Re-entry Analysis from Serendipitous Radar data (RASR)

Technical Presentation

Yash Sarda - 10/2020

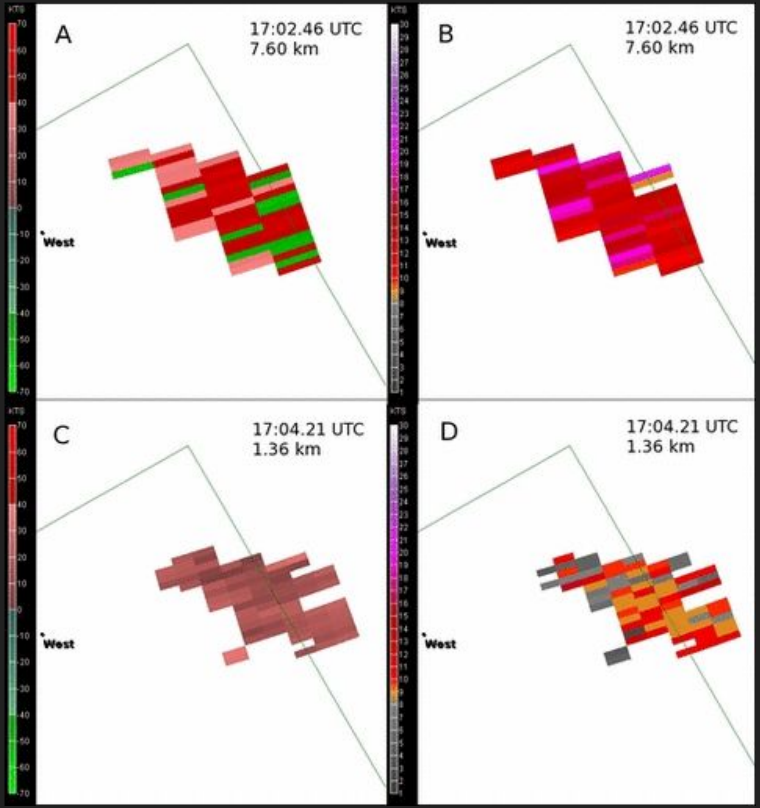
Table of Contents

Glossary

- TACC - Texas Advanced Computing Center, home to Stampede2 supercomputer and Corral file storage server
- NEXRAD - Next Generation Weather Radar
- NOAA - National Oceanic and Atmospheric Association
- CNN - Convolutional Neural Network
- ASTRIA - Advanced Science and Technology Research in Astronautics, an astrodynamics research program led by Dr. Moriba Jah
- ARES - Astromaterials Research and Exploration Science

Background

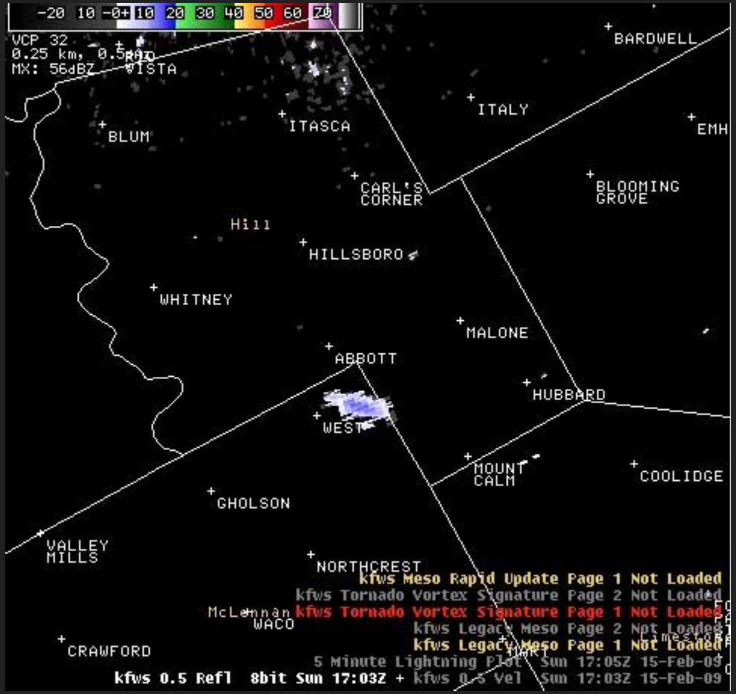
- Re-entering objects become lost quite frequently
- Most US Govt. radar (Military) loses objects somewhere between 200 km and ground
- NEXRAD weather data can be co-opted to catch meteors, as shown by Fries [1]
- NASA ARES keeps a manually updated database of visual sightings with corresponding velocity signature



