow12',
            title='NOXP in Py-ART!', colorbar_label='Reflectivity (dBZ)')
lim = [-50, 50]
display.set_limits(xlim=lim, ylim=[-100, 0])
```



## ARTview Contributions

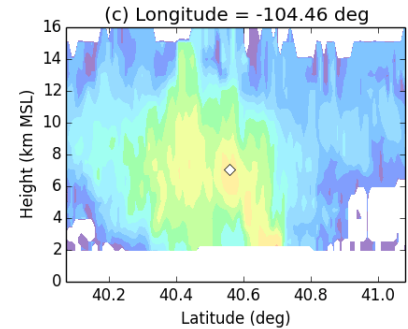
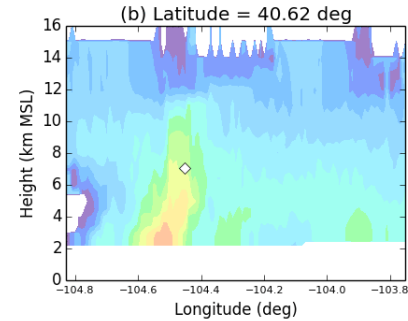
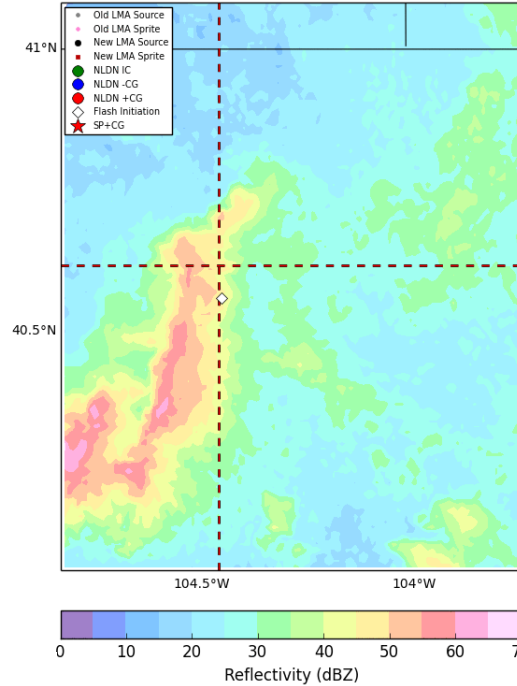
- Minor feature additions
- Code cleanup/bug fixes
- Testing and evaluation



