The Lidar-Radar Open Software Environment (LROSE) : An Overview

Wen-Chau Lee
NCAR

Michael M. Bell (PI)
Colorado State University

Mike Dixon
NCAR

LROSE is supported by NSF SI2-SSI Award ACI-1661663

What problem are we trying to solve?

- We have a large code base of software, of varying ages and maintainability.
- We have inherited data formats that are not optimal for scientific data exchange.
- The scientific community has needs that are not supported by our current software - Many new software have been developed by the community but the overhead is high for others to adopt and/or adapt.
- NCAR has assumed the responsibility and supported radar/lidar software for the scientific community since late 1970s, but NCAR found it difficult to provide good quality support for aging legacy applications and the continuing needs for developing new capabilities.
- NCAR envisioned a new paradigm to encourage user community to jointly develop and support future software needs together with NCAR, e.g., the WRF “model”.

Why Do We Need “Radar Data *Exchange* Format”?

- Radar manufactures like to record the data using their own format for various reasons: Efficiency (in early years), Unique features (like airborne radars), Proprietary (“Company XXX Format”) – more than 25 radar formats are currently used in the community
- Multiple Doppler radar analyses using data recorded with different native formats
- Community tried to unify radar format with limited success
- “Radar data exchange format” is a compromise of the reality in the research community from the beginning and has been adopted by the operational and data assimilation communities

History of “Radar Data Exchange Format”?

- Universal Format (UF) – 1979
  - RDSS: First interactive radar perusal and editing software on VAX, use command line operation to control RAMTEC graphic display
- Modified UF for NOAA P3 tail airborne Doppler radar ~ 1986
- Doppler Radar Data Exchange (DORADE) for dual-polarization and airborne Doppler radar – 1991
  - Solo: unix, X-window based
- Hierarchical Data Format version 5 (HDF5) – 2008
- Climate Forecast complied NETCDF (CfRadial) - 2011

NSF Radar Workshop: 27-29 November 2012

- Workshop summary: Bluestein et al. (2014) BAMS
- Specifically discuss Radar Analysis and Software requirements (~120 participants)

FIG. 1. Small discussion group (from left to right: John Williams, Bob Palmer, Jim Wilson, and Phil Chilson) during the radar analysis and software session. (Courtesy of H. Bluestein.)

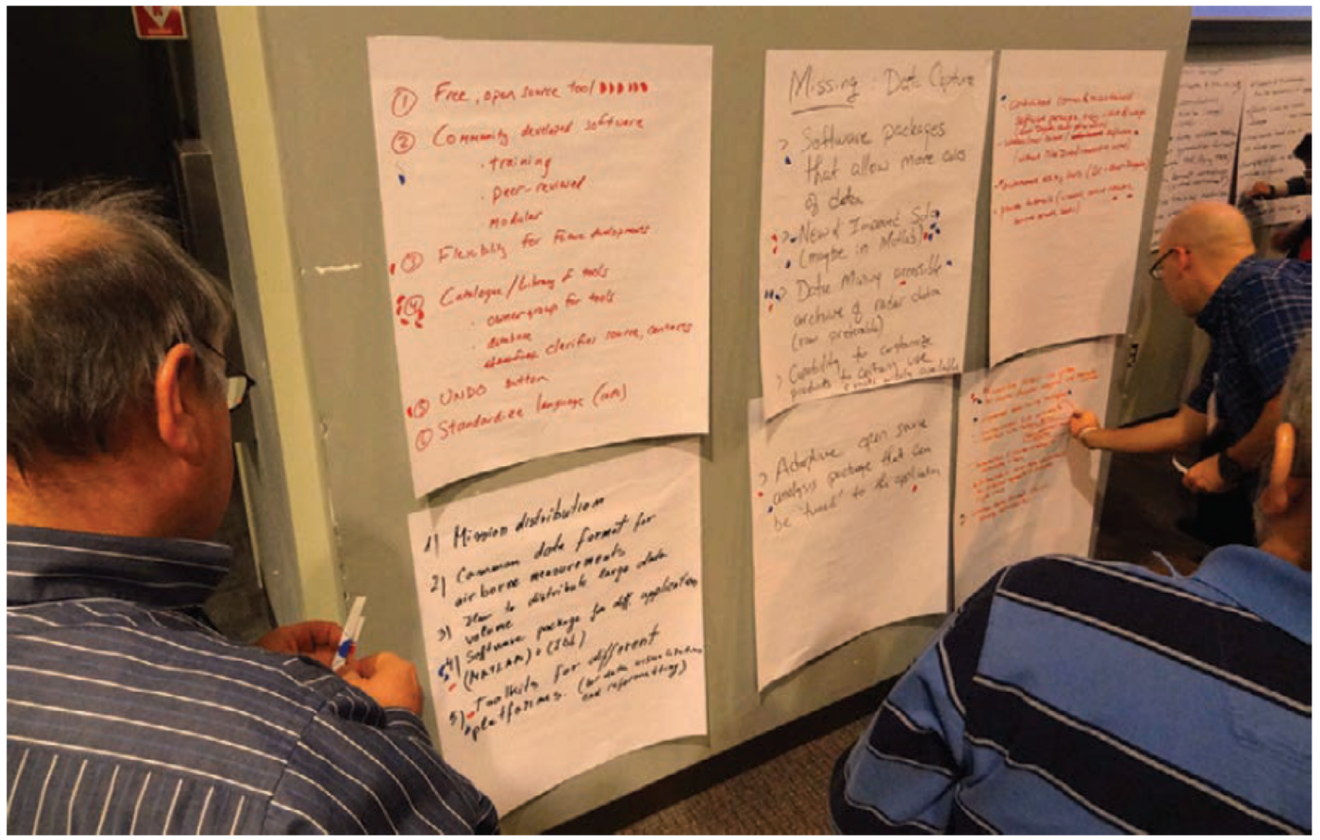


FIG. 2. Participants (Dave Jorgensen on the left; unidentified on the right) checking lists of priorities posted on the walls (gallery walk) during radar analysis and software session. (Courtesy of H. Bluestein.)

Radar Software Priorities From Workshop Participants

Radar Software Needs	Personal Research Needs	Community Needs	Total Score
NCAR-maintained centralized repository for radar software (esp. including wind synthesis) with data sets for software testing	55	41	96
Standardized software packages and toolkits (multi-platform, modular, menu-driven, easy for community to add to, ease of conversion among new and old radar data formats)	30	53	83
Training (workshops/online tutorials)	48	15	63
Ability to integrate radar and non-radar data sets	25	32	57
Open source tools and software	30	16	46
3D/4D Visualization Software (with publication quality output)	24	21	45
Next generation wind synthesis software to replace the legacy (REORDER/CEDRIC) algorithms, while maintaining current functionality	15	27	42
Common radar data format standard and a common metadata standard (e.g. CfRadial)	19	15	34
64-bit compatible real-time display software tool	19	11	30
Improved radar data quality control (solo) (Oye et al., 1995)	12	20	32
Automated quality control software	14	13	27
Detailed documentation for data products, tools, and code	18	7	25
Improved dual-polarization processing	10	12	22
Accessible variational Doppler radar assimilation and thermodynamic retrieval	7	4	11
Totals	326	287	613

What is LROSE?

- NCAR provided the initial funding support for the design study and prototype development
- Joint 4-year project between Colorado State University and NCAR Earth Observing Laboratory funded by NSF SI2-SSI
- **LROSE** will develop common software for the LIDAR, RADAR and PROFILER community.
- It is based on **collaborative, open source** development. The code would be freely available on the web - nsf-lrose.github.io.
- **The core**: developed by NCAR/EOL, largely based on existing code.
- **Algorithms and analysis tools:** jointly developed and supported by the community.
- **Data** would be stored in portable data formats, based on UNIDATA NetCDF, called CfRadial.
- GitHub open source collaboration (github.org)
- NCAR maintains the centralized code repository

Infrastructure

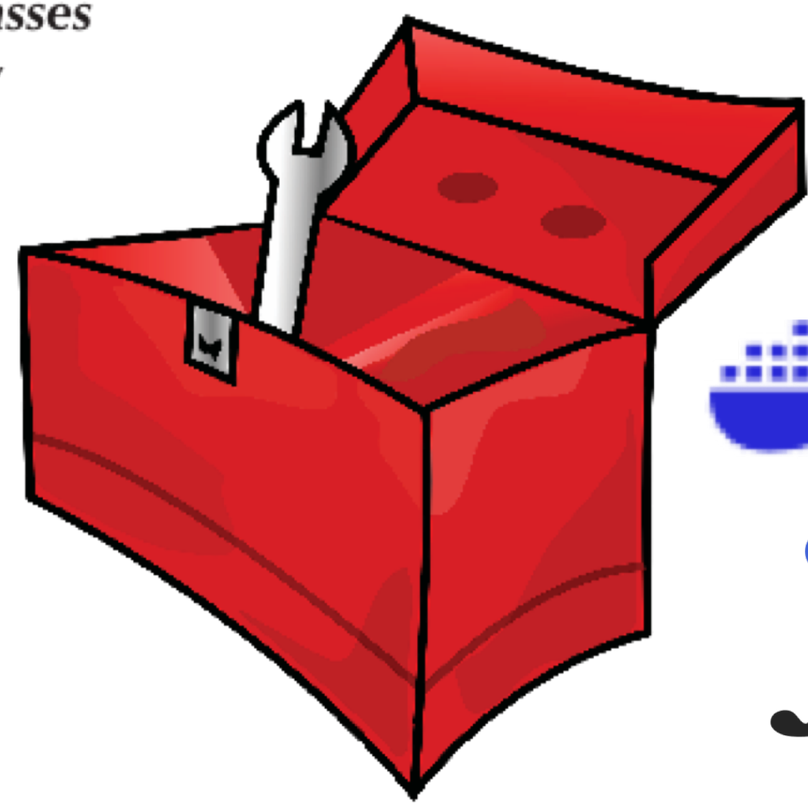
NetCDF data support layer
Data servers for display applications
Higher-level language bindings for C++ library classes
Higher-level language native data handling library
Portability layer
Open source software distribution mechanism

