Installing the software:

Prerequisites: The software provided with the NTR is written in C++. To install the programs, you need to compile and link the provided source code with a C++ compiler.

The instructions provided here assume you have a GNU C++ (gcc) compiler installed under a Linux or UNIX (e.g., Sun Solaris) operating system, or under a Linux-type environment on Windows. Use cygwin (http://www.cygwin.com/) or MinGW-msys (http://www.mingw.org) to provide a Linux-type environment for Windows machines. With appropriate minor modifications, this code should also work with C++ compilers for Windows machines.

The makepp software package is also required to compile and link the provided software. The makepp software can be obtained from http://www.makepp.sourceforge.net. The provided software also requires the GDAL software package which can be obtained from http://trac.osgeo.org/gdal/wiki/DownloadSource.

While not directly required for building and running the water_cloud_mask, find_scans and simulate programs programs, we also include the "Common" software package that provides several utility programs that we use in manipulating the data to set up the examples we provide for using the water_cloud_mask, find_scans and simulate programs.

Building and installing makepp: We recommend installing makepp (and all other executables) in your $HOME/bin directory. If you do this be sure that you have created your $HOME/bin directory before building and installing any of this software. To run the installed software you will have to include that directory in your PATH environment variable.

At the time of this writing, the latest version of makepp was version 2.0. I recommend unpacking the makepp-2.0.tgz file (using tar xvfz) in your $HOME/src directory. Then just enter into the $HOME/src/makepp-2.0 directory and execute the command:

```
perl install.pl
```

and answer the questions appropriately in order to install makepp into $HOME/bin.

Building and installing GDAL: In our tests we used GDAL version 2.4.0. This was NOT the latest version of the software available at the time of this writing, but this is the version that we have been successfully using on a RedHat Linux workstation. We recommend unpacking the gdal-2.4.0.tar.gz (using tar xvfz) in your $HOME/src directory. Then just enter into the $HOME/src/gdal-2.4.0 directory and execute the following sequence of commands:

```
./configure --prefix=$HOME/local --with-libtiff=internal --without-netcdf --with-python
make
make install
```

This installs the gdal libraries, include files and shared data into $HOME/local. To make sure that you access this version of GDAL instead of one of the versions installed in /usr or /usr/local, you need to include $HOME/local/bin in your PATH environment variable before /usr/bin or /usr/local/bin.

Note: Even though the configure specifies '—without-netcdf", this version of GDAL actually does work with HDF5/netCDF image files. The "—without-netcdf" flag is necessary to avoid a conflict with HDF4.

Building and installing the Common software package: We recommend unpacking the CommonV1.70.tar.gz (using tar xvfz) in your $HOME/src directory. Then just enter into the $HOME/src/CommonV1.70 directory and execute the following sequence of commands:

```
cd utilities
makepp
cd ../gdal_utilities
makepp
```

This will install all the utility programs in $HOME/bin. We recommend executing the build of the programs in the utilities directory first, because the build of the gdal_utilities will overwrite some of the programs from the utilities directory with preferred GDAL versions of the same programs (there are some programs in the utilities directory for which there are no preferred GDAL versions).

NOTE: If you want to build the Common software from another directory (not the recommended $HOME/src/Common/V1.70 directory) and install the executables in another directory (not the recommended $HOME/bin) you must edit the first two lines of the standard_defs.mk file accordingly.

Building and installing the provided water_cloud_mask software: I recommend that you install the water_cloud_mask software in your $HOME/src directory. Call this directory $BLDDIR. Place the provided water_cloud_mask.tar.gz file into your build directory and unpack the gzip'd tar files in $BLDDIR using the command "tar xvfz".

To build and install the program, simply type "makepp install" while in the $BLDDIR directory. The executables are installed in $HOME/bin, so make sure you have previously created the $HOME/bin directory.