Radar signature examples at two sweeps in West, TX, from Fries [1]

Purpose

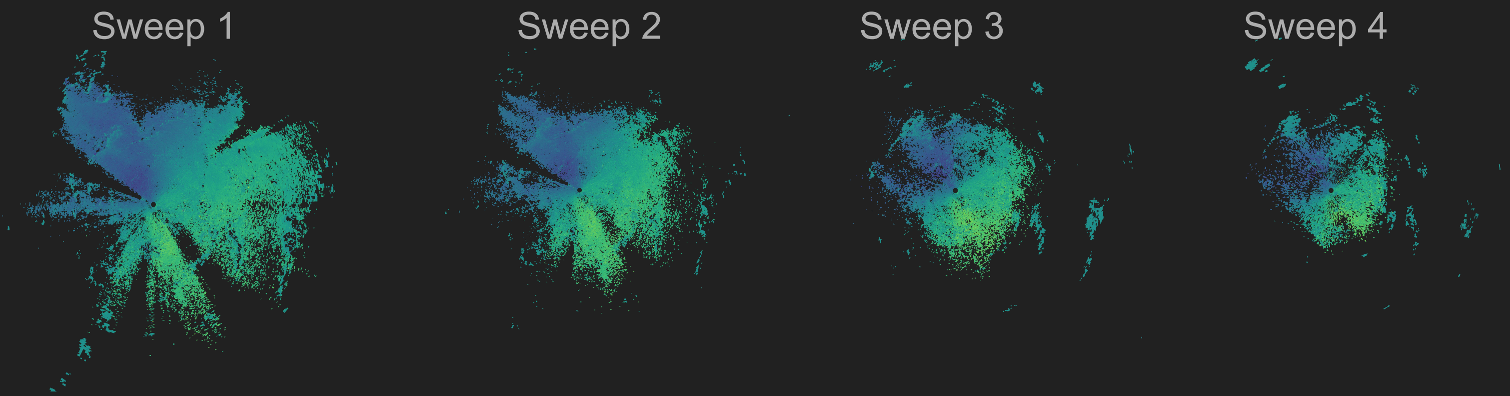
1. Automate the process of detecting and locating these falls
2. Determine if the object is true meteor or orbital debris
3. Kinematically back-calculate trajectory to determine correlation with known orbital bodies



Larger example of same detection, from Fries [1]

Radar Setup


Measured
Distance
Radar
Sweep Angle
Actual Distance


Sweep 1
Sweep 2
Sweep 3
Sweep 4

First Attempt - Automation

1. Developed Python NEXRAD website parser with Graduate student
2. Data files unpacked with Py-ART, NOAA's open source Python package
3. OpenCV package used to process sweeps with colormaps
4. Computations parallelized using Pool

```python
def getMaps():
    #Set velocity colormap
    cmaplist = np.array([[255,0,0],#-70
                         [255,0,0],#-60
                         [210,0,0],#-50
                         [180,0,0],#-40
                         [150,0,0],#-30
                         [115,0,0],#-20
                         [70,40,40],#-10
                         [70,70,70],#0
                         [40,70,40],#10
                         [0,115,0],#20
                         [0,150,0],#30
                         [0,180,0],#40
                         [0,210,0],#50
                         [0,255,0],#60
                         [0,255,0]])/255#70
    cmaplist = np.flip(cmaplist,axis=0)
    velMap = color.LinearSegmentedColormap.from_list('velMap',cmaplist,N=256)

    #Set spectrum width colormap
    cmaplist = np.array([[20,20,20],#0
                         [40,40,40],#6
                         [40,150,40],#12 40 150
                         [150,40,40],#18 150 40
                         [150,70,0],#24
                         [255,255,0]])/255#30
    spwMap = color.LinearSegmentedColormap.from_list('spwMap',cmaplist,N=256)
    return velMap, spwMap

def getData(filename, velMap, spwMap, writeImgs):
    cutoff = 100
    edgeFilter = 8#12
    areaFraction = 1*(10**-4)
    circRatio = 0.3
    fillFilt = 30
    colorIntensity = cutoff
    date=filename
```