Displays

ASCOPE for spectral radar I/Q time series
BSCAN for vertically pointing data
Low-level viewer and editor for polar data
Platform-independent viewer for data integration
Display and editor tool for profiler data

Core Suite Algorithms

Well-tested, published algorithms
Common tasks and widely used tools



Community Algorithms

Specialized software toolsets
Configurable modules for different instruments and science goals
Partner-maintained packages (DOE PyArt, BALTRAD, and others)

April 11-12 2017 LROSE Workshop Summary

1. Six key applications were identified for initial development: Convert, Display, QC, Grid, Echo, and Winds

2. Good documentation, ease of use, and starter kits were emphasized as crucial aspects for software adoption, not just the availability of code or algorithms

3. Community contributions beyond code such as discussion forums, how-to videos, and tutorials will help build user base

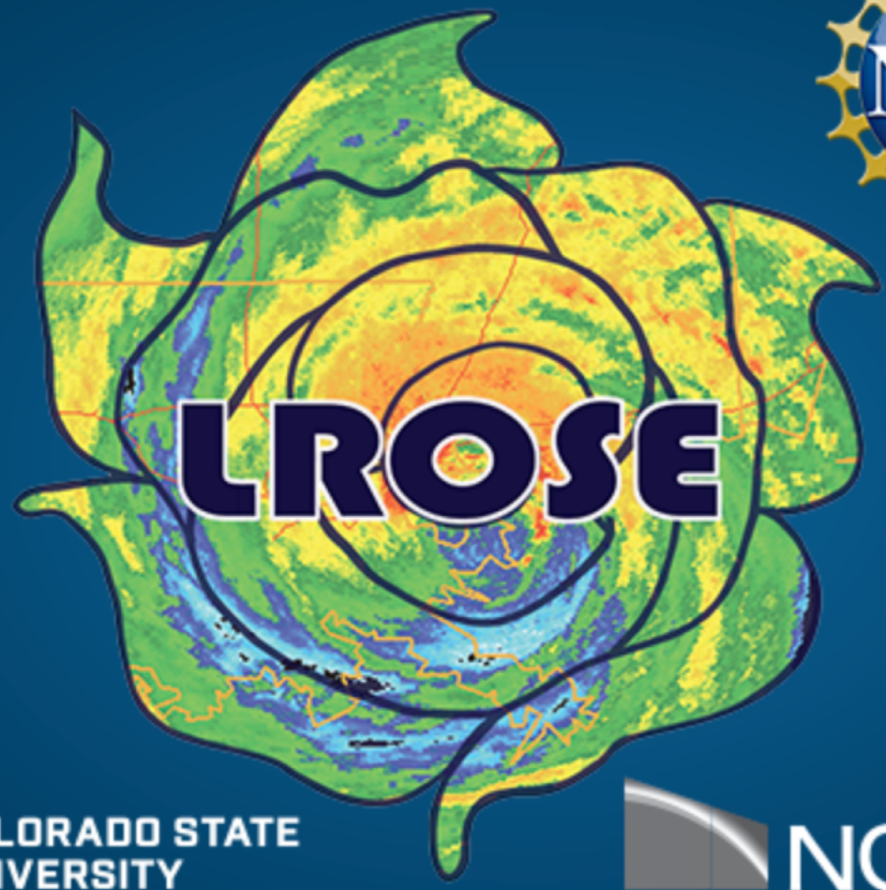
4. Focus on high-quality, well-tested, well-maintained and well-documented key applications as ‘building blocks’, allowing users to assemble trusted, reproducible workflows to accomplish more complex scientific tasks

LROSE

Home | Software | Documentation | Help

LROSE: LIDAR RADAR OPEN SOFTWARE ENVIRONMENT

LROSE is an National Science Foundation (NSF) supported project to develop common software for the Lidar, Radar, and Profiler community based on collaborative, open source development. The core package is being jointly developed by Colorado State University (CSU) and the Earth Observing Laboratory at the National Center for Atmospheric Research (NCAR/EOL)



NSF

COLORADO STATE UNIVERSITY

NCAR
NATIONAL CENTER FOR ATMOSPHERIC RESEARCH

NSF Supported Project

LROSE is based on collaborative open-source development, supported by a 4-year SI2-SSI grant #1550597 from NSF to CSU and NCAR/EOL.

Current Release

"Blaze"

Latest 2018-06-27

Gitter User Forum

Sign-up for Mailing List

GitHub

https://nsf-lrose.github.io/

Planned LROSE Releases (2018-2022)

LROSE development plan and schedule

Year	Major release name	Looks like ...	Contents for each release
2018	Blaze		1. Format conversion 2. Cartesian transformation 3. QPE 4. Doppler winds analysis
2019	Cyclone		To be announced
2020	Elle		To be announced
2021	Jade		To be announced
2022	Topaz		To be announced

Agenda for the LROSE Mini Workshop

- 1:00 - 1:15: LROSE Overview and current status - Wen-Chau Lee
- 1:15 - 1:30: "Blaze" Convert and Grid toolsets - Ting-Yu Cha
- 1:30 - 1:45: "HawkEye" Display tool - Brenda Javornik
- 1:45 - 2:00: LROSE Basic Dual-polarization applications - Mike Dixon
- 2:00 - 2:15: "Blaze" Single and Multi-Doppler Wind toolsets - Michael Bell
- 2:15 - 2:30: "Cyclone" release and future plans - Michael Bell
- 2:30 - 4 pm: Open discussion of capabilities, plans, and progress

Other LROSE Team Members: Bruno Melli, Erik Johnson, Jen DeHart, Ya-Chien Feng
Dave Albo, Arnaud Dumont

Questions?

"Blaze" Convert and Grid toolsets

RadxConvert, RadxPrint, Radx2Grid

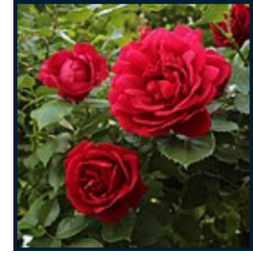
Ting-Yu Cha

27 June 2018
Blaze Alpha released!

"Blaze" (a climbing rose) is the 2018 release of the LROSE project. Check out our Software page for more detials.

01/06/2019 LROSE Mini workshop @ AMS 99th

A Basic LROSE-Blaze Workflow



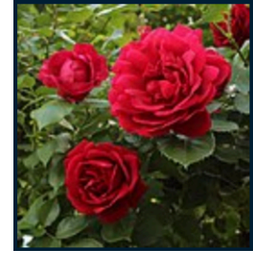
1. Print data header from a NEXRAD file using **RadxPrint**.

2. Convert Level II data to CfRadial format using **RadxConvert**.

3. Display the data in CfRadial format using **Hawkeye**.

4. Perform coordinate transformations from the polar grid to a Cartesian grid using **Radx2Grid**.

A Basic LROSE-Blaze Workflow



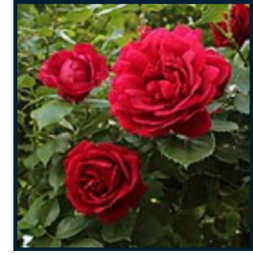
✓ 1. Print data header from a NEXRAD file using **RadxPrint**.

✓ 2. Convert Level II data to CfRadial format using **RadxConvert**.

3. Display the data in CfRadial format using **Hawkeye**.

✓ 4. Perform coordinate transformations from the polar grid to a Cartesian grid using **Radx2Grid**.

Radx application



To see all command line options for a Radx application:

```
RadxPrint -h
```

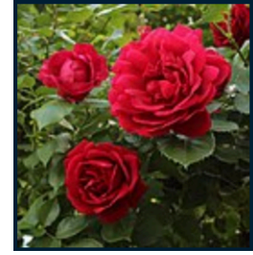
To create a default parameter file, use the -print_params command line option:

```
RadxPrint -print_params > RadxPrint.params
```

To run the application, the user invokes the -params command line option:

```
RadxPrint -params RadxPrint.params -f </path/to/CfRadial_filename>
```

RadxPrint



- To print data header from any of the files supported by Radx.

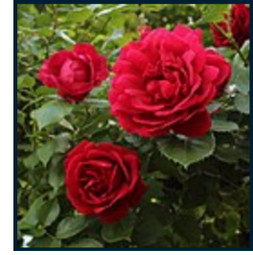
- To find out if a file is supported by LROSE, and to look at the metadata.

To print the reflectivity field for a file and store in a text file using docker, type the following command into a terminal:

1
```
lrose -- RadxPrint -field REF -f /path/to/data/filename > REF.txt
```

```
===== NOTE: Field is from first ray =====
============== RadxField ==============
  name: REF
  longName: radar_reflectivity
  standardName: equivalent_reflectivity_factor
  units: dBZ
  nRays: 1
  nPoints: 1832
  nBytes: 1832
  dataType: si08
  byteWidth: 1
```

Running RadxPrint



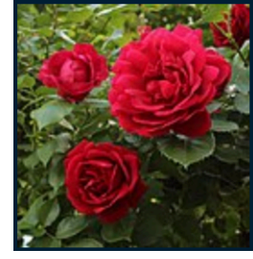
- To print data header from any of the files supported by Radx.

