National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

Surface Water and Ocean Topography (SWOT) Mission

SWOT Hydrology Simulator Tutorial

Brent Williams, Ernesto Rodriguez

2016 June 15

This document has been reviewed and determined not to contain export controlled technical data.
Not for Public Release or Redistribution.

What It Does and Where to Get It

- Two primary packages

- SWOT Hydrology Simulator (JPL)
  - Generates radar interferograms, spatially smooths (multilooking), detects water (pixel-wise detection), and geolocates to a pixel cloud (one layer of smoothing)
  - **Contact Dani Fernandez** (dani@jpl.nasa.gov) to obtain binary executables

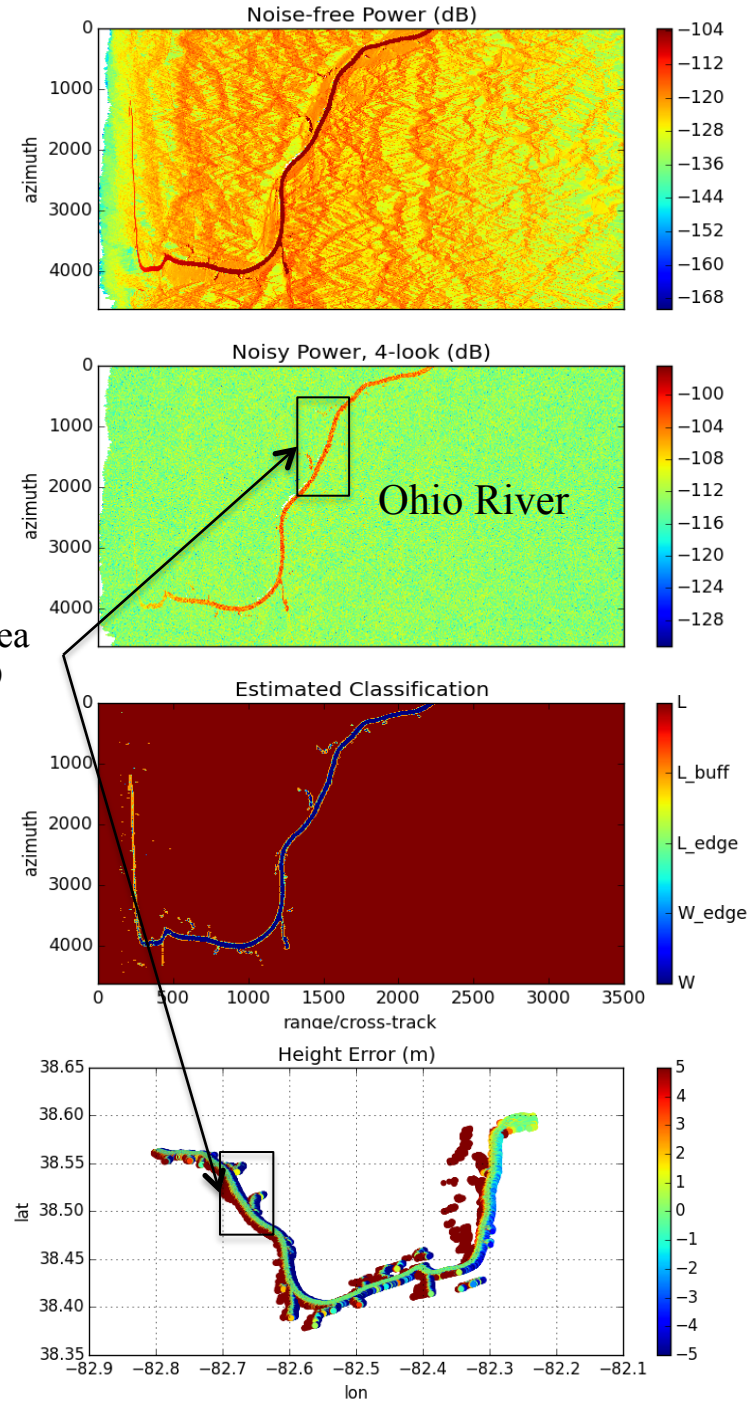
- RiverObs (JPL)
  - Takes in the pixel cloud output from SWOT Hydrology Simulator makes estimates of key river parameters over sub-reaches (width, height , slope)
  - Source code soon to be approved for unlimited release, **contact Ernesto Rodriguez** (Ernesto.Rodriguez@jpl.nasa.gov) to obtain

SWOT Hydrology Simulator (JPL)

- Simulates data and processing from radar interferograms to the geolocated pixel clouc

- Distributed target simulator
  - Directly generates interferograms with appropriate statistics for SWOT geometry

- Implements
  - Geophysical processing
    - ♦ Sensor geometry for SWOT orbits
    - ♦ GDEM-elevation map in radar along-track, ground-range coordinates
  - Interferogram generation considering
    - ♦ Presumming
    - ♦ Coherence-time azimuth smearing of water
    - ♦ Noise addition considering coherent gain power
  - Multilooking (simple and adaptive)
  - Water detection
    - ♦ Pixel-wise Bayes detection with clean-up filter
    - ♦ Pixel-wise fractional water estimation
  - Geolocation to pixel cloud (one smoothing layer)



SWOT Hydrology Simulator (JPL)

- Update soon to be distributed with
  - Updated orbits (including fast sampling phase)
  - Algorithm improvements
  - Improvements to simulating geophysical uncertainties (e.g., dark water)
  - Pixel-cloud product prototype with multiple smoothing layers (e.g., rare, med, well-done)

- Distributed to users as Linux binary executables
  - Various modules as separate executables, e.g.,:
    - ♦ makeGDEM – makes swath oriented DEM and water mask
    - ♦ makeInterferogram – generates interferogram
    - ♦ makeClassification – detects water
    - ♦ makeWaterData – geolocates pixels near water to a pixel cloud

- Run on command line by calling module name with RDF file and options
  - >>moduleName RDFfile <options>
  - >>makeInterferogram SWOT_Ifgram.rdf 0 0 1 0

What is Distributed?