OpenCV colormaps from our code

First Attempt - Issues

- Color maps and tuned filters were specific to one example fall
- Hard to generalize to all examples
- **Solution**: More flexible algorithm for image processing

- Memory issues with Matplotlib, TACC resources inefficiently allocated
- **Solution**: Bypass Matplotlib or fix memory issue

Current Solution

- Decided to move to convolutional neural networks, chose Tensorflow Keras
    - Computationally efficient and complex-filtering capable
- Fixed Matplotlib issues with figure management
- Inherited much of the I/O functionality from first iteration

```
 This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (on
eDNN)to use the following CPU instructions in performance-critical operations:
AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate comp
iler flags.
2020-10-07 15:48:23.822981: I tensorflow/core/platform/profile_utils/cpu_utils.c
c:104] CPU Frequency: 1999965000 Hz
2020-10-07 15:48:23.823418: I tensorflow/compiler/xla/service/service.cc:168] XL
A service 0x563a8abdc4e0 initialized for platform Host (this does not guarantee
that XLA will be used). Devices:
2020-10-07 15:48:23.823440: I tensorflow/compiler/xla/service/service.cc:176]
StreamExecutor device (0): Host, Default Version
Checking KCBX at 07/18/2020 23:53:05
```

Current Solution - CNN

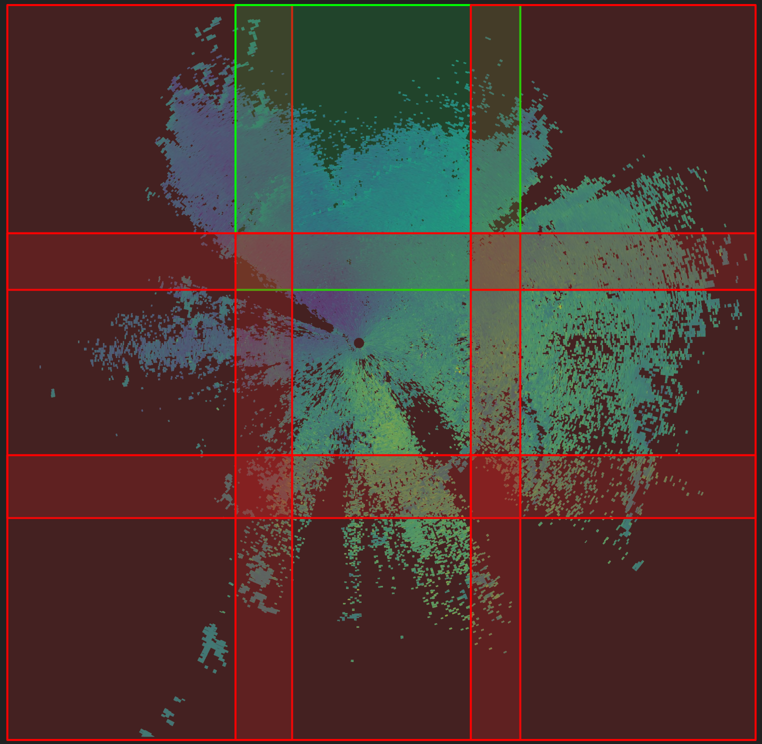
- 8 layer CNN
  - 3 Convolution
  - 2 Pooling
  - 1 Flattening
  - 2 Dense
- Designed with default structure
- Next steps:
  - Sensitivity analysis
  - Optimization

```
################################################################################

#start_time = time.time()
thresh = 0.98
nd = 10
dim = 2500
h = int(dim/nd)
cpath = os.getcwd()
dirname = cpath + '/data/'

model = models.Sequential()
model.add(layers.Conv2D(h, (4, 4), activation='relu', input_shape=(h, h, 4)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(500, (4, 4), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(32, activation='relu'))
model.add(layers.Dense(1, activation = "sigmoid"))
#model.load_weights('rasrmodl')
```

