Birkett Water Mapping NASA Goddard Innovation Lab

# Introduction

This project computes surface water extents from lake boundary maps. It uses the NRT Global Flood Mapping products produced from the LANCE-MODIS data processing system at NASA Goddard to compute the probability of water in each spatial cell within each boundary map. It then thresholds this probability to produce water mask files while filling in nodata values from the local history.

# Workflow

The user configures the workflow by editing the parameters in the spec.yml config file (described below). The processing program is then executed. This program executes the following workflow:

* Determine lat-lon bounding boxes for the lakes of interest using one of the following config methods (from the spec file):
  + A single lat-lon bounding box
  + A csv file with a list of lat-lon bounding boxes
  + A set of geotiff lake boundary masks
    - Lake boundary masks exist for ~400 lakes that show the max extent of the lake plus a nominal buffer.
* For each lake:
  + Download (and merge) the MCDWD NRT tile(s) that overlap the bounding box.
    - The LANCE NRT server holds data for the most recent 8 days (one file per day per tile).
    - New data files are downloaded and added to the data archive.
    - Each tile represents 10 deg lat by 10 deg lon
    - Each file contains a single spatial layer with the following classes:
      * Nodata, Land, Water, Flood, Permanent Flood.
  + Subset the MCDWD data cube using the lake bounding boxes.
    - If a lake boundary mask exists, use it to mask out the pixels outside of the lake area boundary.
  + Make a composite of the previous eight days.
    - For each pixel, count the number of occurrences of water and land over the eight days.
    - Calculate “probability of water” as (# obs water/(# obs water + # obs land)).
    - Compute water masks by thresholding the water probabilities (threshold initially set to 0.5).
  + Fill “nodata” values with the most recent classification value from the MCDWD data.
    - Use a forward fill over time to fill missing values.
  + Calculate area of water within lake region
    - Reproject the result to UTM with a target spatial resolution of 250 x 250 meters
    - Count the number of water pixels and compute total area in sq. km.
    - Similarly, compute the interpolated area by counting interpolated pixels
  + Generate a textfile for each lake with columns containing information about the lake.
    - Append information from the current run to the lake textfile.
    - Output columns: <year>:<day>, water-area, interpolated-water-area

# Application Execution

There are two ways to run the floodmap application:

1. Running with the delivered container (see “Application Configuration” for yml file definition):
2. cd <path-to-container>
3. *singularity pull oras://gitlab.nccs.nasa.gov:5050/cisto-ilab/containers/floodmap:4.1.4*
4. Singularity run -B /explore,/adapt,/gpfsm,/panfs <path-to-container>/floodmap\_4.1.4.sif
5. python /usr/local/ilab/floodmap/exe/production.py <local\_path/lakeID#.yml>
6. Build and execute locally using conda env:

The steps to build using anaconda env, installing from git repo, downloading archived data, configuring and running, as well as visualizing the results of the application can be found in the README file at: <https://github.com/nasa-nccs-cds/floodmap/blob/master/README.md>

# Application Configuration:

The application is configured using a yml file (several samples can be found in the distribution’s *floodmap/specs* directory). Below is a sample spec file (the parameters are explained in the next section). This configuration uses lake mask files. The path to the spec file is passed as the first argument to the execution script.

defaults:

data\_dir: '/explore/nobackup/projects/ilab/projects/Birkett/MOD44W/data'  
 results\_dir: "/explore/nobackup/projects/ilab/projects/Birkett/MOD44W/results"  
 results\_file: 'lake\_{lake\_index}\_stats.csv'  
 water\_class\_thresholds: [ 0.02, 0.93 ]  
 log\_dir: "/explore/nobackup/projects/ilab/cache/floodmap/logs/nrt"  
 ncores: 8  
 parallel: False  
 skip\_existing: False  
 download\_only: False  
 format: 'nc'  
 lake\_masks:  
 basedir: "/explore/nobackup/projects/ilab/projects/Birkett/MOD44W/lakes\_lat\_lon/"  
 file: "{lake\_index}\_2019.tif"  
 lake\_index: 317  
 mask: 3  
 water: 1  
  
 source:  
 url: 'https://nrt3.modaps.eosdis.nasa.gov/api/v2/content/archives/'  
 path: 'allData/{collection}/{product}/Recent'  
 file: "{product}.A{year}{day:03d}.{tile}.{collection:03d}.tif"  
 product: 'MCDWD\_L3\_F1\_NRT'  
 collection: 61  
 token: 'dHBtYXh3ZWw6ZEdodmJXRnpMbTFoZUhkb…1'  
  
 water\_maps:  
 threshold: 0.5  
 bin\_size: 8  
 land\_values: [0]  
 water\_values: [1,2,3]