- Binary executables
  - Stand-alone Linux binaries (no library installs needed)
  - Called from command-line or script
    - ♦ Process.txt file given with the examples outlines sequence of calls
    - ♦ Users can make a shell or python script to run all modules in sequence
- Auxiliary data
  - Orbit files, antenna patterns, etc
- Documentation
  - PDF quick user guide explaining each module and how to run them
- Examples
  - Ohio river example
    - ♦ Illustrates how to run the simulator for a real river case
  - Fake snake and lake example
    - ♦ Illustrates how to run the simulator to assess performance with a simplified lake or river tiled throughout the swath

Interfaces

- Data files (netCDF, input and output)
  - GDEM DEM – input DEM with water mask
  - GDEM – swath oriented DEM and mask
  - Interferogram (truth, noisy reference)
  - Water mask – water classification output
  - Pixel cloud – geolocated water mask, heights and other variables

- RDF file inputs
  - Text file with name = value pairs
  - Define configuration as to how the simulator is to be run
    - ♦ Input files
    - ♦ Option switches
  - Can be archived with the outputs to keep track of the configuration that generated the outputs

- Some modules have input RDF files as well as command line options

GDEM DEM netCDF file (ncdump)

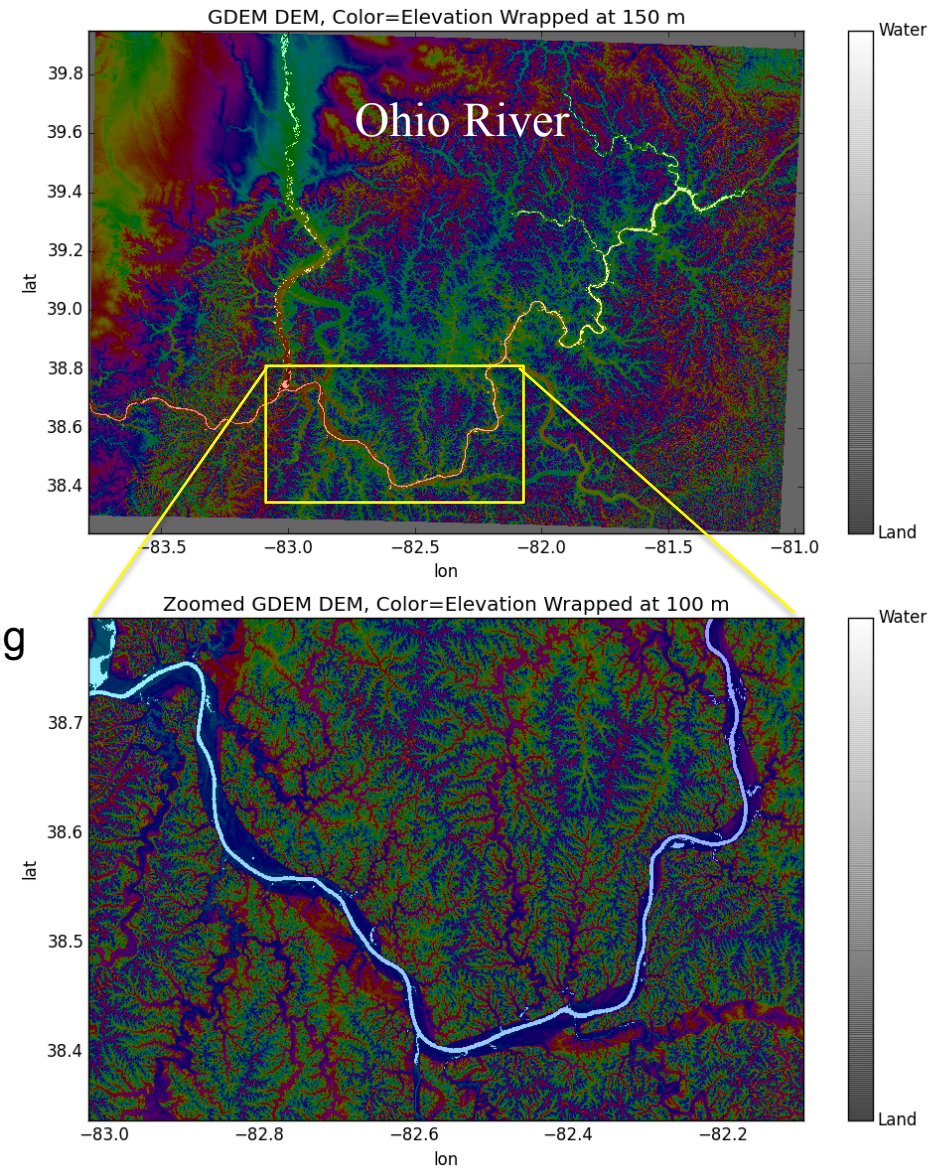
```
netcdf ohio_gdem_dem_slp0.1deg{
dimensions:
    latitude = 5357 ;
    longitude = 8864 ;
variables:
    double longitude(longitude) ;
            longitude:units = "degrees_east" ;
            longitude:_FillValue = -9990000000. ;
    double latitude(latitude) ;
            latitude:units = "degrees_north" ;
            latitude:_FillValue = -9990000000. ;
    byte landtype(latitude, longitude) ;
            landtype:_FillValue = -128b ;
    double elevation(latitude, longitude) ;
            elevation:units = "m" ;
            elevation:_FillValue = -9990000000. ;
// global attributes:
    :water_depth_reference_index = 1 ;
    :water_depth_reference_time = 1543586. ;
    :orbit_start_time = 1457211 ;
    :orbit_end_time = 1457218 ;
}
```

makeGDEM RDF file

```
! makeGDEM parameter RDF file
Sensor file                 = sensor/swot_sensor_ohio_cycle_0001_pass_0413.nc
GDEM DEM file               = gdem_dem/ohio_gdem_dem_slp0.1deg.nc
GDEM basename               = gdem/ohio_gdem_cycle_0001_pass_0413
Interferogram basename      = interferogram/swot_intf_ohio_cycle_0001_pass_0413
Ground spacing              = 10
number of landtypes         = 2 ! minimum number: a water type and a land type
landtype 0 input rdf file   = landtype/land_-5dB.rdf !name is arbitrary
landtype 1 input rdf file   = landtype/water_10dB_25m.rdf !name is arbitrary
```