CNN model setup

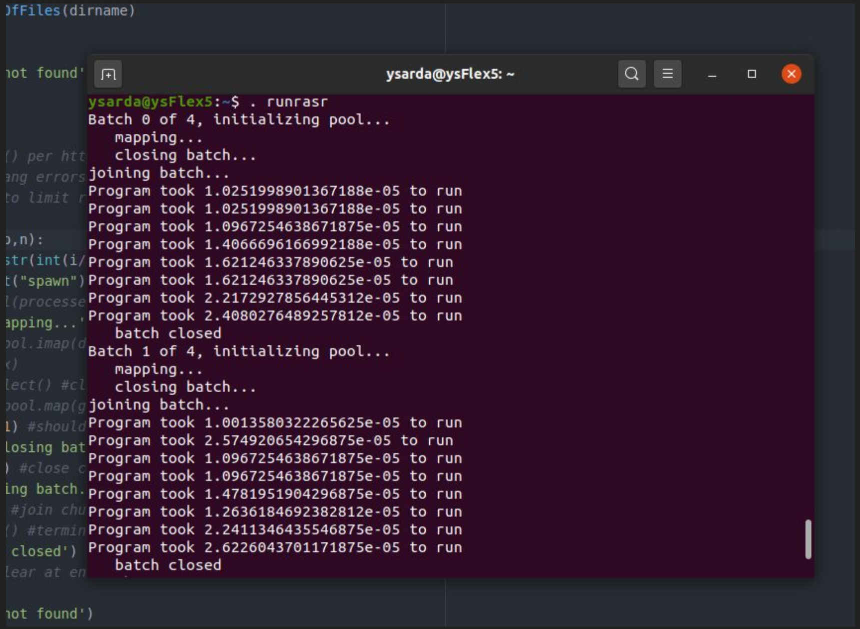
Testing

- 1 sweep is 2500x2500x4 RGBY image
- Batch testing locally: Slow
  - Laptop is 16 GB RAM, 8 core
  - 800% core usage for serial use
- Developed algorithm to split images into overlapping pieces, retaining edge information
- Based on CPU available, image can split into n^2 pieces
- Unfortunately, has to be re-trained for each case