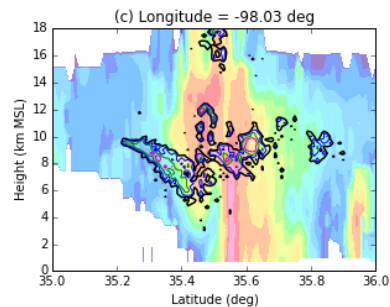
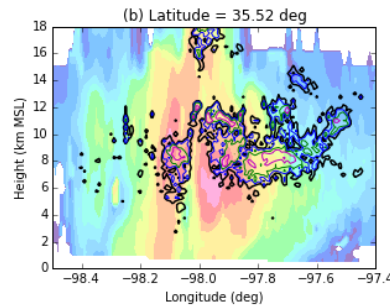
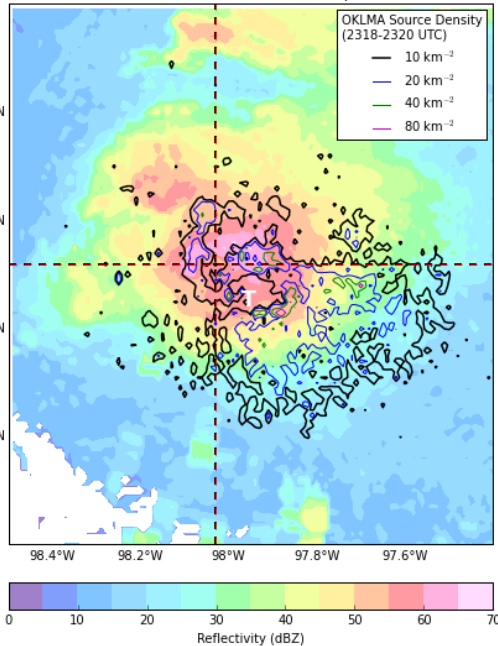
# Putting it all together ...

MMM-Py combined with in-house Python modules that merge MRMS and Lightning Mapping Array (LMA) data to study storms that produce large charge moment change lightning and/or sprites

6/8/2012, 05:05:06.443 UTC, Elapsed Time = 0.00 sec



(a) 05/31/2013 23:15 UTC Composite



- Colorado sprite-producing storm (2012)
- MRMS w/ animated Colorado LMA sources

- El Reno storm (2013)
- Tornadic Stage
- MRMS w/ Oklahoma LMA source densities

# Putting it all together ...

IPHEX Field Campaign Data  
Fusion in support of GPM GV

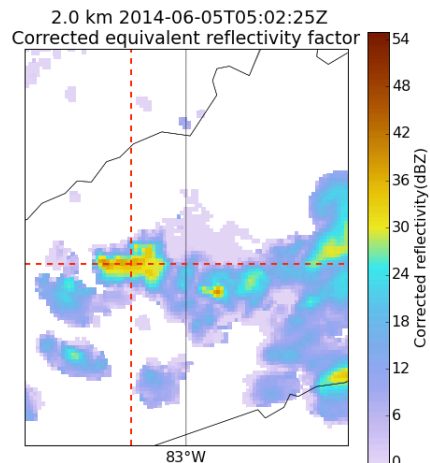
## Radars

Mixture of **research** and ops:  
**NPOL**, **NOXP**, KCAE, KGSP,  
KHTX, KMRX, KRAX, KFFC

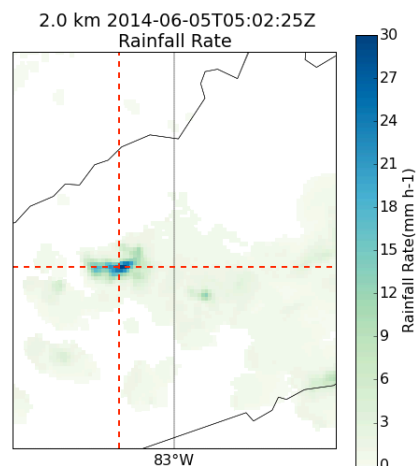
## Tasks

- Py-ART to ingest and merge radar volumes onto common grid
- DualPol to compute rainfall and DSD parameters
- CSU\_RadarTools to mask insect echoes & high differential phase texture, and despeckle remainder

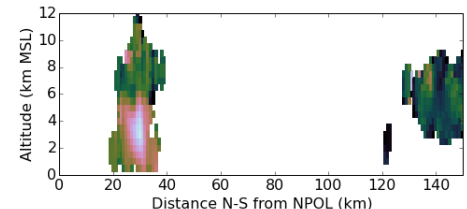
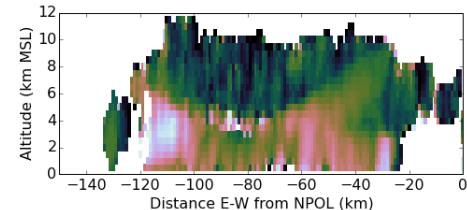
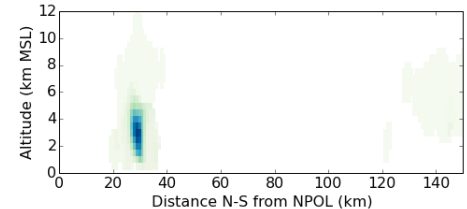
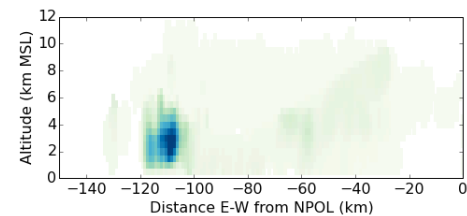