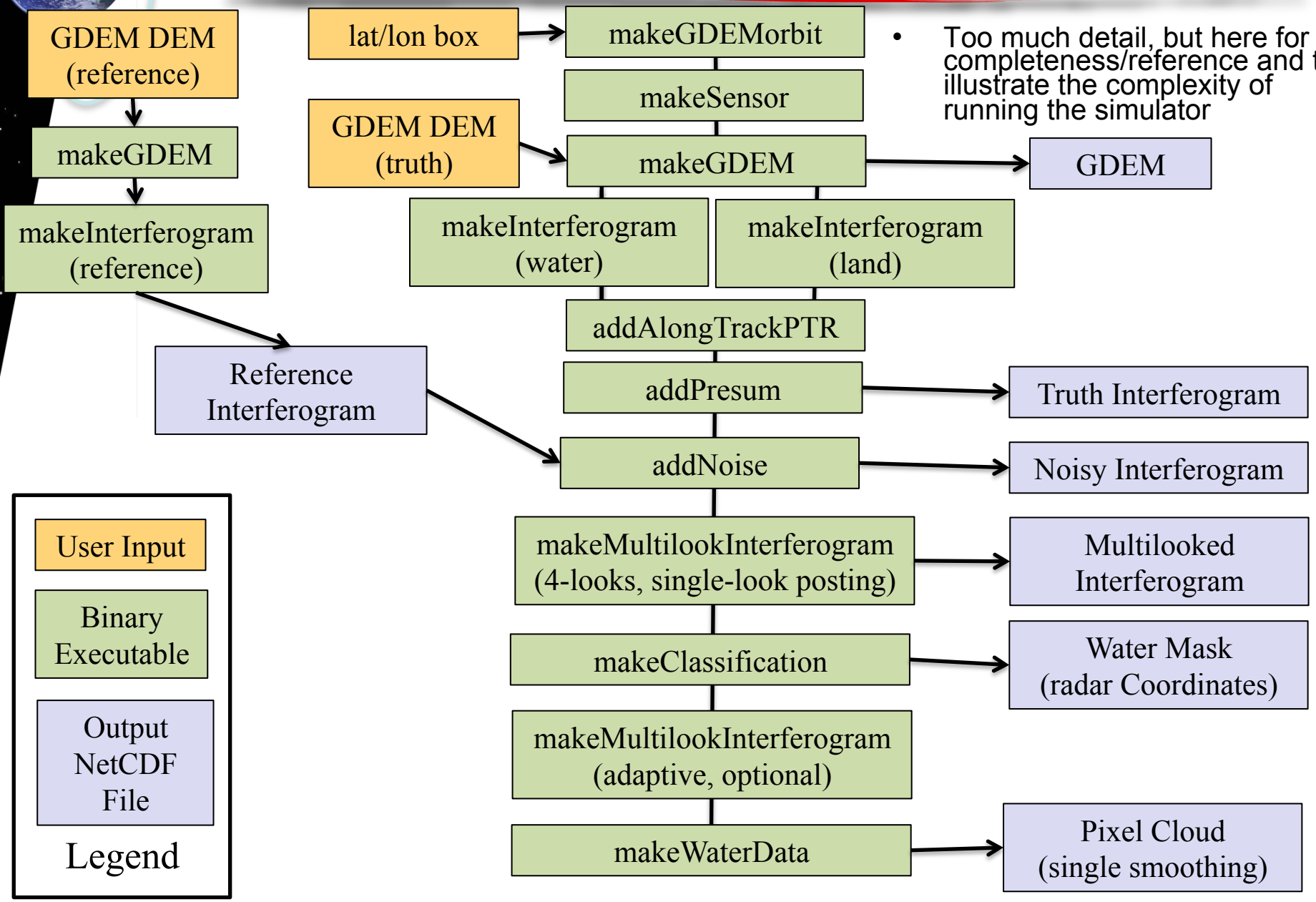
What is Needed to Run It?

- Linux machine
- Define the lat/lon region of interest
  - Change the lat/lon box in the makeGDEMOrbit RDF file
- GDEM DEM files
  - Users will have to generate these for their desired area
    - Ohio example distributed with the code
  - Elevation and land/water type mask
  - Slide 5 shows netCDF structure
  - May desire two different GDEM DEMs to simulate artifacts resulting from using a imperfect DEM or water mask
    - One for truth (e.g., lidar)
    - One for reference (e.g., SRTM)
- Set up all the RDF files to point to the correct places, filenames, etc
- Call binaries following the example process.txt file (see backup slide)