- To find out if a file is supported by LROSE, and to look at the metadata.

To print the data for a file in a generic form, type the following command into a terminal:

2
```
RadxPrint -f /path/to/data/filename
```

Running RadxPrint



- To print data header from any of the files supported by Radx.

- To find out if a file is supported by LROSE, and to look at the metadata.

To print the data for a file in a generic form using python notebook, type the following command:

3

jupyter Tutorial in Python Last Checkpoint: 03/27/2018 (unsaved changes)

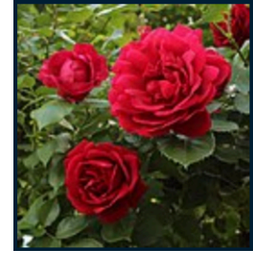
File Edit View Insert Cell Kernel Widgets Help Trusted Python

Code

```
In [15]: ! /usr/local/bin/RadxPrint -f ./output/20161006/swp.1161006190750.KAMX.6.0.9_SUR_v285

=============== RadxVol ===============
  version:
  title:
  institution:
  references:
  source: ARCHIVE
  history:
  comment: Written by DoradeRadxFile object
  scanName:
  scanId(VCP): 0
--------------- RadxPlatform ---------------
  instrumentName: KAMX
  siteName:
  instrumentType: radar
  platformType: fixed
  primaryAxis: axis_z
```

RadxConvert



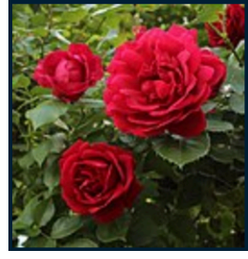
e.g. NEXRAD Level 2 format → CfRadial format

Format	Read Access	Write Access
CfRadial-1	Yes	Yes
CfRadial-2(WMO)(in development)	Yes	Yes
BUFR (in development)	Yes(partial)	No
CFARR	Yes	No
DORADE - NCAR/EOL	Yes	Yes
D23R	Yes	No
DOE ARM netCDF - precedes CfRadial	Yes	No
EEC-Edge	Yes(partial)	No
Foray-1 netCDF - NCAR/EOL	Yes	Yes
GAMIC	Yes	No
Gematronik Rainbow	Yes	No
HSRL (Lidar)	Yes	No
HRD - (Hurricane Research Division)	Yes	No
Leosphere (LIDAR ASCII format)	Yes	No
MDV radial - NCAR/RAL	Yes	Yes
NEXRAD Level 2	Yes	Yes
NOXP	Yes	No
NSSL-MRD	Yes	No
ODIM-H5 (in development)	Yes	Yes
RAPIC - BOM Australia	Yes	No
SIGMET - raw format (Vaisala)	Yes	No
TDWR	Yes	No
TWOLF	Yes	No
UF - Universal Format	Yes	Yes

- New Bufr support in RadxBufr

Supported Lidar and Radar Data Formats:

Running RadxConvert



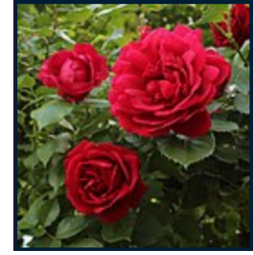
If you don't have specific requirements for the conversion settings, the easiest way to convert data to CfRadial format is to just use the command line '-f' option

```
lrose -- RadxConvert -f /path/to/data/filename
```

or specify the desired output format before -f, such as:

```
lrose -- RadxConvert -dorade -f /path/to/data/filename
```

Running RadxConvert



To create a default parameter file, use the -print_params command line option:

jupyter Tutorial in Python Last Checkpoint: 03/27/2018 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Code

```
In [3]: ! /usr/local/bin/RadxConvert -f -print_params RadxConvert.params
```

Once you have set the parameters, then add the '-params' flag to the command:

jupyter Tutorial in Python Last Checkpoint: 03/27/2018 (autosaved)

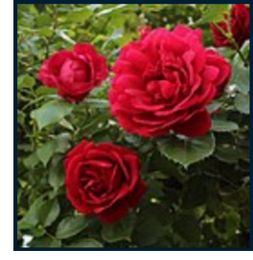
File Edit View Insert Cell Kernel Widgets Help

Code

```
In [18]: ! /usr/local/bin/RadxConvert -f Level2_KAMX_20161006_1906.ar2v -params RadxConvert.params
========================================================================
Program 'RadxConvert'
Run-time 2018/03/27 19:13:54.

Copyright (c) 1992 - 2018
University Corporation for Atmospheric Research (UCAR)
National Center for Atmospheric Research (NCAR)
Boulder, Colorado, USA.
```

Important note for RadxConvert



1. We encourage users to not move the converted data from the date subdirectory directory or rename the files.

2. Compile the CfRadial files as an aggregation of sweep files which makes up a single volume scan.

3. CfRadial netCDF format has well defined metadata standards, so we recommend migrating your workflow to use the CfRadial for the data analysis.

    - Interoperability of CfRadial format with both Py-ART and BALTRAD, and ultimately as WMO standard with CfRadial 2.0

Radx2Grid

1. Coordinate transformations for ground-based radars from a spherical grid on which radar data is collected to a regular grid.

2. The following regular grids are supported:

   - Cartesian grid in (Z,Y,X)

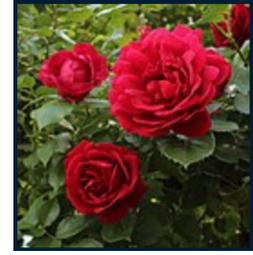
   - PPI grid in (EL, Y, Z) - performs Cartesian transformation on each elevation angle separately

   - Polar grid, regular in (EL, AZ, RANGE)

3. Radx2Grid is beneficial to use for the interpolation with ground-based radars and vertically pointing airborne lidar and radar data.

4. ELDORA and the NOAA P-3 tail radar the current interpolation method is not appropriate and Radx2Grid will not work properly.

Running Radx2Grid



To see all command line options for Radx2Grid

```
lrose -- Radx2Grid -print_params > Radx2Grid.param
```

A few key parameters to look for in the parameter file are listed below:

- start_time, end_time: set the start time and end time for ARCHIVE mode analysis.
- input_dir: input directory for searching for files.
- interp_mode: set the interpolation mode.
- grid_z_geom: specify vertical grid levels.
- grid_xy_geom: similar as grid_z_geom. It specifies the grid parameters in x and y.
- netcdf_style: specify different format of netCDF. If output_format is CFRADIAL, specify the netCDF format.
- ncf_title: title string for netCDF file.
- ncf_institution: institution string for netCDF file.
- ncf_source: source string for netCDF file.

After completing the parameter setting, perform the command:

```
lrose -- Radx2Grid -f /path/to/data/filename
-params Radx2Grid.param
```

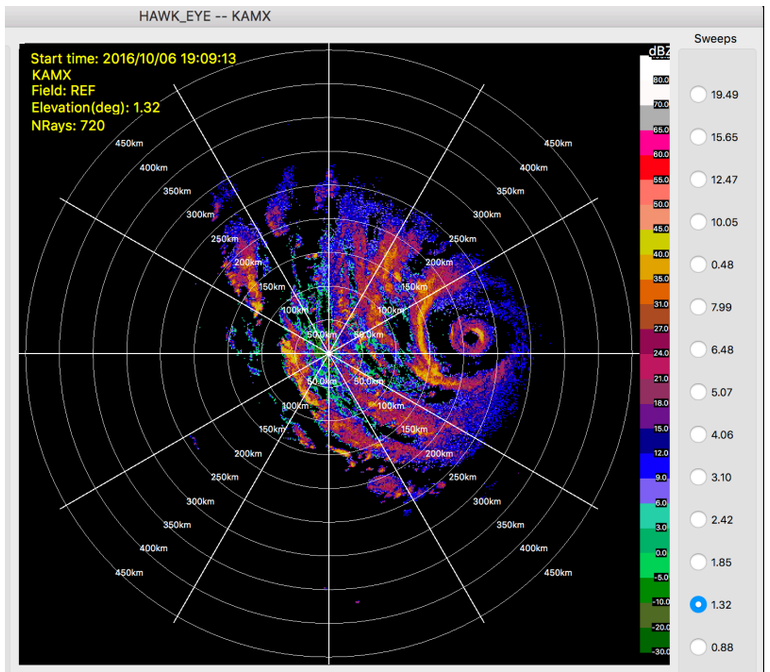
A Basic LROSE-Blaze Workflow

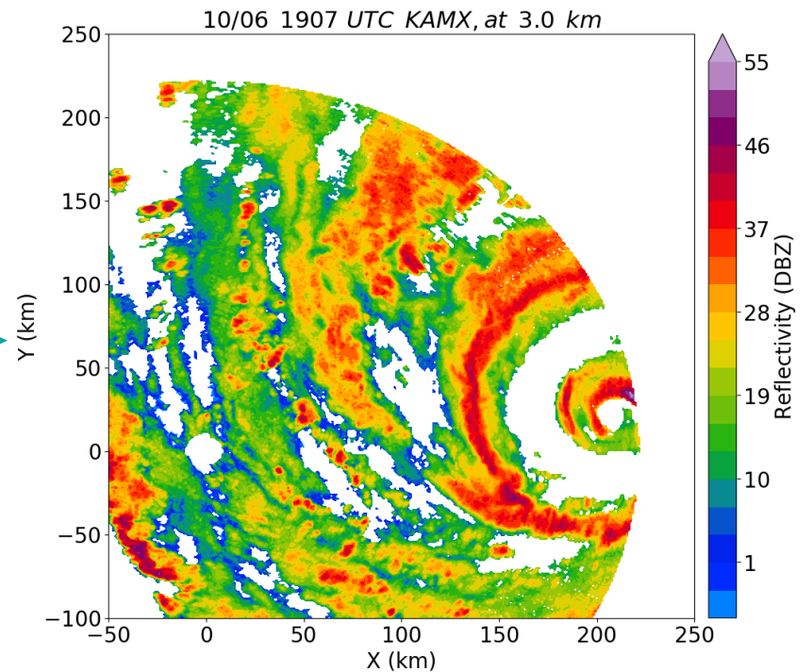
1. Print data header from a NEXRAD file using **RadxPrint**.