Testing

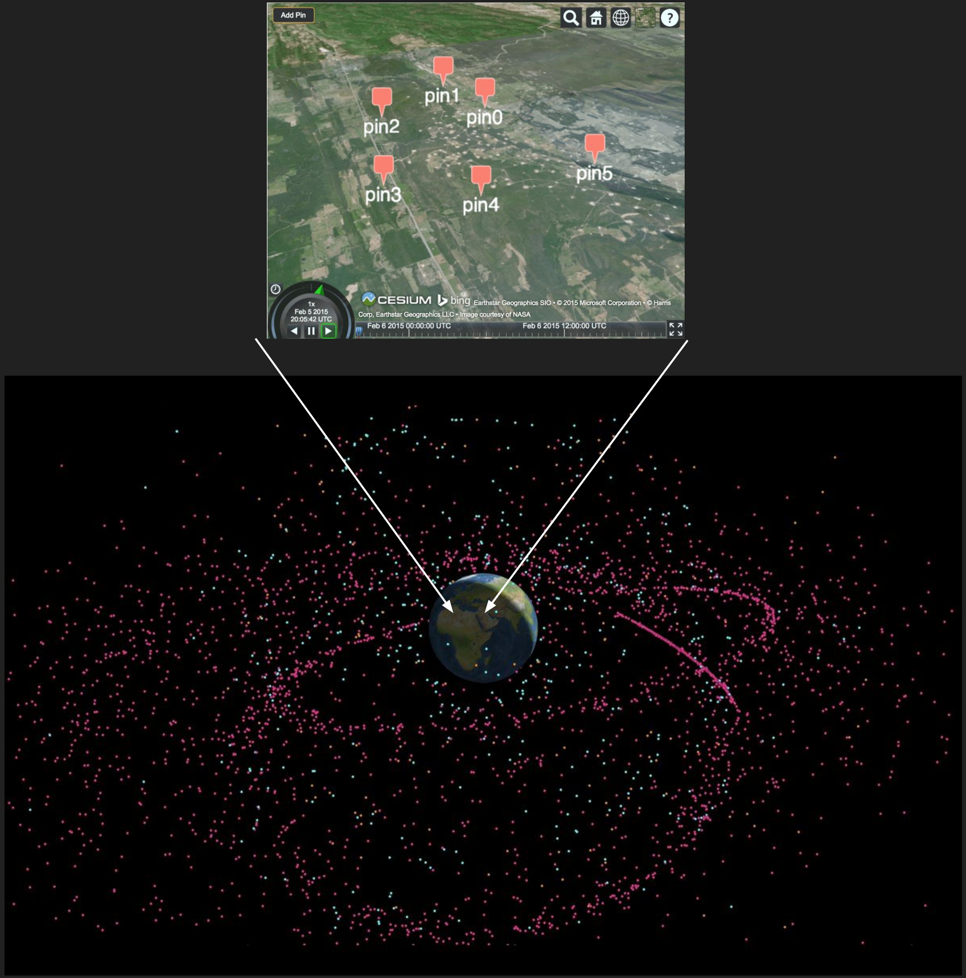
- CNN is run through sigmoid function to normalize and discretize output
- Sifts site by site, sweep by sweep
- Training is simple, image matched with either 1 or 0 from ARES database, and vectorized
- Parallel training on TACC



Batch testing screenshot

Output

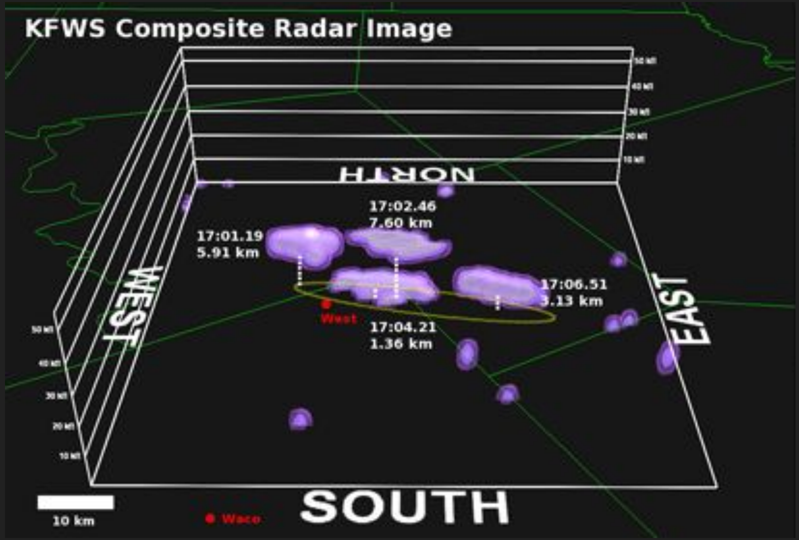
- After detection, relevant data is packaged into a json file
- To be displayed on [ASTRIAGraph]()
- All code, data, and algorithms are publicly available



Next Evolution

- Moving from 2D sweeps to 3D datacubes, for fidelity and use in kinematics
- This means input: 3D to 4D

- Currently, CNN outputs binary
- OD software being written to integrate into CNN
- Will require more extensive labeling, but allow for centroids



Example concept datacube from Fries [1]

References

Thanks to Dr. Moriba Jah, head of ASTRIA and PI of this project, as well as Ben Miller and Robby Keh.

Shoutout to the Tensorflow tutorials

[1] FRIES, M. and FRIES, J. (2010), Doppler weather radar as a meteorite recovery tool. Meteoritics & Planetary Science, 45: 1476-1487. doi:10.1111/j.1945-5100.2010.01115.x