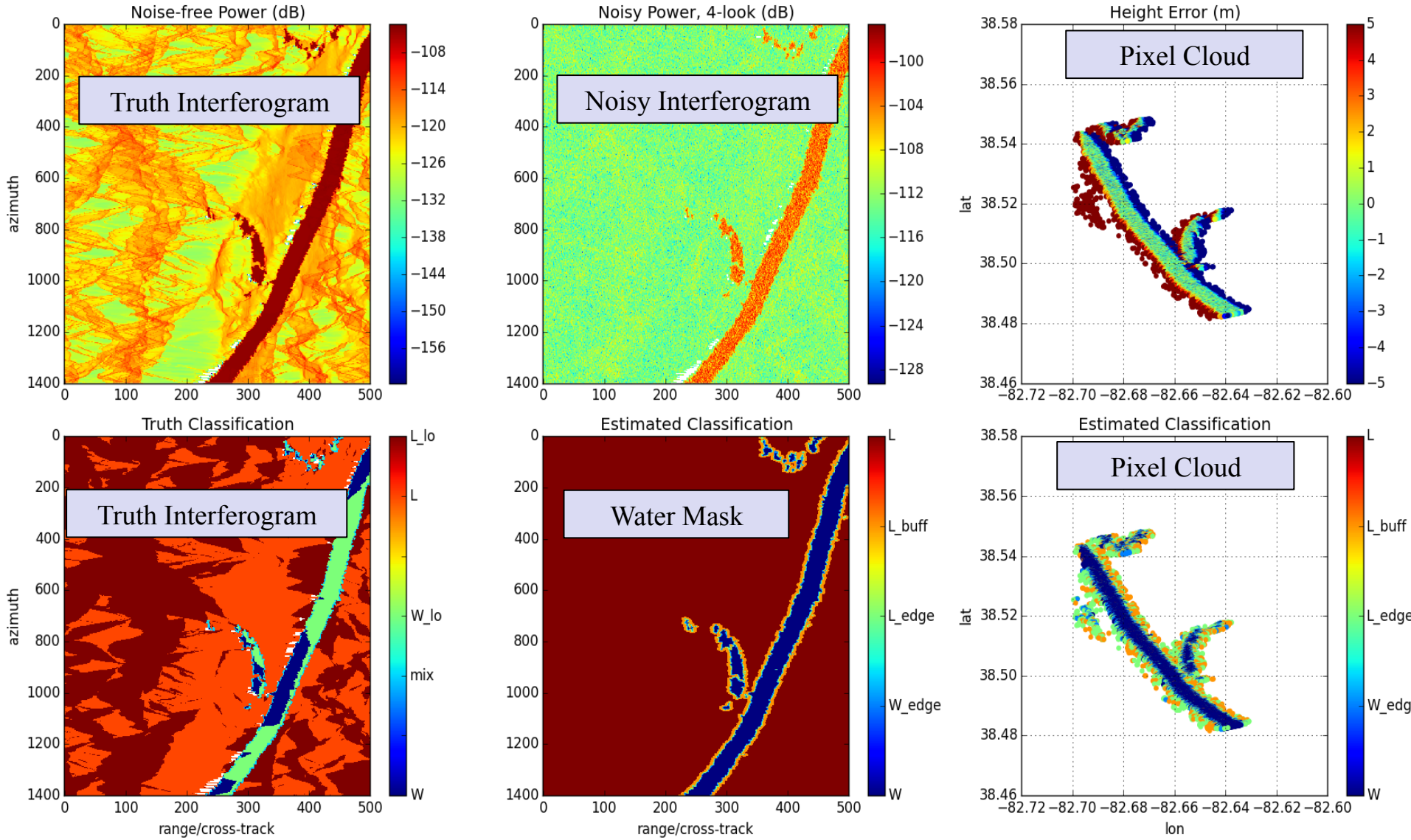


JPL SWOT Hydrology Simulator

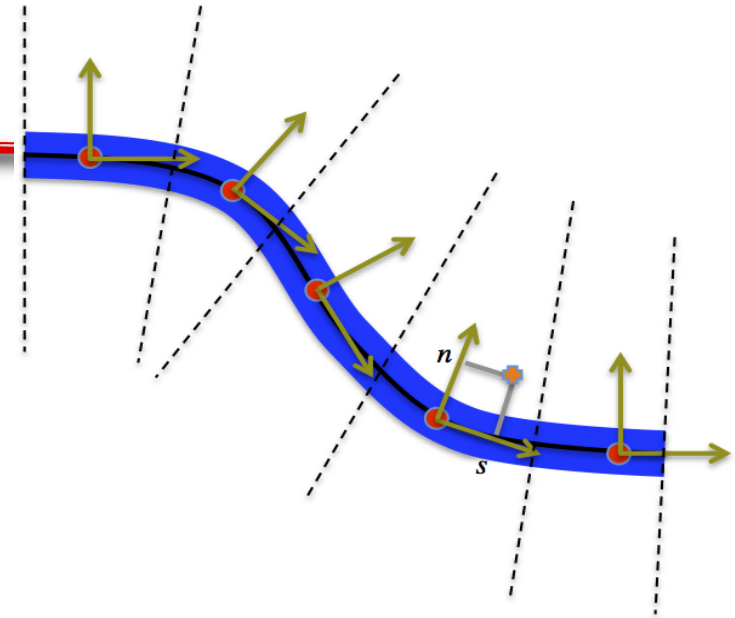
- Too much detail, but here for completeness/reference and to illustrate the complexity of running the simulator



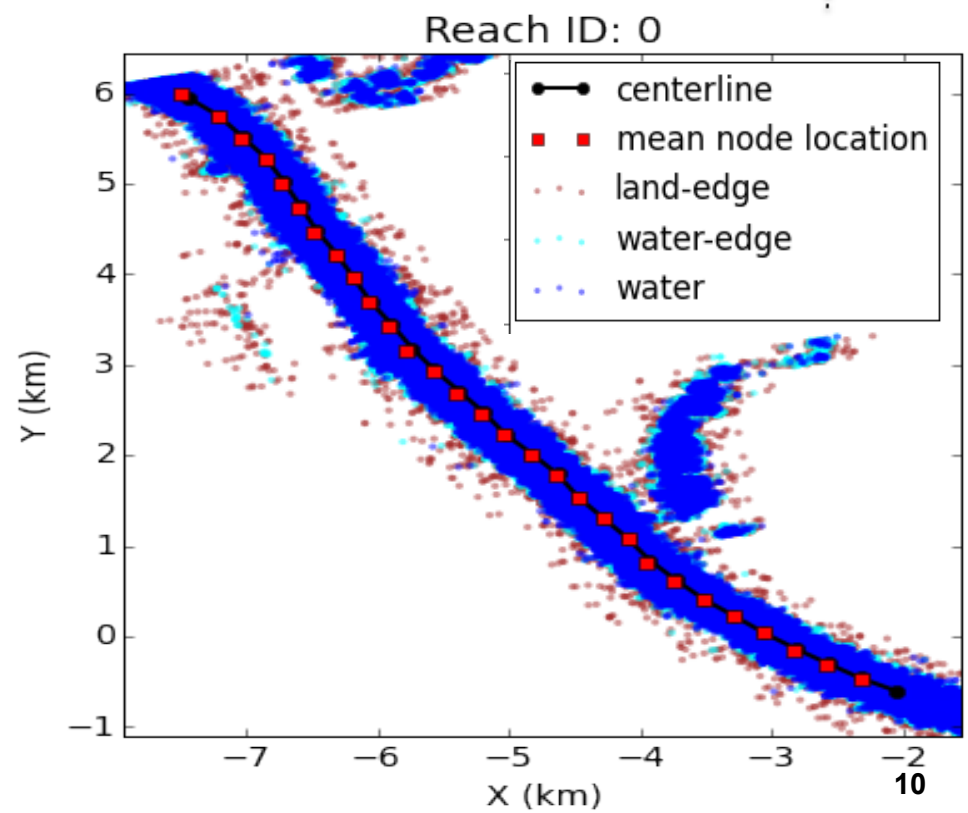
Ohio Example (Zoomed, Mid-swath)