## Configuration Parameters

The application execution is configured using the following parameters in the spec.yml file:

**data\_dir:** The directory where the downloaded data will be stored (defaults to **results\_dir**).

**results\_dir:** The directory where the processing results are written.

**results\_file:** Template describing the results file name.

**water\_class\_thresholds:** The min and max percent water thresholds calculated over time for each cell. Any cell with a percent water that falls below the min threshold is declared to be permanent land, and any cell with a percent water that falls above the max threshold is declared to be permanent water.

**log\_dir:** The directory where log files will be written.

**ncores:** The number of cores used for parallel processing.

**parallel:** Specifies whether the lakes will be processed in parallel.

**format:** Output map format- can be either "nc" (netcdf) or "tif" (geotif)

**skip\_existing:** If true, will not reprocess any results that already exist.

**download\_only:**  If true, will download the current data files but not run the processing workflow.

**source:** Specifications for reading the floodmap files.

**path:** Path to data on server.  
**url:** URL of NRT data server.

**file:** Template describing the MCDWD file name.

**day:** Target day- will process the previous eight days. Will use today if not specified.

**year:** Target year- will use this year if not specified.

**collection:** Collection index (currently 61)

**product:** Floodmap product (determines composite level, e.g. F2 = two day composite, F1= one day).

**token:** Server authentication token from earthdata.nasa.gov.

**archive\_tiles:** Determines how many tiles are downloaded and archived. There are two options:

**‘current**’**:** Only download tiles that are required for the current set of lakes

**‘global**’**:** Download and archive all new tiles.

**history\_length:** Specifies (in days) how long to keep archived files. Any files that represent times that are more than **history\_length** days in the past are deleted.

**water\_maps:** Parameters for the water mapping workflow:

**threshold:** Probability of water threshold for computing water masks.

**bin\_size:** Number of days to composite when computing water masks.

**land\_values:** Values used to represent land in the MCDWD data.

**water\_values:** Values used to represent water in the MCDWD data.

In addition, there are multiple methods for configuring the lake regions. If lake mask files are available, then one can use the following configuration (as shown above):

**lake\_masks:** Specifications for reading the lake boundary masks or rois.

**basedir:** Base directory of lake mask files.

**file:** Pattern describing file names

**mask:** Mask index value

**water:** Water index value.

**lake\_index\_range:** Range [) of lake indices that will be processed.

If lake mask files are not available, then one can specify a set of lat-lon bounding boxes in a csv file. That file will have five comma-separated columns (first row is the column headers) as follows**: lake\_index, lon-min, lon-max, lat-min, lat-max**. A sample file can be found in the distribution’s *floodmap/specs* directory. The configuration would then be as follows:

lake\_masks:  
 basedir: "/Users/somebody/.floodmap"  
 file: "lake\_bounds.csv"

**basedir:** Directory of the lake bounds file.

**file:** Filename of the lake bounds file.

Alternately, one can specify a single latlon bounding box directly in the spec file:

lake\_masks:  
 roi: 29.2812, 29.9303, -0.7261, -0.0520 lake\_index: 4

**roi:** Lake region bounding box **: lon-min, lon-max, lat-min, lat-max**.

**lake\_index:** Index given to this lake.

# Application Output:

The application will produce output in the directory configured using the **results\_dir** parameter in the *spec.yml* file. For each lake the application will produce a tiff file and a text file. For example, for lake 11 (processed on day 256 of 2021) the files ***lake\_11\_patched\_water\_masks\_2021256.tif*** and ***lake\_11\_patched\_water\_masks\_2021256.csv*** will be produced. The text file (.csv) is the final product, containing the calculated water extents; the tiff file is used for diagnostics. If the process exits with an error then a file called ***lake\_11\_task\_report.txt*** is produced containing the error report.