Typing "water_cloud_mask –h" on the command line produces the following output:

```
The water_cloud_mask progam is called as follows:

water_cloud_mask parameter_file_name

where "parameter_file_name" is the name of the input parameter
file. For contents see below.

For this help: water_cloud_mask -h or water_cloud_mask -help

For version information: water_cloud_mask -v or water_cloud_mask -version
```

The parameter file consists of entries of the form:

-parameter_name parameter_value(s)

The following parameters may be specified in the input parameter file:

Input parameters:

-input_VXX02	(string)	Input VNP02IMG, VNP02MOD or VNP02DNB (for SNPP) or VJ102IMG, VJ102MOD or VJ102DNB (for J1) file name from which the input subdataset is extracted (required)
-input_VXX03	(string)	Input VNP03IMG, VNP03MOD or VNP03DNB (for SNPP) or VJ103IMG, VJ103MOD or VJ103DNB (for J1) file name from which the land_water_mask is extracted (required)
-subdataset	(string)	Subdataset (required)
-bad_detectors	(int)	Bad detectors (Row numbers) (a comma delimited list - optional default: an empty list)

One of the following two forms of Cloud Masks are required:

-input_CLDMSK_L2	(string)	Input CLDMSK_L2_VIIRS HDF file name for cloud_mask
-input_VJ135_L2	(string)	Input VJ135_L2 HDF4 file name for cloud_mask
-cloud_threshold	(float)	Maximum ratio of cloud pixels in chip for chip to be used int water_cloud_mask correlation (optional, default = 0.05)
-water_threshold	(float)	Maximum ratio of water pixels in chip for chip to be used int water_cloud_mask correlation (optional, default = 0.75)
-log	(string)	Output log file (no default)

Output parameters:

-output_no_data_mask	(string)	Output no data mask file name (optional, no default)
-output_cloud_mask	(string)	Output cloud mask file name (optional, no default)
-output_water_mask	(string)	Output water mask file name (optional, no default)
-output_image	(string)	Output image file name for image from subdataset (optional, no default)

The following parameter may also be specified:

-output_format	(string)	Format for output image(s) (optional, default = GTiff (for GeoTIFF), must be a format recognized by GDAL)

Works with SNPP VIIRS Archive Set 5110 and J1 VIIRS Archive Set 3194

Building and installing the provided find_scans and simulate: As with the water_cloud_mask software, we recommend that you install the find_scans and simulate software in your $HOME/src directory ($BLDDIR). Place the provided find_scans.tar.gz and simulateV1.51.tar.gz files into your build directory and unpack the gzip'd tar files in $BLDDIR using the command "tar xvfz".

To build and install each program, simply type "makepp install" while in each program directory ($BLDDIR/find_scans, and $BLDDIR/simulateV1.51). The executables are installed in $HOME/bin, so make sure you have previously created the $HOME/bin directory.

The find_scans and simulate program can be also used with MODIS data. These programs require input of a water mask and can also optionally use a cloud mask. The provided water_cloud_mask software can be used to create water and cloud mask files for SNPP and J1 VIIRS data. Cloud and water masks are available for MODIS data, but the water_cloud_mask program cannot be used to create water and cloud masks for MODIS data. However, the "WaterPresent" mask provided in the MYD03 and MOD03 products is nearly suitable for use with find_scans and simulate. This mask has the value "1" for water pixels and "0" otherwise, which is the inverse of what find_scans and simulate expect. This mask can be inverted, though, to the needed form by using the "compare" utility program provided in the Common software package. While the water mask is required for the find_scans and simulate programs, the cloud mask is optional.

