# MODIS SNOW COVER

## 1.INTRODUCTION

The MODIS mission recently released two cloud gap filled data products that are updated on a daily basis. The NDSI\_Snow\_Cover data is the result of the NDSI snow detection algorithm with the cloud mask, ocean mask and night mask overlaid. Snow cover is given in the range of 0-100%, which is the NDSI value of a pixel.A pixel with NDSI > 0.0 is considered to have some snow present. A pixel with NDSI ≤ 0.0 is a snow free land surface. The NDSI is effective at detecting snow cover on the landscape when skies are clear, and viewing geometry and solar illumination are good. Snow cover always has an NDSI >0.0 but not all surface features with NDSI > 0.0 are snow cover. Some surface features, e.g. salt pans, or cloud contaminated pixels at edges of cloud, can have NDSI > 0.0 and be erroneously detected as snow cover, which results in a snow commission error (detecting snow where there is no snow). Snow commission errors are frequently associated with cloud fringes. To alleviate snow commission errors, several data screens based on snow spectral features or other characteristics are applied in the algorithm. The screens are used to reverse snow cover detection or are used to flag uncertain snow cover detection. Snow omission errors occur infrequently. In the algorithm the NDSI is calculated for all land and inland water bodies in daylight, then the data screens are applied to snow detections. All the data screens are applied to each snow pixel.

**Data Screens Applied**

A pixel that has been determined to have some snow present, a snow pixel, is subjected to the following series of screens to alleviate snow commission errors and to flag uncertain snow detections. The first screen is a low visible reflectance screen. Applying all the data screens to a pixel allows for more than one data screen to be set for a snow commission error or uncertain snow detection. A snow pixel that fails any single data screen will be reversed to ‘not snow’ and since all the data screens are applied more than a single QA algorithm bit flag may be set. The same data screens are applied to land and inland water pixels. Inland water bodies are mapped with bit 0 of the algorithm bit flags. The cloud mask, ocean mask, and night mask are laid on the NDSI snow cover to make a thematic map of snow cover. The NDSI value is output for all land and inland water pixels.

The objective in C6 is to minimize snow cover detection errors of omission and commission for the purpose of mapping snow cover extent (SCE) accurately on the global scale. To reach that objective a “snow-conservative approach” was taken in the algorithm. The snow-conservative approach focuses on detection of snow wherever it might be present based on reflectance features, then screens for false snow detections. Detection of snow is pushed to the limits e.g., under low illumination conditions, high solar zenith angles (SZAs) and shadowed surfaces

The surface temperature screen that was used in C5 to reverse a snow detection to a “not snow” decision if the surface was too warm is now linked to surface height and does not change a snow detection at high elevations, > 1300 m. Instead, a bit flag is set to flag a pixel that was detected as “warm snow” while at lower elevations a snow detection is reversed. That approach alleviates the significant problem in C5 where high elevation snow cover on mountains in the spring or summer was reversed to “no decision” by the surface temperature screen (see http://modis-snow-ice.gsfc.gov/?c=collection6 . A

Revisions for C6 are more focused on improving snow detection in clear sky conditions vs. making dramatic improvements in cloud/snow discrimination. A notable cloud/snow confusion situation that can occur is associated with fringes of clouds that are not detected as “certain cloud” by the cloud mask. Also in C6, data content is significantly revised, and snow cover is reported as Normalized Difference Snow Index (NDSI) snow cover instead of as Fractional Snow Cover (FSC) as was done in C5. The NDSI snow cover index is related to the presence of snow in a pixel and is a more accurate description of the snow detection as compared to FSC.