RiverObs



- Takes pixel cloud and estimates quantities over sub-reaches (nodes)
- Uses center-line and reach definitions from prior database
- Center-line and nodes
  - Associates pixels to closest node (exclude those beyond some threshold)
  - Assigns an along-river and cross-river coordinate
  - Computes node-level metrics: width and height
    - Width estimated 3 different ways
      - Based on total pixel area (more robust to geolocation errors)
      - Based on STD around centerline
      - Based on maximum distance (Peak-to-peak)
  - Estimates reach slope using nodes
- Outputs reach-level vector product
  - Reach length, slope, avg. height
  - Center-line location (mean node coordinates)
  - Other useful quantities (viewing geometry, inundated area…)



RiverObs: From Cloud to Shape

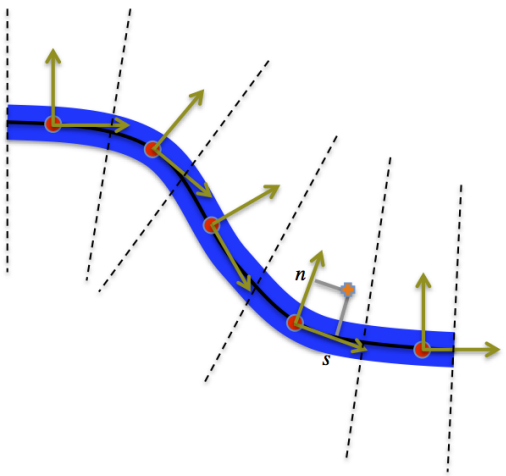
- Most hydrology users will use vector products, rather than the unstructured point cloud data
- The RiverObs software is a proof of concept version of how this might be done
- Basic RiverObs facts:
  - General purpose code, not specific to SWOT data, aside from SWOT data reading modules
  - Code is in Python
    - General purpose toolbox suitable for experimentation and software development
    - Command line program also exists for the non-Python savvy.

- Source code soon to be released open source
  - Mostly through JPL unlimited release process

RiverObs Release

The RiverObs distribution contains:

1. 83-page manual documenting library interfaces, algorithm description, and giving typical use examples.
2. Output shapefiles and KML files illustrating sample data products
   - Right now, supports output into any OGR supported vector format, and HDF5 format (pandas compatible)
3. Analysis of data based on Ohio sample case provided by v2 simulator, including classification layer.



RiverObs Documentation

Release 0.1

Ernesto Rodriguez

January 05, 2015

Summary

- Simulators are a good example interaction between ADT and Science Team

- Bit of a learning curve but
  - Valuable studies can be done
  - Can become familiar with SWOT-like datasets and errors/artifacts

- Science Team is encouraged to use simulators and give feedback
  - Utility of various modules
  - Help define/refine products and algorithms