Customization of the find_scans and simulate programs: The find_scans and simulate programs can be customized (tailored) for either the MODIS or VIIRS sensors and for the band types available for each sensor. The customization is effected by defining or undefining compiler flags in the provided defines.h file. The following version of the defines.h tailors find_scans and simulate to process SNPP and J1 VIIRS I-band imagery data:

```
#ifndef DEFINES_H
#define DEFINES_H

#define MODIS
#undef MODIS  // Undefine MODIS if VIIRS (not MODIS) is to be analyzed

#define J1
#undef J1  // Undefine J1 if SNPP VIIRS is to be analyzed

// Comment out (with //) all values of SWATH_SIZE except the desired value.
#ifdef MODIS
//#define SWATH_SIZE 40 // Number of rows in a 250m resolution MODIS swath
//#define SWATH_SIZE 20 // Number of rows in a 500m resolution MODIS swath
//#define SWATH_SIZE 10 // Number of rows in a 1km resolution MODIS swath
#else // Analyze VIIRS data
//#define SWATH_SIZE 16 // Maximum number of rows in an NPP VIIRS data swath for Mbands or DNB
#define SWATH_SIZE 32 // Maximum number of rows in an NPP VIIRS data swath for Ibands
#define DNB
#undef DNB // Undefine if not processing NPP VIIRS or J1 DNB
#endif

#define COLUMN_FACTOR  4  // Number of columns in simulated subset will be COLUMN_FACTOR*SWATH_SIZE
#define SUBSET_PCT 25  // Percentage of pixels that need to be valid in a subset to compute the correlation factor
```

```
#define TIME_SIZE  26  // Dimension size for vector used in timing buffer string
#define PI 3.14159265358

#endif // DEFINES_H
```

When compiled and built with the above version of the defines.h file, typing “find_scans –h” on the command line produces the following output:

The find_scans progam is called as follows:

find_scans parameter_file_name

where "parameter_file_name" is the name of the input parameter
file. For contents see below.

For this help: find_scans -h or find_scans -help

For version information: find_scans -v or find_scans -version

The parameter file consists of entries of the form:

-parameter_name parameter_value(s)

The following parameters may be specified in the input parameter file:

-VIIRS_radiance_image	(string)	Input VIIRS radiance image (single band - required)
-latitude_image	(string)	Input latitude data image (required)
-longitude_image	(string)	Input longitude data image (required)
-water_mask	(string)	Input water mask image (required)
-cloud_mask	(string)	Input cloud mask image (required)
-starting_scan	(int)	Starting scan number for VIIRS image (optional, default = 0)
-water_threshold	(float)	Water pixels threshold (optional, default = 0.2)
-cloud_threshold	(float)	Cloud pixels threshold (optional, default = 0.1)
-Landsat_OLI_mask	(string)	Input Landsat OLI image mask (required)
-found_scans	(string)	Output file for list of scans found to intersect the Landsat image (required)

NOTE: Compiled for VIIRS I-band radiance subimages:
NOTE: If a multiband input image is provided, the first band will be used.

When compiled and built with the above version of the defines.h file, typing “simulate –h” on the command line produces the following output:

The simulate progam is called as follows:

simulate parameter_file_name

where "parameter_file_name" is the name of the input parameter file. For contents see below.

For this help: simulate -h or simulate -help

For version information: simulate -v or simulate -version

The parameter file consists of entries of the form:

-parameter_name parameter_value(s)

The following parameters may be specified in the input parameter file:

-VIIRS_radiance_image	(string)	Input VIIRS radiance image (single band - required)
-latitude_image	(string)	Input latitude data image (required)
-longitude_image	(string)	Input longitude data image (required)
-water_mask	(string)	Input water mask image (required)
-cloud_mask	(string)	Input cloud mask image (optional)
-scan	(int)	Scan number for VIIRS image subset simulated (required)
-column_offset	(int)	Column offset for VIIRS image subset simulated (optional)
-min_column_offset	(int)	Minimum column offset for VIIRS image subset simulated (optional, ignored if column_offset provided)
-max_column_offset	(int)	Maximum column offset for VIIRS image subset simulated (optional, ignored if column_offset provided)
-water_threshold	(float)	Water pixels threshold (optional, default = 0.2)
-cloud_threshold	(float)	Cloud pixels threshold (optional, default = 0.1)
-Landsat_OLI_image	(string)	Input Landsat OLI image (single band - required)
-Landsat_OLI_mask	(string)	Input Landsat OLI image mask (single band - required)
-OLI_column_shift	(int)	OLI image column shift (pixels) (optional, default = 0)
-OLI_row_shift	(int)	OLI image row shift (pixels) (optional, default = 0)
-number_OLI_shifts	(int)	Number of OLI shifts in each direction (optional, default = 1, i.e. no shift search)
-corr_type	(int)	Correlation type 1."Spatial Cross Correlation", 2."Normalized Mutual Information" (default: 1."Spatial Cross Correlation")
-min_corr_peak	(float)	Minimum correlation peak for "best" correlation (default: 0.5)
-Landsat_OLI_subset_image	(string)	Output Landsat OLI subset image (required)
-VIIRS_subset_image	(string)	Output VIIRS subset image (required)
-simulated_VIIRS_image	(string)	Output simulated VIIRS image (required)

NOTE: Compiled for VIIRS I-band radiance subimages:
NOTE: If a multiband input image is provided, the first band will be used.

We recommend building distinct versions of the programs for each sensor and band type as listed in the following table (just rename the executable accordingly after each tailored build):

Executable Name	Sensor	Band Type
find_scansV	VIIRS	I-bands
find_scansM	MODIS	1km bands
simulateI	VIIRS	I-bands
simulateM	VIIRS	M-bands
simulateD	SNPP VIIRS	DNB
simulateD_J1	J1 VIIRS	DNB
simulateQKM	MODIS	250m bands
simulateHKM	MODIS	500m bands
simulate1KM	MODIS	1km bands

We recommend building find_scans only for the VIIRS I-bands, and for only the MODIS 1km bands. This is because the water mask is available at the high spatial resolution of the VIIRS I-bands, so we can take advantage of that to get a more accurate measure of the number of water pixels in a particular scan subset. However, for MODIS the water mask is only available to the coarser resolution 1km bands, so there is no advantage in using the finer resolution MODIS data with find_scans.

Obtaining Landsat and VIIRS or MODIS data for analysis:

It is probably unnecessary to describe how this is done for we expect that anyone who is reading this to already have broad experience in obtaining Landsat, VIIRS and MODIS data. This material is included here for completeness.

Since the Landsat data has less frequent temporal coverage, we like to first look at the available Landsat data for cloud free scenes. I like using GloVis (http://glovis.usgs.gov) for this purpose. We would usually look for Landsat 8 OLI data (if trying to match up with early MODIS data we would, instead, look for Landsat 7 ETM+ data).

I normally select the scenes I want using the GloVis GUI and then obtain the Level-1 GeoTIFF Data Product for the selected scenes using the Bulk Download facility.

To determine the availability of cloud free MODIS or SNPP VIIRS you can go to the LAADS DAAC (https://ladsweb.modaps.eosdis.nasa.gov/). You can also browse for SNPP or J1 VIIRS data at the NPP VIIRS Land Product Quality Assessment web site (https://landweb.modaps.eosdis.nasa.gov/NPP-QA/).

Once the date and time of the desired data sets are determined, they can be downloaded directly from https://ladsweb.modaps.eosdis.nasa.gov/archive/allData/.

Using the water_cloud_mask, find_scans and simulate programs:

We find it convenient to stage the data for analysis in a particular directory structure. We separate the data sets by year and date. For example, we put the data for the year 2019 and day April 20 into the directory $TOPDIR/Year_2019/Apr_20 (where $TOPDIR is directory you are placing all your data into). Under this directory, we use the following directory structure:

$TOPDIR/Year_2019/Apr_20/aqua_1845
$TOPDIR/Year_2019/Apr_20/j1_viirs_1936
$TOPDIR/Year_2019/Apr_20/j1_viirs_1942
$TOPDIR/Year_2019/Apr_20/terra_1700
$TOPDIR/Year_2019/Apr_20/terra_1705
$TOPDIR/Year_2019/Apr_20/viirs_1848

We did not find any appropriate Landsat OLI data for April 20, but we did find Landsat OLI data from April 19 and 21 that we put in the directories:

$TOPDIR/Year_2019/Apr_19/oli
$TOPDIR/Year_2019/Apr_21/oli

For convenience, we put links to ../Apr_19/oli named oli_Apr_19 and ../Apr_21 names oli_Apr_21 into $TOPDIR/Year_2019/Apr_20.