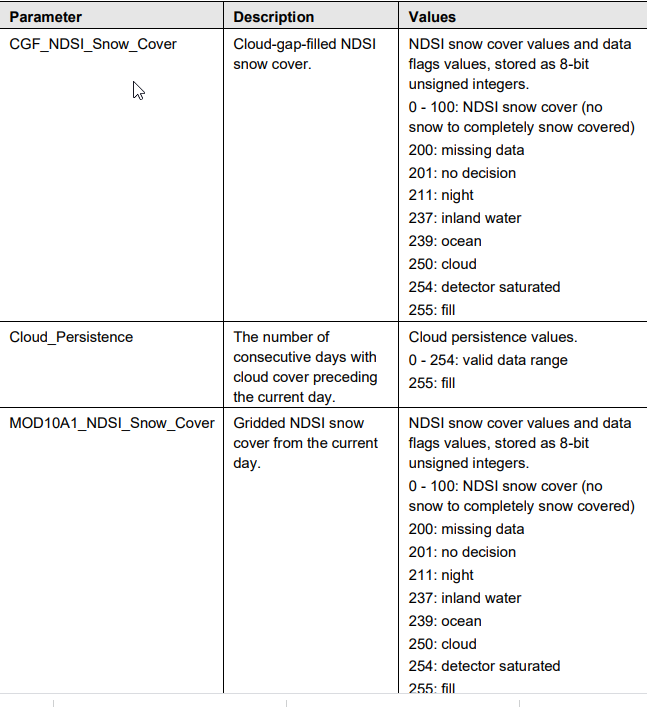
;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

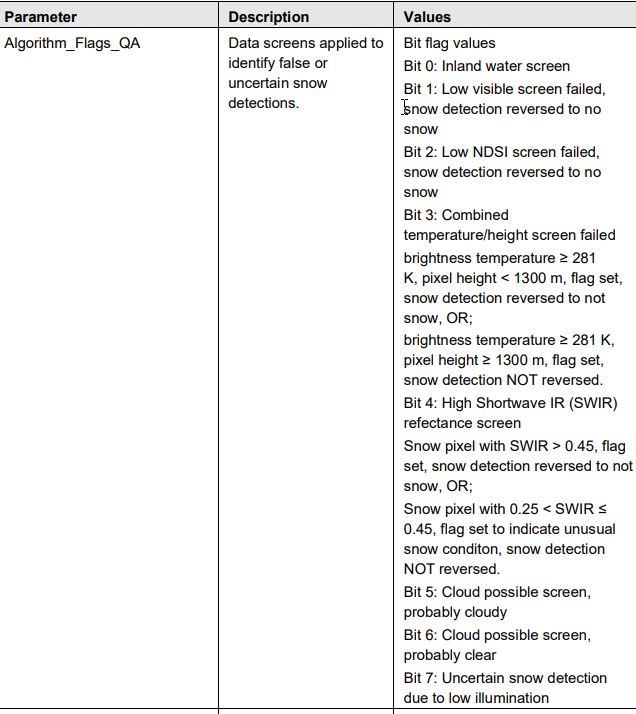
National Snow & Ice Data Center (NSIDC) hosts MOD10A1F.061 snow products.

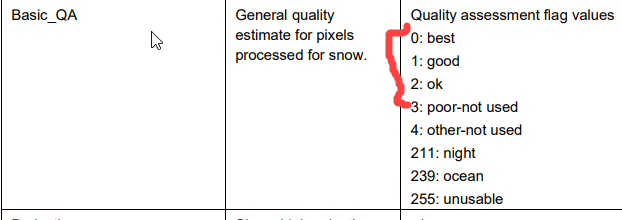
New Snow Cover Data Product in C6.1 is a daily cloud-gap-filled (CGF) tiled snow cover product. The CGF snow cover product is produced from the daily tiled M\*D10A1 and the previous day M\*D10A1F. Daily gaps in observations caused by cloud cover are filled by retaining the previous clear view data for a cell if the current day is cloud obscured (Hall et al., 2010). A data layer that tracks the number of days since last clear view of a cell is included in the product.

As shown in the following figure, each data file includes three Scientific Data Sets (CGF\_NDSI\_Snow\_Cover, Cloud Persistence, and MOD10A1\_NDSI\_Snow\_Cover) and two quality fields (Basic\_QA and Algorithm\_Flags\_QA).

Those files were created using daily MOD10A1 files and their NDSI (normalized diference snow index) is used to create snow cover. All pixels were screened for different flags from QA band. Wherever clouds present, algorithm utilizes cloud free clear pixels from previous days.