Backup

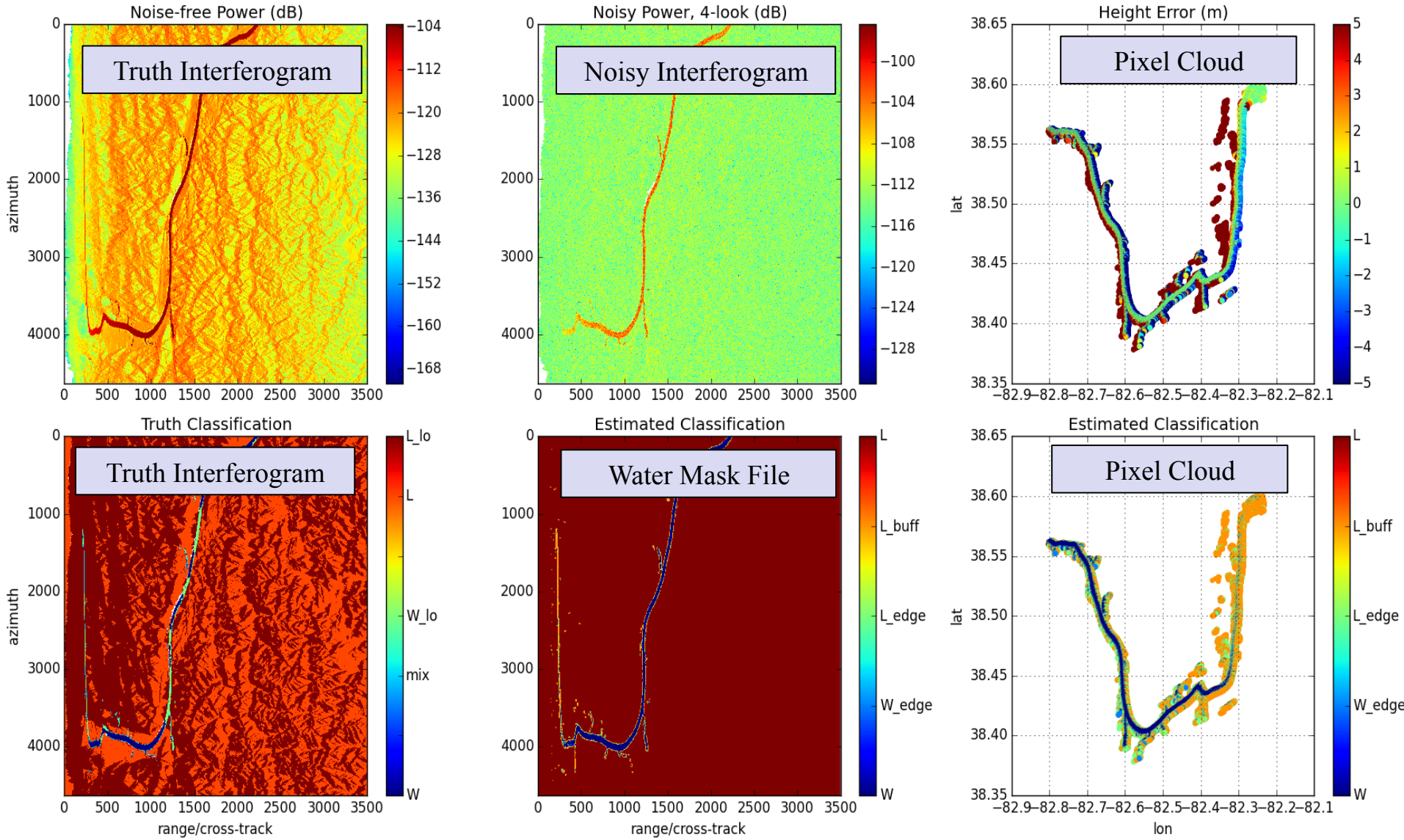
Ohio Example Process.txt

- To run simulator call the following modules from the commandline in the sequence illustrated in the example on the right

- # are comments

```
# Make GDEM orbits
makeGDEMOrbit rdf/SWOT_GDEM_orbit_ohio.rdf

# Make sensor file
makeSensor rdf/SWOT_Ifgram_ohio.rdf

# Make GDEM file for right swath
makeGDEM rdf/SWOT_Ifgram_ohio.rdf 1

# Make Interferogram right swath for land and water and reference
makeInterferogram rdf/SWOT_Ifgram_ohio.rdf 0 0 1 0
makeInterferogram rdf/SWOT_Ifgram_ohio.rdf 1 0 1 0
makeInterferogram rdf/SWOT_Ifgram_ohio.rdf -1 1 1 0

# Add along track response
addAlongTrackPTR rdf/SWOT_Ifgram_ohio.rdf -1 1

# Presumming by 2
addPresum rdf/SWOT_Ifgram_ohio_presum2.rdf 1 -1 1

# Make reference Interferogram right swath at the new presummed coordinates.
makeGDEM rdf/SWOT_Ifgram_ohio_presum2_ref.rdf 1
makeInterferogram rdf/SWOT_Ifgram_ohio_presum2_ref.rdf -1 1 1 0

# Add noise
addNoise rdf/SWOT_Ifgram_ohio_presum2.rdf -1 0 1 1

# multilook the noisy interferogram
makeMultilookInterferogram rdf/SWOT_Multilook_ohio_presum2_Right.rdf

# estimate classification
makeClassification rdf/SWOT_Class_ohio_presum2_Right.rdf

# Compute height using estimated classification
makeWaterDataFile rdf/SWOT_Heights_ohio_presum2_Right_EstClass.rdf

# Compute height using estimated classification
makeLeanPointCloud rdf/SWOT_Heights_ohio_presum2_Right_EstClass.rdf

# Create instrument and wet tropo errors
createInstErrors rdf/SWOT_Heights_ohio_presum2_Right_EstClass.rdf
createWetTropo rdf/SWOT_Heights_ohio_presum2_Right_EstClass.rdf
```

Ohio Example (Left Swath)