Staging the Landsat data: The Landsat data must be extracted from the data files downloaded from the USGS. To make this easy, we provide several script files and some parameter files for the utility programs. The script files are extract_B#.script (where # is 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 and 11), make_bgri.script, make_mask.script and extract_make_all.script. The parameter files are add_BG.params, add_BGR.params, add_BGRI.params and mask_extract.params. The extract_make_all.script calls the other script files and the add and mask extract programs are called in the make_bgri.script and make_mask.script. The script files need to be edited to make them correspond to the names of the files downloaded from USGS. Once you edit the script files appropriately, just run the extract_make_all.script file and all the Landsat files will be extracted and created.

For the example Landsat 8 OLI data sets from April 19 & 21, 2019, we put all the Landsat 8 gzip'd tar files downloaded from USGS into the directories:

$TOPDIR/Year_2019/Apr_19/oli, and
$TOPDIR/Year_2019/Apr_21/oli

and ran the script file "extract_make_all.script" to extract all of the needed Landsat 8 OLI data sets.

Staging the SNPP VIIRS data: For the example, data was downloaded from archive set 5110 (AS5110). For the example SNPP VIIRS data set from April 20, 2019 at 1848 UTC, we set up the following directory structure:

$TOPDIR/Year_2019/Apr_20/viirs_1848
$TOPDIR/Year_2019/Apr_20/viirs_1848/AS5110

Running the “extract.script” file in the subdirectory AS5110 will extract the needed data files into geoTIFF format. Running water_cloud_mask with the three parameter files wcmask_I.params, wcmask_M.params and wcmask_D.params will produce these files in the appropriate format.

Staging the J1 VIIRS data: For the example J1 VIIRS data sets from April 20 and April 21, 2019, we set up the following directory structure

```
$TOPDIR/Year_2019/Apr_20/j1_viirs_1936
$TOPDIR/Year_2019/Apr_20/j1_viirs_1936/AS3194
$TOPDIR/Year_2019/Apr_20/j1_viirs_1942
$TOPDIR/Year_2019/Apr_20/j1_viirs_1942/AS3194
$TOPDIR/Year_2019/Apr_21/j1_viirs_1918
$TOPDIR/Year_2019/Apr_21/j1_viirs_1918/AS3194
$TOPDIR/Year_2019/Apr_21/j1_viirs_1924
$TOPDIR/Year_2019/Apr_21/j1_viirs_1924/AS3194
```

Running the “extract.script” file in the subdirectories AS3194 will extract the needed data files into geoTIFF format. Running water_cloud_mask with the three parameter files wcmask_I.params, wcmask_M.params and wcmask_D.params will produce these files in the appropriate format.

There is one bad detector in I-band 3. It is necessary to mask out the rows corresponding to this detector in order for the simulate software to work properly. I created the fix_Radiance_I3.script to create a masked version of Radiance_I3.tif. This script file must be edited to create a masked file the same number of rows as in the original Radiance_I3.tif (either 6464 or 6432 lines). The parameter file gen_image.params must also be edited to have the correct number of rows. Once these files are properly edited, run the fix_Radiance_I3.script file and finish the process following the instructions output at the end of the script file. Then double check the resulting Radiance_I3_fixed.tif file to make sure the masking (with the value 65533) was performed properly for the bad data lines. Once the correctness is confirmed, move Radiance_I3_fixed.tif to Radiance_I3.tif.