2. Convert Level II data to CfRadial format using **RadxConvert**.

3. Display the data in CfRadial format using Hawkeye.

4. Perform coordinate transformations from the polar grid to a Cartesian grid using **Radx2Grid**.





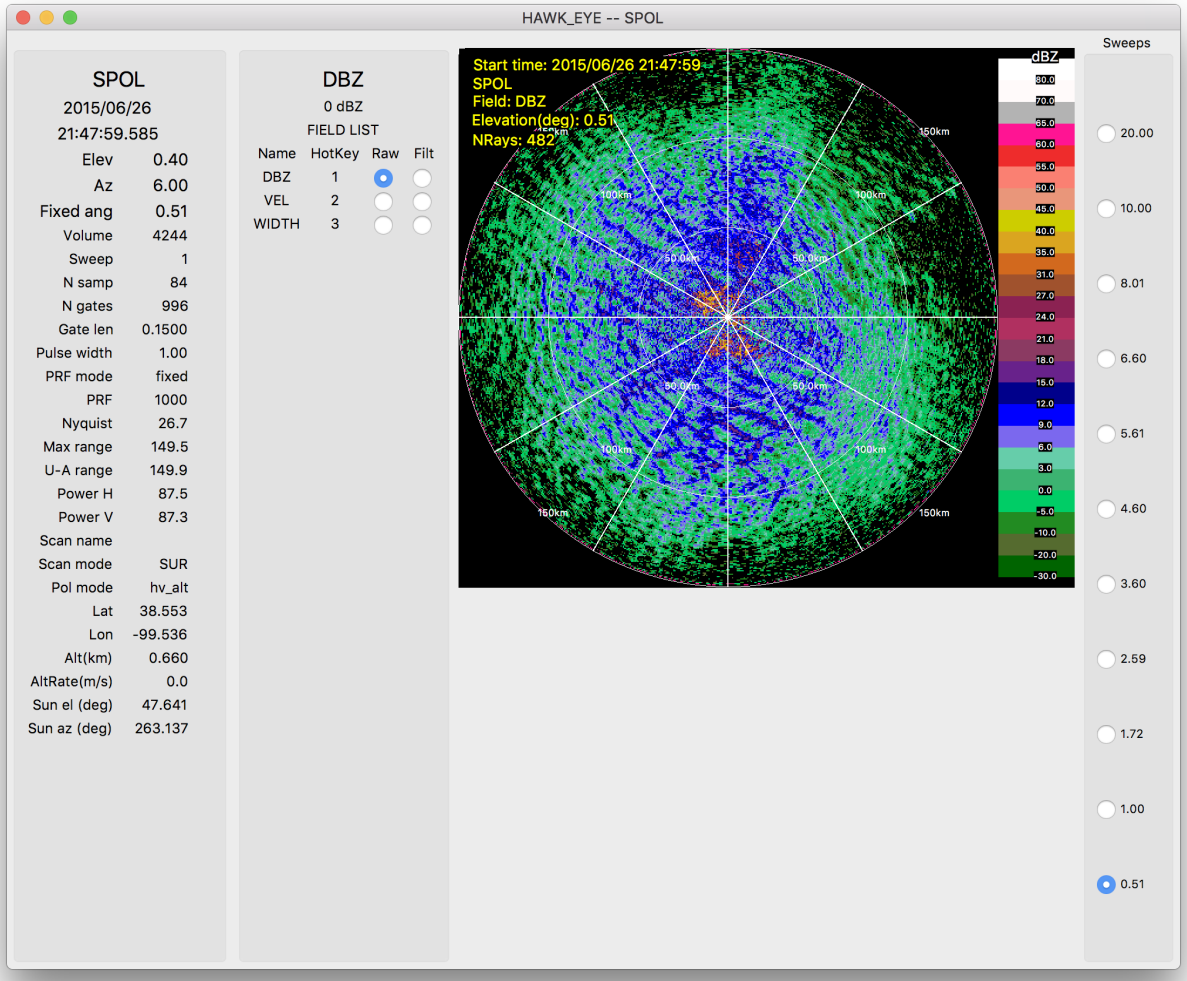
HawkEye

What is HawkEye?

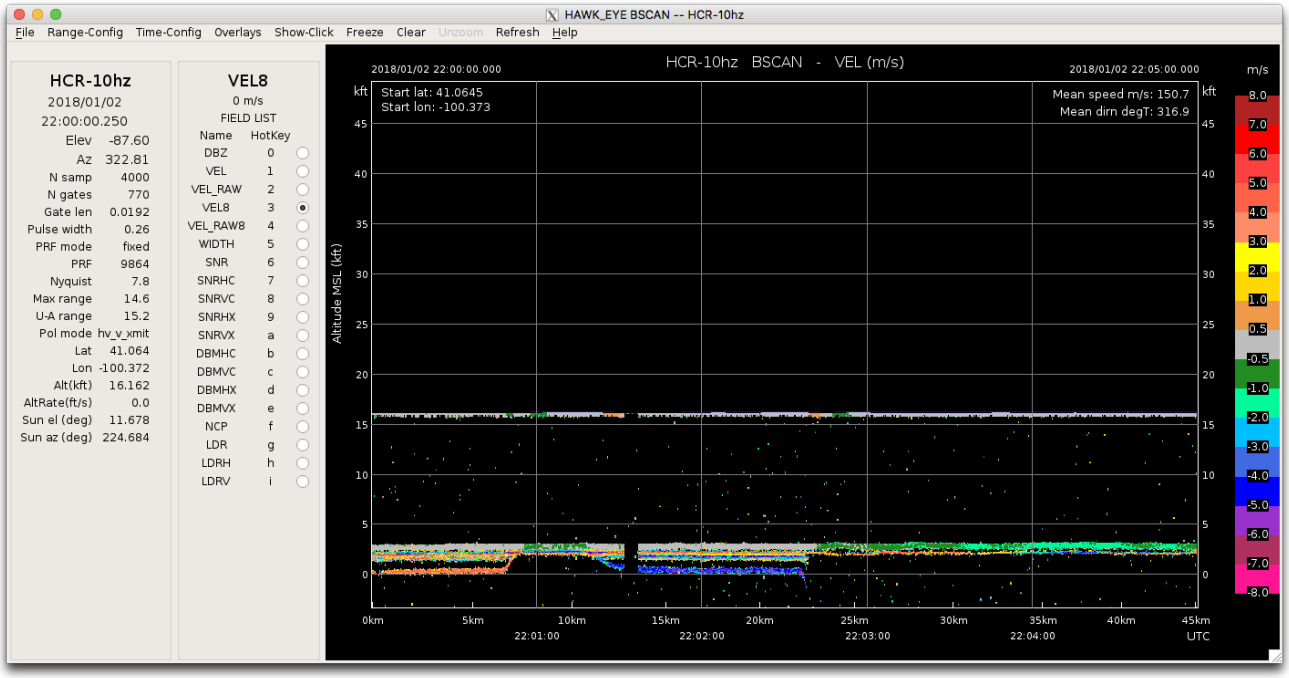
Quote from NSF-LROSE

> HawkEye is a lidar and radar display tool. It can display real-time and archived CfRadial (and Dorade) data either in BSCAN, PPI, or RHI geometry.

Polar (PPI/RHI) Display



BSCAN Display



How to install HawkEye

Linux

From ...	Download Location	Install with ...	Start Command
source	NSF LROSE	python build script	`/install/path/bin/HawkEye`
container (Docker)	NSF LROSE	`docker pull nsflrose/lrose-blaze`	lrose wrapper script (ask Bruno) `lrose -h`
			or `docker run lrose-blaze <various args>`
RPM	(test)lrose-alpha `lrose-blaze-20180629.x86_64.rpm`	`rpm -i lrose-blaze-yyyymmdd.x86_64.rpm`	`/usr/local/lrose/bin/HawkEye`

MacOS

From ...	Download Location	Install with ...	Start Command
source	NSF LROSE	python build script	`/install/path/bin/HawkEye`
container (Docker)	same as for Linux above	same as for Linux above	same as for Linux above
brew		`brew install lrose-blaze.rb`	`/usr/local/bin/HawkEye`
			** menus may not work **
			click away then back
App	(test)lrose-release-test `HawkEye_Blaze.dmg`	download .dmg file	click on App

From ...	Download Location	Install with ...	Start Command
		drag icon to Applications folder	

Windows

as ...
Linux subsystem

How to use HawkEye

Generally, just start HawkEye via the command line or click the App.

```
HawkEye
HawkEye -h
HawkEye -f
test_data/cfradial/kddc/20150626/cfrad.20150626_025610.151_to_20150626_030145.891_KDDC_v270_Surveillance_Sl

HawkEye -p field_project.HawkEye.params
```

Then, you can play with the ...