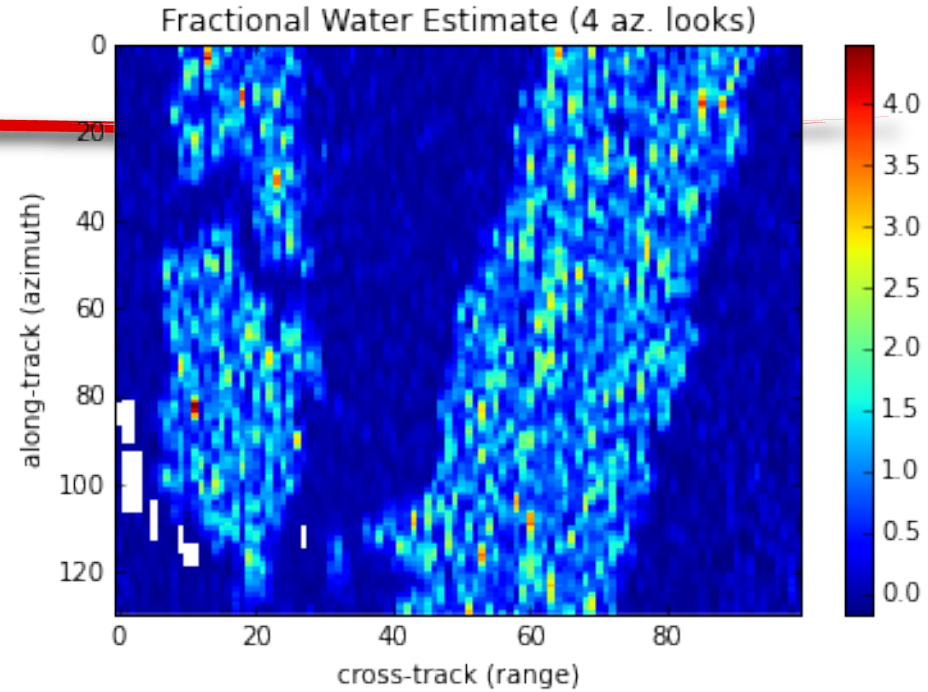


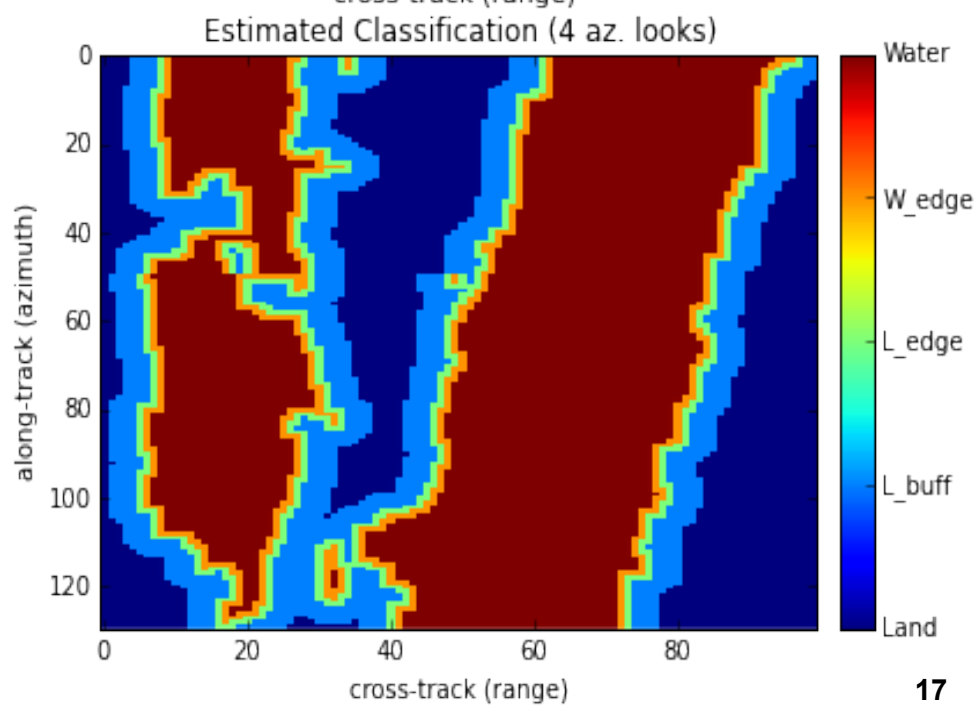
4 az. looks

Current baseline is 4 azimuth looks

- Multi-looking module allows other number of looks (input option)

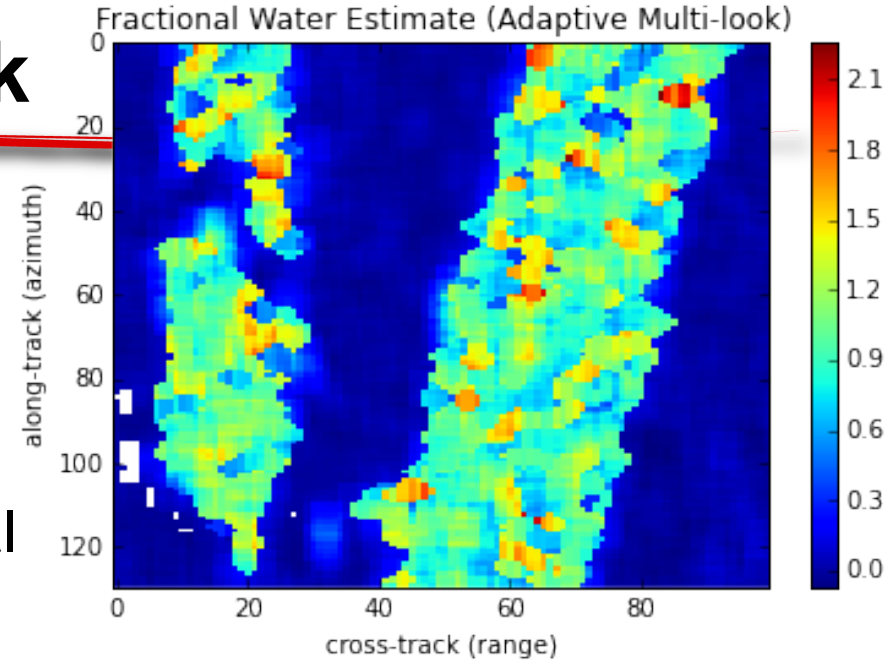
- Fractional water estimator
  - Very noisy
  - Often outside of sensible range (0,1)
- Mixed pixels
  - Flagging using fractional estimator difficult (without more averaging)
  - But can be flagged by using edge boundary as proxy
- Binary classification is good in this part of swath