## **2. SCRIPTING PHASE**

The first thing first is to create virtual environment with necessary libraries. In this case we use anaconda evnironments where the most important one has gdal library installed. Download process requreires only requests library to read and parse server.

We have a few python scripts dealing with data identification and download. We download following tiles: h10v03, h10v04,h11v02, h11v03, h12v02, h12v03.

The first is[**ident\_downl\_hdf.py**](scripts/ident_downl_hdf.py)and this script hits National Snow & Ice Data Center page at <https://n5eil01u.ecs.nsidc.org/MOST/MOD10A1F.061>. This repository host the data from 2000 until present. It is based on [***requests***](https://pypi.org/project/requests/) python library. Two functions in the script are:

*def re\_read\_urlbase(url\_base):*

''' read the content from the web and get ready for parsing '''

*def get\_MOD10A1F\_hdfs():*

'''read the NSIDC web content, parse it for daily folders

identify the most recent folder and go and grab 4 HDF files covering ALBERTA province '''

The second function outputs a text file with links to the four-hdf files covering Alberta province.

Another file, [**bitBucket.py**](file:///C:\_LOCALdata\_PROJECTS_hist\prj_2021\NDSI\bitBucket.py)**,** usesthe output text file to do a downloading process.

>python bitBucket.py -d C:\\_LOCALdata\\_PROJECTS\_hist\prj\_2021\NDSI -f C:\\_LOCALdata\\_PROJECTS\_hist\prj\_2021\NDSI\find\_files2down\mod10A1f\_list\_2.txt

Now, when we have downloaded our files, we use our environment having gdal installed and script called [**MOD\_proc\_functs\_fin.py**](scripts/MOD_proc_functs_fin.py)**.**

This script process all hdf files and create final output. This output will be copied to various servers within Agriculature and Forestry and Envrioment and Parks

At the end we communicate to distribution centre that product is placed onto the server. For this step [***final\_goa\_1.py***](scripts/final_goa_1.py) ***script*** automatically sends an email to a recipients telling them that file is available.

**BATCH process**

**We can create a batch file and put it into a task scheduler to run on a daily basis. There is one issue though. Download of the hdf files cuts short. Most of the specified files are in size greater than 100 Mb, but batch process downloads o**

**@echo off**

**echo %DATE%**

**echo %TIME%**

**python -V**

**@echo OFF**

**rem How to run a Python script in a given conda environment from a batch file.**

**rem It doesn't require:**

**rem - conda to be in the PATH**

**rem - cmd.exe to be initialized with conda init**

**rem Define here the path to your conda installation**

**set CONDAPATH=C:\ProgramData\Anaconda3**

**rem Define here the name of the environment**

**set ENVNAME=modisd**

**rem The following command activates the base environment.**

**rem call C:\ProgramData\Miniconda3\Scripts\activate.bat C:\ProgramData\Miniconda3**

**if %ENVNAME%==base (set ENVPATH=%CONDAPATH%) else (set ENVPATH=%UserProfile%\AppData\Local\conda\conda\envs\%ENVNAME%)**

**rem Activate the conda environment**

**rem Using call is required here, see: https://stackoverflow.com/questions/24678144/conda-environments-and-bat-files**

**call %CONDAPATH%\Scripts\activate.bat %ENVPATH%**

**rem Run a python script in that environment that identify and download hdf files**

**python U:\RS\_Task\_Workspaces\MODIS\_reflectance\scripts\** [**ident\_downl\_hdf.py**](scripts/ident_downl_hdf.py)

**rem Deactivate the environment**

**rem call conda deactivate**

**call deactivate**

**rem If conda is directly available from the command line then the following code works.**

**rem call activate someenv**

**rem python script.py**

**rem conda deactivate**

**rem One could also use the conda run command**

**rem conda run -n someenv python script.py**

Reference

1. <https://nsidc.org/data/mod10a1f>
2. <https://nsidc.org/data/myd10a1f>

in Edmonton

March 19, 2021