1. parameter file
2. color scales
3. directory structure

The parameter file

- Many parameters available to customize the display
- Many parameters can also be set on the command line

How to generate one

```
HawkEye --print_params
HawkEye --print_params > field_project.HawkEye.params
```

Where is a default one?

The Top 10 Parameters

These are my favorites, the parameters I check and change to get things running.

1. archive vs. realtime mode

```
begin_in_archive_mode = TRUE; or FALSE;
```

2. parameter/field names

```
fields = {
        label = “DBZ”,
        raw_name = “DBZ”,
        filtered_name = “”,
        units = “dBZ”,
        color_map = “dbz.colors”,
```

```
        shortcut = "1"
};
```

3. start date and time

```
archive_start_time = "1970 01 01 00 00 00";
```

4. some kind of end time

```
archive_stop_time = "1970 01 01 00 00 00";
archive_time_span_secs = 3600;
```

5. color scales

```
color_scale_dir = "../share/color_scales";
```

6. data source ** Note: There is an expected directory structure for the data files

```
archive_data_url = "/data/cfradial/kddc";
```

7. display mode (POLAR or BSCAN)

```
display_mode = POLAR_DISPLAY;
```

8. saving images to file

```
images_output_dir = "/tmp/images/HawkEye";
```

9. image file format (png, jpg, gif)

```
images_file_name_extension = "png";
```

10. debug mode

```
debug = DEBUG_NORM;
```

The Color Scales

The color scales are expected to be in a particular location. However, there are some internal, default color scales:

- default
- rainbow
- eldoraDbz
- spolDbz
- eldoraVel
- spolVel
- spolDiv

You can set the directory for color scales in 2 ways:

- Set to the absolute path
- Set as a path relative to the location of the application binary executable.

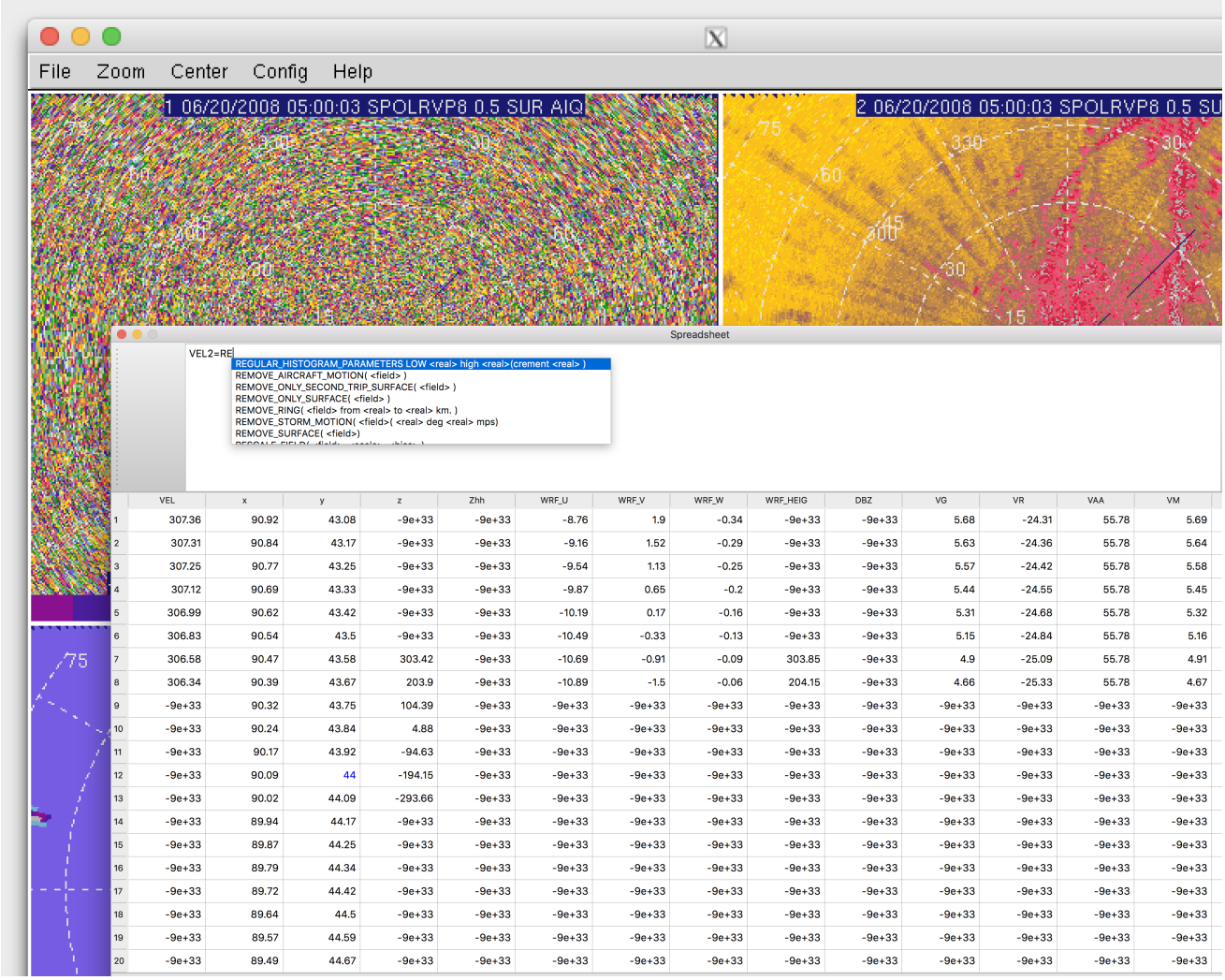
Expected directory structure and data file names

```
+-- field_project
|   +-- YYYYMMDD
|       +-- cfrad.YYYYMMDD_HHMMSS.SSS_to_YYYYMMDD_HHMMSS.SSS_*
```

The Future of HawkEye - advertisement

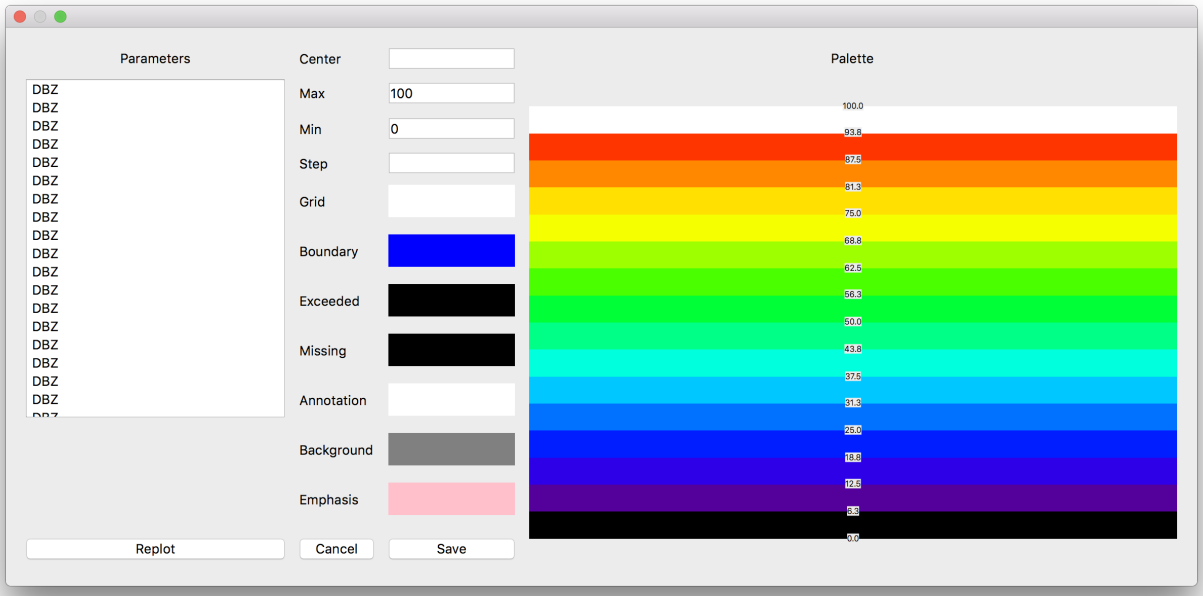
- Open Params File menu option
- Merge SOLOII into HawkEye
- undo editing
- A few screen shots