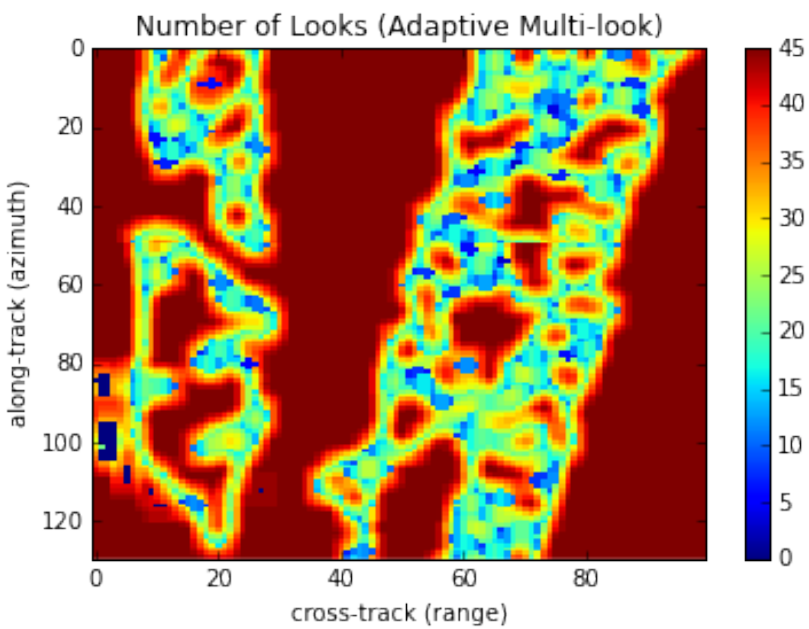


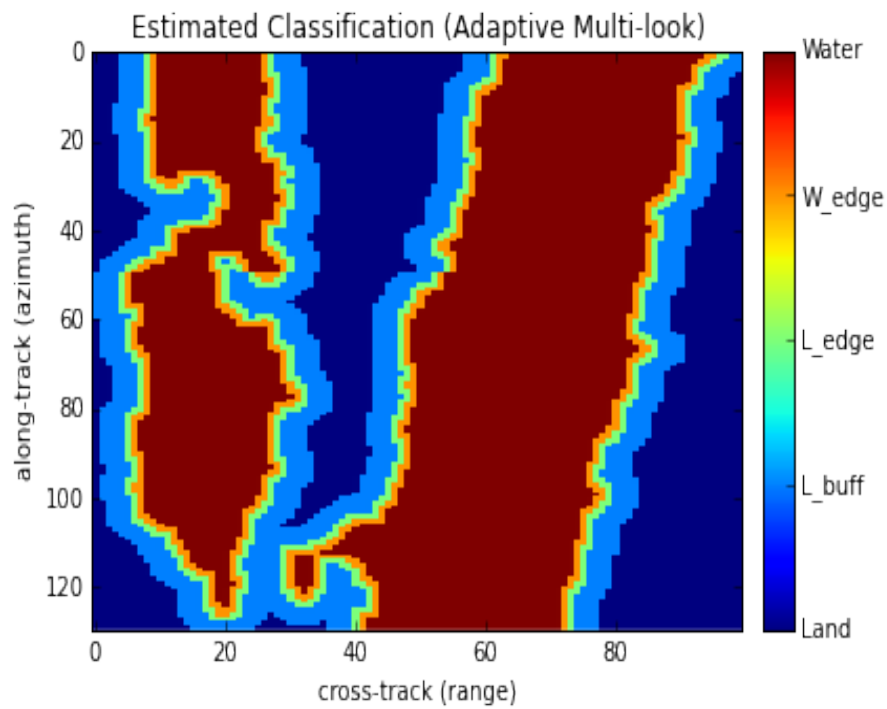
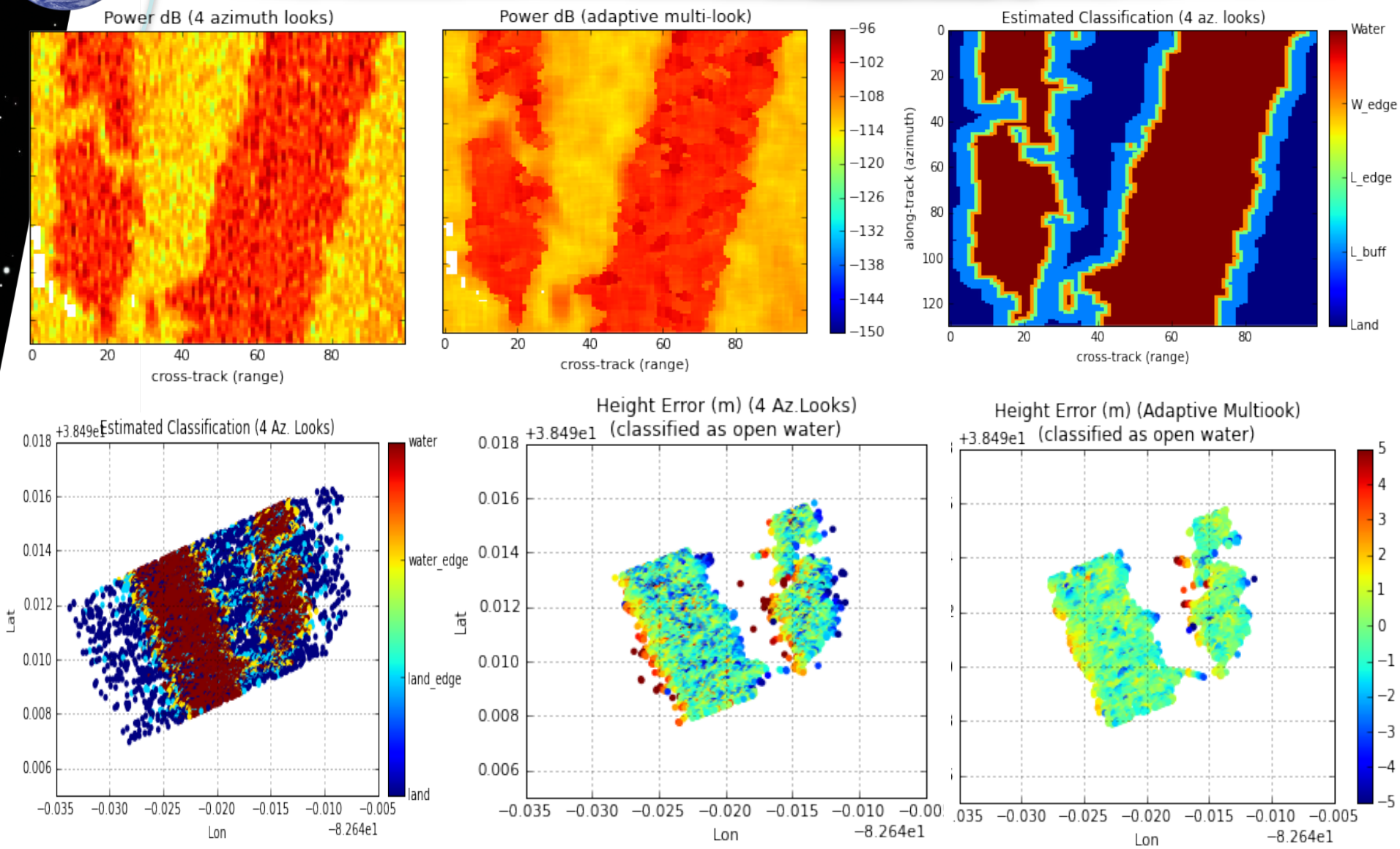


Adaptive Multi-look

- Experimental implementation
  - Edge preserving image filter
  - Watershed segmentation
  - Simple average over window, but only using pixels with the same segmentation label as the center pixel
- Additional averaging helps fractional estimator, but still more averaging needed (100s) to give subpixel geolocation







Ohio Example (Zoomed, Mid-swath)