The process described in the previous paragraph can be shortcut once it is run once for each case (6464 rows and 6432 rows) by simply copying the files RowMask.tif and RowMask65533.tif from a case of the same number of rows and running the commands:

```
multiply multiplyRowMask.params
add addRowMask65533.params
```

Again, once the correctness of Radiance_I3_fixed.tif is confirmed, move Radiance_I3_fixed.tif to Radiance_I3.tif.

Staging the MODIS Terra and Aqua data: For the example MODIS data sets, we set up the following data structure:

$TOPDIR/Year_2019/Apr_20/aqua_1845
$TOPDIR/Year_2019/Apr_20/aqua_1845/coll6.1
$TOPDIR/Year_2019/Apr_21/aqua_1925
$TOPDIR/Year_2019/Apr_21/aqua_1925/coll6.1
$TOPDIR/Year_2019/Apr_20/terra_1700
$TOPDIR/Year_2019/Apr_20/terra_1700/coll6.1
$TOPDIR/Year_2019/Apr_20/terra_1705
$TOPDIR/Year_2019/Apr_20/terra_1705/coll6.1

Running the “extract.script” file in the subdirectories coll6.1 will extract the needed data files into geoTIFF format. The compare utility can be run with the provided compare.params parameter file to produce the water mask in the appropriate format for use with find_scans and simulateV1.51.

Processing the data: The general process scenario is a follows:

i. Use the water_cloud_mask program to produce water and cloud masks for VIIRS I-band and M-band data, and the water mask for the VIIRS DNB data (there is no cloud mask available for the DNB). Extract the WaterPresent mask from the MODIS MOD03 or MYD03 products and invert this mask using the compare utility program to create a properly formatted water mask. (The water_cloud_mask program cannot be used to produce a MODIS cloud mask from the MOD35 or MYD35 products.)
ii. Use find_scans to find which scans intersect with a particular Landsat OLI (or ETM+) data set and are not over too much water area and (optionally) do not include too many cloud pixels. This program uses the Landsat data mask along with a water mask and (optionally) a cloud mask.
iii. Use simulate to process each scan selected by find_scans to find the best subset in each selected scan. Version 1.51 of this program uses required Landsat image and water masks and can optionally use a cloud mask.

Once find_scans is determined to have found eligible scans, step iii can be automated with a script.

We found it convenient to stage the data for analysis in a particular directory structure. We separated the data sets by year and date. For example, we put the data for the year 2019 and day April 20 into the directory $TOPDIR/Year_2019/Apr_20 (where $TOPDIR is the directory you are placing all your data into). In the previous sections, we described this directory structure for the available SNP VIIRS, J1 VIIRS, MODIS Aqua and MODIS Terra data sets. In addition to the directory structure described above, we added an additional directory for each available Landsat OLI dataset:

$TOPDIR/Year_2019/Apr_20/LC08_L1TP_pathrow
$TOPDIR/Year_2019/Apr_21/LC08_L1TP_pathrow

The “pathrow” in the above refers to the path and row of the Landsat OLI data. We set up separate directories for the processing of the data intersecting each particular Landsat OLI data set.

For example, for Landsat OLI path 024 and row 030, find_scans found that scans 147 through 163 intersect with the SNPP VIIRS UTC 1848 data set and are sufficiently water and cloud free for analysis. In the directory $TOPDIR/Year_2019/Apr_20/LC08_L1TP_024030/viirs_1848 you will find a set of parameter files and scripts that I used to semi-automatically process these particular scans. From that directory the processing is performed with the following commands:

```
$ find_scans find_scans.params
$ more found_scans.txt          // to confirm that eligible scans were found
$ ./run_simulate.script
```

A similar approach can be used for J1 VIIRS and MODIS data.

The example.tar.gz gzip’d tar file contains all the parameter and script files mentioned in the recommended directory structure. However, the mentioned VIIRS, MODIS and Landsat data files are not included.