Examine as Spreadsheet

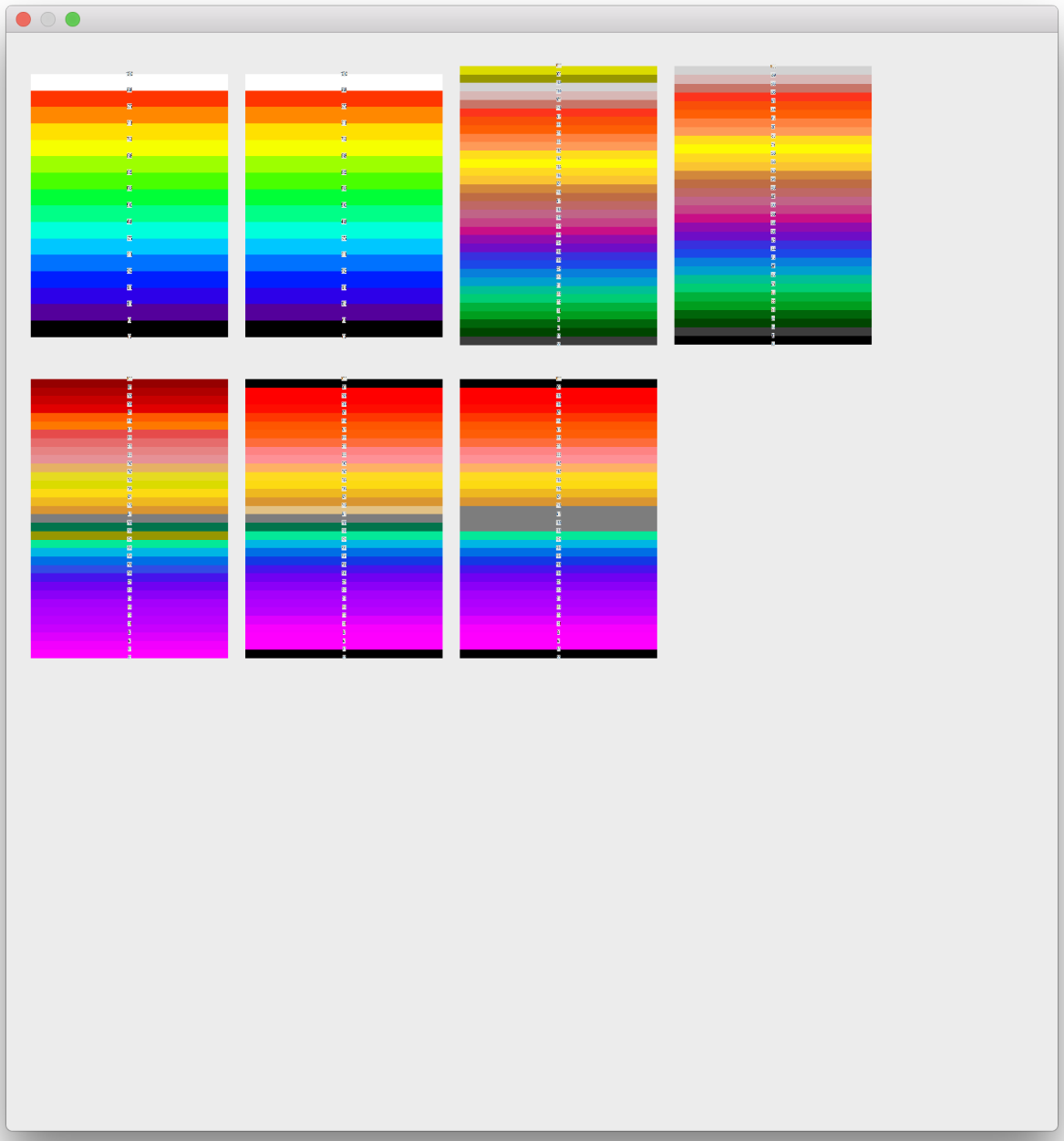


	VEL	x	y	z	Zhh	WRF_U	WRF_V	WRF_W	WRF_HEIG	DBZ	VG	VR	VAA	VM
1	307.36	90.92	43.08	-9e+33	-9e+33	-8.76	1.9	-0.34	-9e+33	-9e+33	5.68	-24.31	55.78	5.69
2	307.31	90.84	43.17	-9e+33	-9e+33	-9.16	1.52	-0.29	-9e+33	-9e+33	5.63	-24.36	55.78	5.64
3	307.25	90.77	43.25	-9e+33	-9e+33	-9.54	1.13	-0.25	-9e+33	-9e+33	5.57	-24.42	55.78	5.58
4	307.12	90.69	43.33	-9e+33	-9e+33	-9.87	0.65	-0.2	-9e+33	-9e+33	5.44	-24.55	55.78	5.45
5	306.99	90.62	43.42	-9e+33	-9e+33	-10.19	0.17	-0.16	-9e+33	-9e+33	5.31	-24.68	55.78	5.32
6	306.83	90.54	43.5	-9e+33	-9e+33	-10.49	-0.33	-0.13	-9e+33	-9e+33	5.15	-24.84	55.78	5.16
7	306.58	90.47	43.58	303.42	-9e+33	-10.69	-0.91	-0.09	303.85	-9e+33	4.9	-25.09	55.78	4.91
8	306.34	90.39	43.67	203.9	-9e+33	-10.89	-1.5	-0.06	204.15	-9e+33	4.66	-25.33	55.78	4.67
9	-9e+33	90.32	43.75	104.39	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33
10	-9e+33	90.24	43.84	4.88	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33
11	-9e+33	90.17	43.92	-94.63	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33
12	-9e+33	90.09	44	-194.15	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33
13	-9e+33	90.02	44.09	-293.66	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33
14	-9e+33	89.94	44.17	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33
15	-9e+33	89.87	44.25	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33
16	-9e+33	89.79	44.34	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33
17	-9e+33	89.72	44.42	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33
18	-9e+33	89.64	44.5	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33
19	-9e+33	89.57	44.59	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33
20	-9e+33	89.49	44.67	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33	-9e+33

Color Palette Editor



Predefined Color Scales



Demo with Data

~/Workshop2019 start_HawkEye.test

data run from time 00:00 to 03:00:00

RADX Dual-Polarization Applications

KDP, PID and Precipitation Rate Applications

The following applications form a group that handle KDP estimation, particle identification (PID) and precipitation rate estimation.

App	KDP	Z & ZDR Attenuation	NCAR PID	Precip Rate
RadxKdp	✓	✓	✗	✗
RadxPid	✓	✓	✓	✗
RadxRate	✓	✓	✓	✓

Use of shared parameter files

Each of these apps has a main parameter file, which controls file handling, operational modes etc.

The main file in turn points to shared parameter files for specific purposes.

RadxKdp reads in a KDP-specific parameter file.

RadxPid reads in the KDP parameter file, and in addition 2 PID-specific files: (a) the PID overview and (b) the interest maps file that specifies the details of the PID fuzzy logic algorithm.

RadxRate reads in all of the above, plus a file specific to the computation of precip rate.

App	App params file	Kdp-specific params	Pid-specific params	Pid Interest Maps	Rate-specific params
RadxKdp	✓	✓	✗	✗	✗
RadxPid	✓	✓	✓	✓	✗
RadxRate	✓	✓	✓	✓	✓

Example images

DualPolProcessing.md presents some example images from this processing chain.