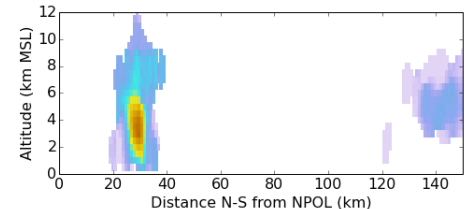
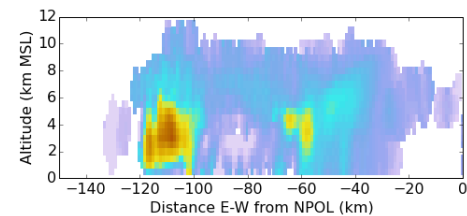
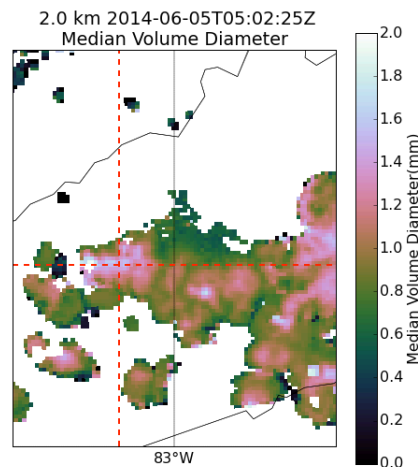
Z



R



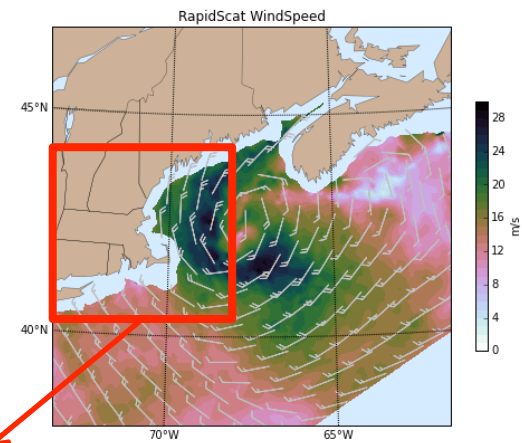
$D_0$



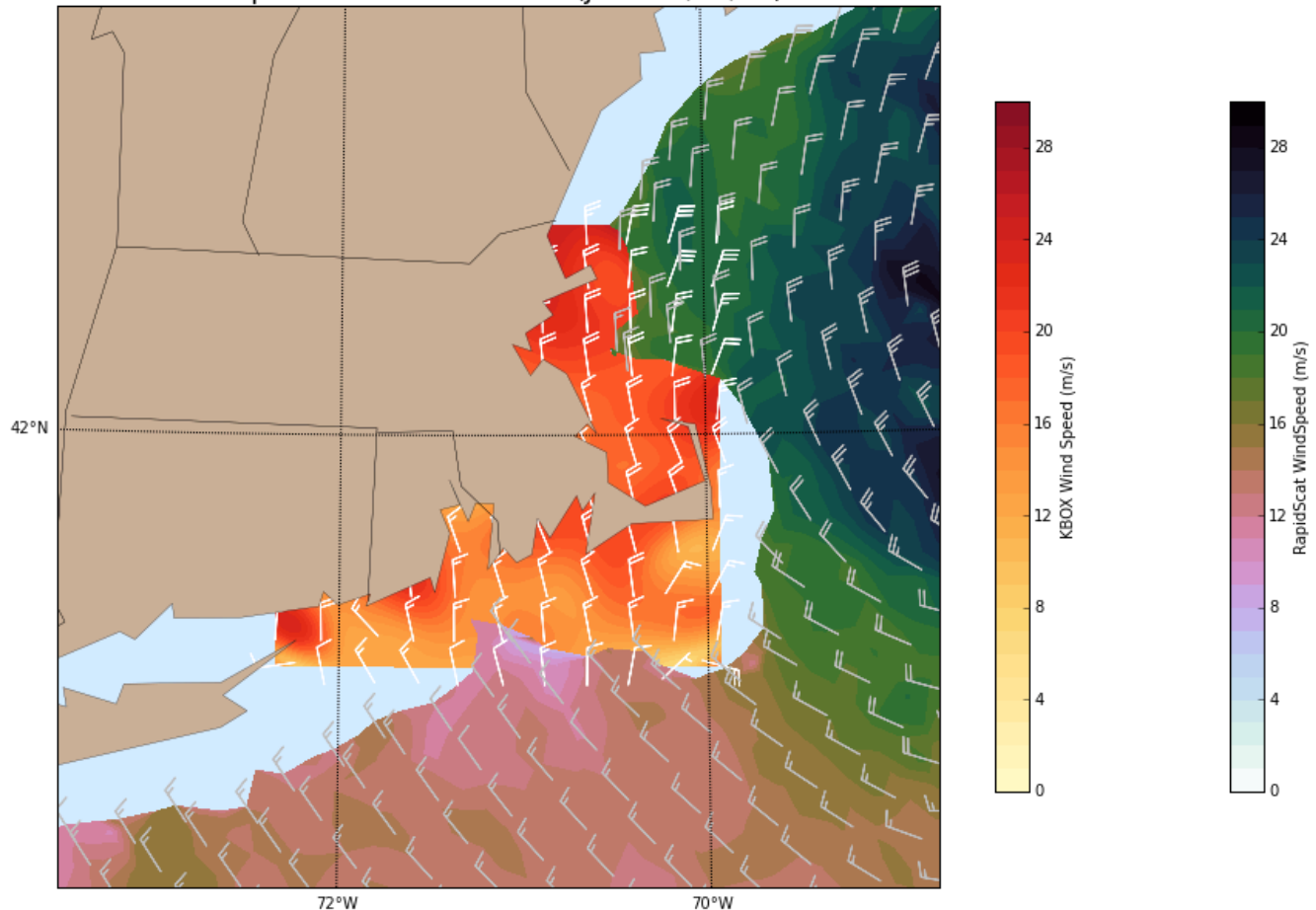


# Putting it all together ...

Exploring the best ways to compare scatterometers to ground-based Doppler radars



RapidScat vs. KBOX Radar (Juno - 1/28/15)



## Tasks

- Pydap to ingest RapidScat data via OPeNDAP
- Py-ART to ingest radar data and dealias velocity
- SingleDop to perform 2D low-level wind retrievals
- Also: DualPol, CSU\_RadarTools

# Summary

## Six Radar Modules

1. MMM-Py
2. PyTDA
3. SingleDop
4. CSU\_RadarTools
5. DualPol
6. PyBlock

Open source, play nice with Py-ART, support Python 3, & are available on GitHub right now (nasa & CSU-Radarmet)

*Contact Info:*

timothy.j.lang@nasa.gov



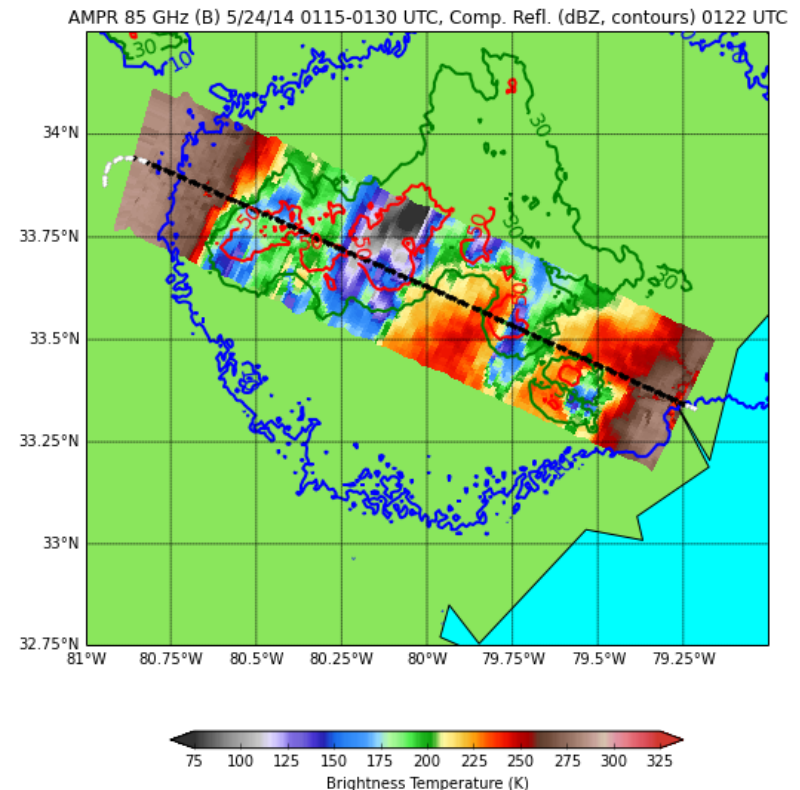
**Timothy Lang**

@tjlang

Just completed a preliminary case study using Python 3 @Py\_ART, ARTview ([zenodo.org/record/27358#...](https://zenodo.org/record/27358#...)), and SingleDop ([github.com/nasa/SingleDop](https://github.com/nasa/SingleDop))

8/11/15, 10:53 AM

+ PyAMPR!



# The Future

Major coding done for now, but some possible ideas for directions to go:

- Merge into MSFC\_RadarTools?
- Integration of components within Py-ART or wradlib?
- Make part of some future scikits-radar?
- NASA airborne radar module or interface?

*But for now ... <http://code.nasa.gov> takeover!*

The screenshot shows the 'code.nasa.gov' website with a search bar and a list of software projects. The projects listed are:

- Python Turbulence Detection Algorithm (PyTDA)**  
<https://github.com/nasa/PyTDA>  
This software provides Python functions that will estimate turbulence from Doppler radar data.  
NASA Open Source 3.0
- Python Advanced Microwave Precipitation Radiometer Data Toolkit (PyAMPR)**  
<https://github.com/nasa/PyAMPR>  
Download AMPR data from <http://ghrc.nsstc.nasa.gov>. AMPR brightness temperature data from NASA field projects are in ASCII format. This python script defines a class that will read in single file from an individual aircraft flight and pull out timing, brightness temperatures from each channel, geolocation, and other information and store them as attributes using numpy arrays of the appropriate type. The approach is to ingest the entire file as a text string and then parse and type convert as necessary. The file is read and the data are populated when the class is instantiated with the full path and name of an AMPR file. Numerous visualization methods are provided, including track plots, strip charts, and Google Earth KMZs. In addition, polarization deconvolution is available.  
NASA Open Source 3.0
- Marshall MRMS Mosaic Python Toolkit (MMM-Py)**  
<https://github.com/nasa/MMM-Py>  
The National Oceanic and Atmospheric Administration (NOAA) regularly produces national 3D radar reflectivity mosaics

The screenshot shows the details for two software projects on the code.nasa.gov website:

- Python Polarimetric Radar Beam Blockage Calculation (PyBlock)**  
<https://github.com/nasa/PyBlock>  
This Python package will calculate beam blockage in polarimetric weather radar data using the specific differential phase (KDP) and fully self-consistent (FSC) methods of Timothy J. Lang et al. (2009; J. Atmos. Oceanic Technol.).  
NASA Open Source 3.0
- Python Interface to Dual-Pol Radar Algorithms (DualPol)**  
<https://github.com/nasa/DualPol>  
Python module that facilitates precipitation retrievals (e.g., hydrometeor type, precipitation rate, precipitation mass, particle size distribution information) from polarimetric weather radar data.  
NASA Open Source 3.0