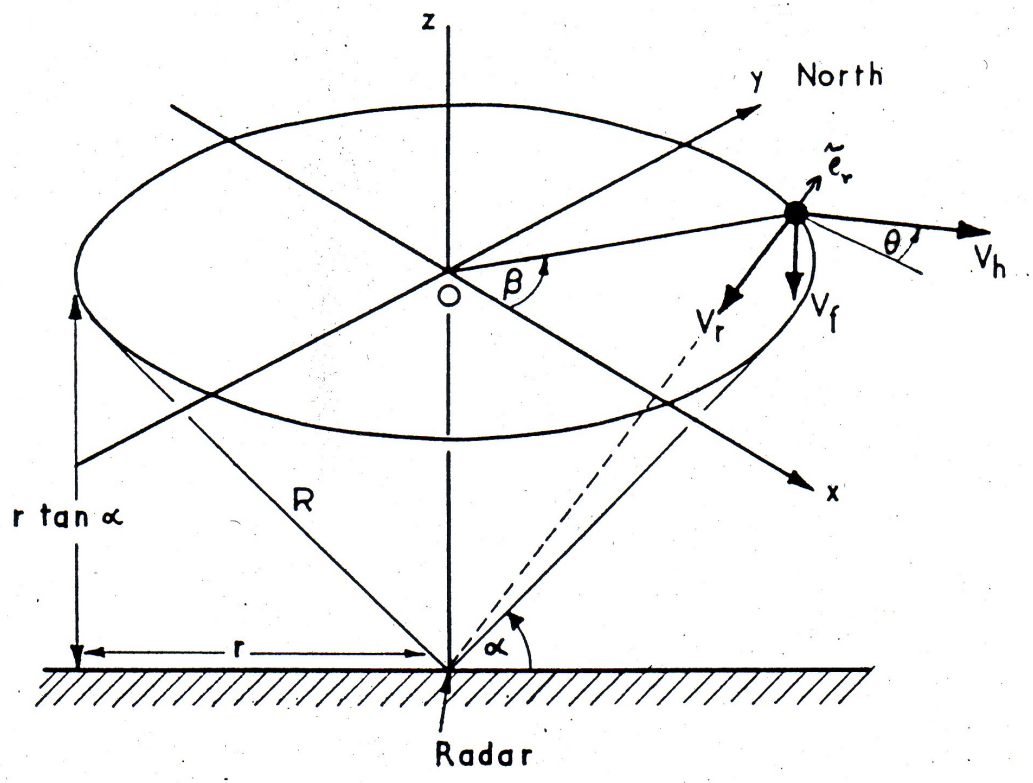
"Blaze" Wind toolsets

RadxEvad, FRACTL, SAMURAI

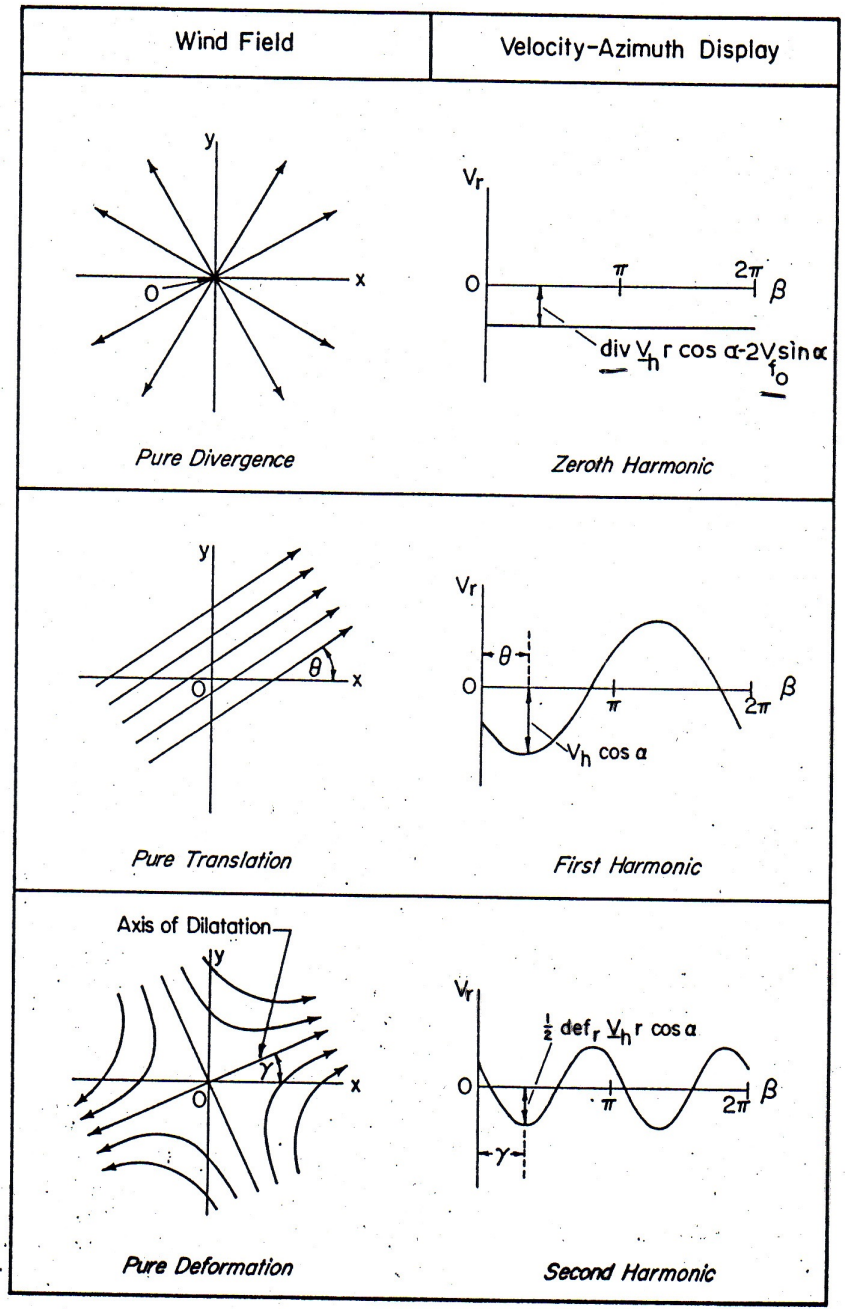
Michael Bell

VELOCITY-AZIMUTH DISPLAY (VAD) PROCESSING OF RADIAL VELOCITY DATA FROM A DOPPLER RADAR

A technique for the measurement of kinematic properties of a wind field in widespread echo coverage using a single Doppler radar



$$V_r(\beta) = -V_x(\beta)\cos\beta\cos\alpha - V_y(\beta)\sin\beta\cos\alpha + V_f(\beta)\sin\alpha$$



Contains information about Divergence and total fall velocity of the particles

Contains information about the horizontal wind speed and direction

Contains information about the flow deformation and its orientation

Extended Velocity Azimuth Display

Goal: To separate the horizontal divergence from the hydrometeor fall velocity in the zeroth harmonic

$$a_0 = -r\cos\alpha\left(\frac{\partial V_x}{\partial x}+\frac{\partial V_y}{\partial y}\right)+2V_f\sin\alpha \qquad D=\left(\frac{\partial V_x}{\partial x}+\frac{\partial V_y}{\partial y}\right)$$

Let's rewrite equation:

$$D=-\left(\frac{2\sin\alpha}{r\cos\alpha}\right)V_f+\frac{2a_0}{r\cos\alpha}$$

$$D = mV_f + b$$

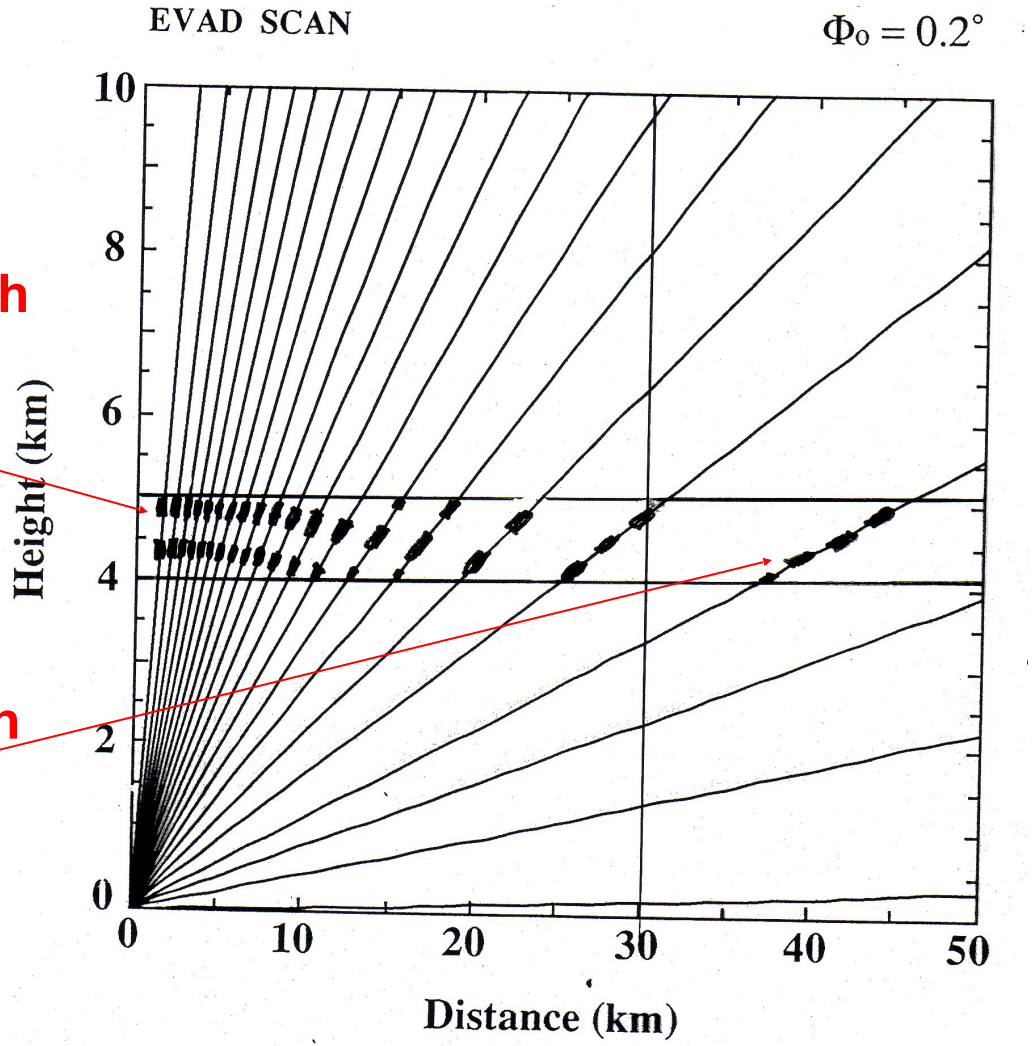
Srivastava et al. (1986): Doppler radar study of the trailing anvil region associated with a squall line. *J. Atmos. Sci.,* **43,** 356-377.

Matejka and Srivastava (1991): An improved version of the Extended Velocity-Azimuth Display analysis of single-Doppler radar data. *J. Atmos. Oceanic Technol.*, **8,** 453-466.

EVAD analysis is done in layers. Each ring gives one data point that is used in a least squares regression to obtain D and V_f within the layer

At high elevation angles, V_f contributes most to zeroth harmonic

At low elevation angles, D contributes most to zeroth harmonic



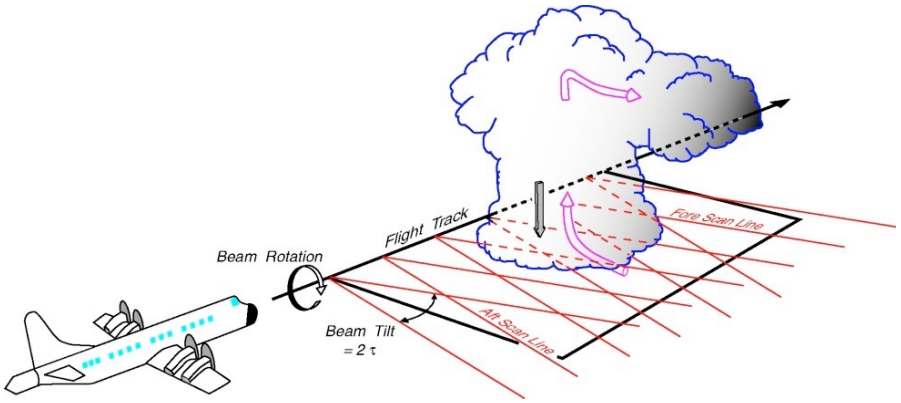
Elevation angles: 0.2, 2.4, 4.3, 6.2, 9.1, 11.9, 14.9, 17.8, 20.8, 24.9, 27.1
(21 angles) 30.3, 33.7, 37.1, 40.8, 44.7, 48.8, 53.3, 58.3, 64.3, 71.9

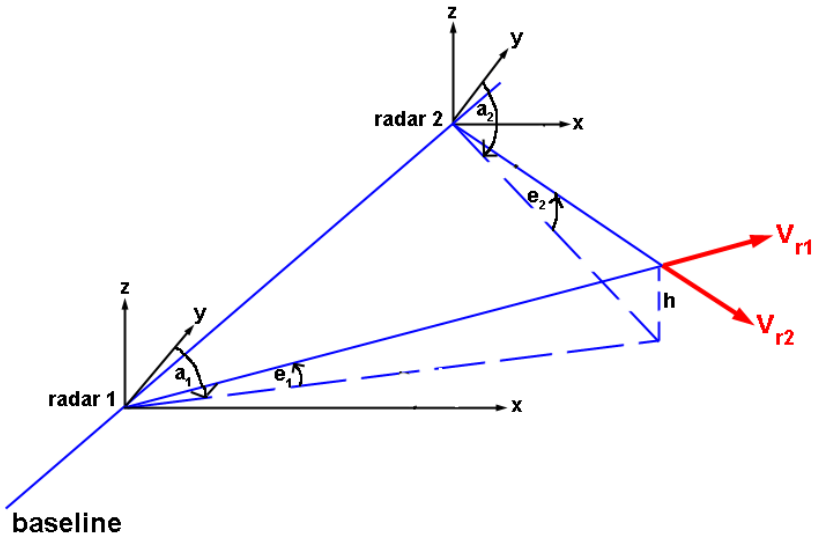
Multi-Doppler Synthesis

$$V_r = u\sin(a)\cos(e) + v\cos(a)\cos(e) + (w + w_t)\sin(e)$$

Three (or more) radar solution (Normal equations)

$$\begin{pmatrix} \sum \sin^2 a\cos^2 e & \sum \sin a\cos a\cos^2 e & \sum \sin a\cos e\sin e \\ \sum \sin a\cos a\cos^2 e & \sum \cos^2 a\cos^2 e & \sum \cos a\cos e\sin e \\ \sum \sin a\cos e\sin e & \sum \cos a\cos e\sin e & \sum \sin^2 e \end{pmatrix} \begin{pmatrix} u \\ v \\ w + w_t \end{pmatrix} = \begin{pmatrix} \sum V_r \sin a\cos e \\ \sum V_r \cos a\cos e \\ \sum V_r \sin e \end{pmatrix}$$





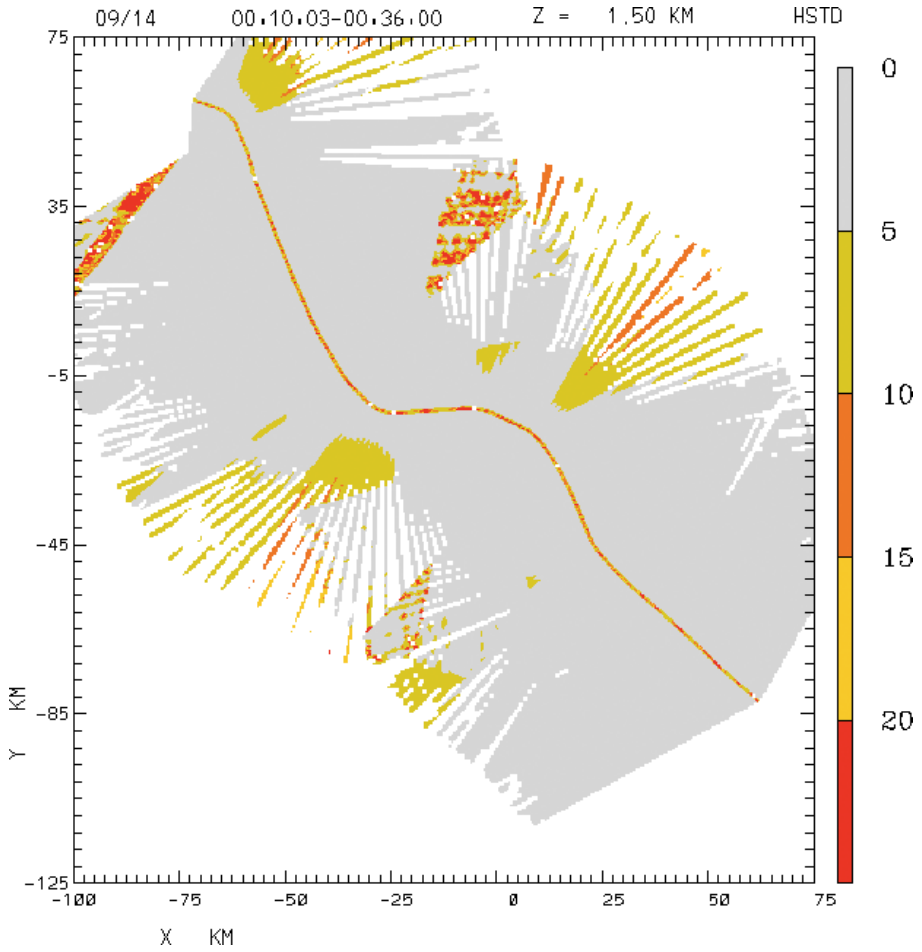
Different analysis techniques have advantages / disadvantages

Traditional "Local" solver (**FRACTL**, CEDRIC)	Variational "Global" solver (**SAMURAI,** HRD, Multi-Dopp)
Fast computation	Computationally demanding
Point-by-point error diagnostics	Global error diagnostics
Doppler data only	Other data sources possible

- **FRACTL (Fast Reorder and CEDRIC Technique in LROSE)**
- **SAMURAI (Spline Analysis at Mesoscale Utilizing Radar and Airborne Instrumentation)**

Radial velocity interpolation and averaging can introduce errors

- A single 'average' radial velocity from each radar is not appropriate for many situations
  - Curved airborne flight tracks
  - Ground-based near the radar
- Solution: Avoid it!
  - Use radar data in its native spherical coordinates



FRACTL

- 'Traditional' multi-Doppler solver written from the ground up in C++
- Quasi-horizontal 2-radar assumption (U,V) with mass-continuity for W, or full 3D solution (U, V, W)
- Nearest neighbor algorithm adds Doppler gates directly to matrix solver at each grid point, no interpolation or averaging of velocity required
- Solve normal equations using Singular Value Decomposition (preferred) or Cramer's method
- Subset of CEDRIC diagnostics implemented, more on the way
- <5 minutes for dual-Doppler (~3M gates)

SAMURAI

- 'Variational' multi-Doppler solver written in C++
- Quasi-horizontal 2-radar assumption (U,V) or full 3-D solution (U, V, W) w/mass continuity for either
- Compares analysis to each radar gate using finite element 3-D function, solves global cost function using conjugate gradient algorithm
- Background 'first-guess' field and non-radar observations can be incorporated into analysis
- CfRadial support
- OpenMP support for shared memory multi-core processing (workstations, single-node of cluster)

LROSE Blaze Wind Summary

- **RadxEvad**
  - Straight-forward implementation of classic technique
- **SAMURAI**
  - Full variational analysis package
  - 8 published papers using technique since 2012
- **FRACTL**
  - Reorder and CEDRIC reborn!
  - Fast traditional solver with integrated interpolation using LROSE infrastructure, no FORTRAN-style input required

Current Status and Next Steps

- Software available now, documentation and tutorial are nearing completion
- Subset of CEDRIC functionality available in FRACTL, more to be implemented
- FRACTL/SAMURAI integration is in experimental stage
- Plan to integrate with Radx2Grid
- Continued optimization to make it faster and bug fixes

“Cyclone” and Future Plans

Michael Bell

Blaze Stable Release	**Cyclone** Development Branch
Bug fix releases will continue in 2019	Will include all bug fixes in Blaze
No new tools, New functionality to existing tools if it doesn't affect reproducibility	New tools and functionality added in each release, may break existing workflows
Citeable by DOI, reproducible DOI 10.5281/zenodo.2532758	Citeable by DOI, but in development and reproducibility not guaranteed with each release

Year	Stable Release		Development Release	Sunset Release
2019		Blaze	Cyclone	N/A
2020		Cyclone	Elle	Blaze
2021		Elle	Jade	Cyclone
2022		Jade	Topaz	Elle

Cyclone Plans

- Radar Quality Control and LIDAR support will be two big focus areas
- Add more CEDRIC functionality to FRACTL, integrate FRACTL/SAMURAI, more HawkEye functionality
- New tools under consideration:
  - RadxFilter - basic filtering and thresholding procedures
  - RadxPersistentClutter - identifies residual clutter
  - RadxTimeMedian - time-based median
  - RadxClutMon - monitoring Z and ZDR stability using clutter
  - RadxSunMon - monitoring sun spikes
  - RadxQC - Automatic QC using PID, sea-clutter, and interference detection
  - RadxModelQC - automatic quality control for data assimilation
  - RadxSelfCons - check for self consistency of CfRadial file
  - RadxDiff - compare two CfRadial files
  - RadxZdrBias - Calculate ZDR bias in ice, light-rain, or Bragg scatter
  - RadxDealias - Velocity dealiasing
  - VORTRAC - single-Doppler retrieval using GBVTD and GVTD technique
  - SAMURAI-TR - variational thermodynamic retrieval
  - Refract - refractivity retrieval

Cyclone Plans

- Aim for 4 releases a year (spring, summer, fall, winter)
- Continue to improve documentation and tutorials
- Develop standardized ‘test suites’ to ensure reproducibility and accuracy
- API development
- Website updates: Better registration form, better message board for community discussion (Discourse?)
- NCAR tutorial/workshop (summer) and AGU workshop (December)