



## Theia Land Data Centre

---

# Algorithm theoretical basis documentation for an operational snow cover extent product from Sentinel-2 and Landsat-8 data (Let-it-snow)

---

Simon Gascoin (CNRS/CESBIO), Manuel Grizonnet (CNES/CT/DSI),  
Tristan Klempka (CNES/CT/DSI), Germain Salgues (Magellum)  
V1.0 (Updated for LIS 1.2) - September 12, 2018

## Abstract

This document describes the algorithm of the Let-it-snow (LIS) processing chain to generate the snow cover extent product for the Theia land data center. The algorithm takes as input a Sentinel-2 or Landsat-8 image of surface reflectance corrected from atmospheric and slope effects, the associated cloud mask (level 2A product provided by Theia) and a digital elevation model. The output is a single band raster at the same resolution of the input image giving the snow presence or absence and a cloud mask. The output cloud mask is different from the input cloud mask because some pixels can be reclassified as snow or no-snow by the algorithm.

The snow detection algorithm works in two passes: first the most evident snow cover is detected using a set of conservative thresholds, then these snow pixels are used to determine the lowest elevation of the snow cover. A second pass is performed for the pixels above this elevation with a new set of less conservative thresholds.

The processing chain also generates a vectorized version of the snow mask after pass 1 and 2 and a color composite that is overlaid by these polygons. These secondary products are intended for expert validation purpose.

## Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Motivation . . . . .	3
1.2	Objective . . . . .	3
1.3	Development . . . . .	4
1.4	Limitations . . . . .	4
<b>2</b>	<b>Algorithm</b>	<b>4</b>
2.1	Inputs . . . . .	4
2.2	Outputs . . . . .	5
2.3	Pre-processing . . . . .	6
2.4	Snow detection . . . . .	6
2.5	Snowline elevation . . . . .	6
2.6	Cloud mask processing . . . . .	6
2.7	Parameters description . . . . .	9
2.7.1	Main algorithm parameters . . . . .	9
2.7.2	JSON schema of configuration file . . . . .	10
<b>3</b>	<b>Validation</b>	<b>13</b>
<b>4</b>	<b>Conclusion and perspectives</b>	<b>18</b>
<b>A</b>	<b>Test script (castest_CESneige.m)</b>	<b>19</b>
<b>B</b>	<b>Snow detection function (S2snow.m)</b>	<b>25</b>

---

# 1 Introduction

## 1.1 Motivation

The snow cover is a key factor of many ecological, climatological and hydrological processes in cold regions. The monitoring of the snow cover is of particular societal relevance in mountain regions since the seasonal snow melt modifies the soil moisture, groundwater recharge and river flow, often providing critical water resources to downstream areas<sup>1</sup>.

The snow cover is one of the 50 Essential Climate Variables (ECVs) that were defined by the Global Observing System for Climate (GCOS)<sup>2</sup> in accordance with the Committee on Earth Observation Satellites (CEOS) agencies<sup>3</sup> to support the work of the UNFCCC and the IPCC.

The *snow cover extent* or *snow cover area* is the extent of the snow cover on the land surface. A snow cover extent product is typically formatted as a georeferenced raster image whose pixel value indicate if snow is present or absent in the pixel.

Other major satellite snow products include: (i) the snow cover fraction (ii) snow albedo (iii) the snow water equivalent. The snow cover fraction and albedo are generated from optical observations, while the snow water equivalent is retrieved using passive or active microwave. The snow water equivalent is potentially the most useful product since it gives directly the amount of accumulated water (snow mass), however current products are unsuitable to address user needs for many applications and places because they are available at coarse scale (25 km) with some limitations in the retrievals. This is due to the lack of observations in the wavelengths that are adapted to snow water equivalent sensing. As of today the snow cover extent product is still the most widely used for hydrological and climatological applications. This will certainly remain true for the next decade since there is no planned mission to retrieve the snow water equivalent or the snow depth at global scale<sup>4</sup>.

Current snow cover area products are derived from low to mid-resolution optical observations (e.g. AVHRR, VEGETATION, MODIS) but their spatial resolution (1 km to 250 m) is too coarse for various applications, in particular in mountain regions where the topography causes large spatial variability of the snow cover at decametric scales. High resolution snow cover maps can be generated from Landsat images but the temporal revisit of 16 days is not sufficient for snow cover monitoring during the melt season. The ESA Sentinel-2 mission offers the unique opportunity to study the snow cover extent dynamics at 20 m resolution with a 5 day revisit time. If combined with Landsat the temporal resolution can be further increased. Both Sentinel-2 and Landsat missions are global missions that are expected to run over long periods, allowing the development of operational products and services<sup>5</sup>.

## 1.2 Objective

The objective of this algorithm is to generate a snow cover extent product from Landsat-8 and Sentinel-2 images at high resolution (30 m for Landsat-8, 20 m for Sentinel-2). The main requirements are:

- The algorithm should be efficient to allow the processing of large areas ( $10^4 \text{ km}^2$ ) with a reasonable computation cost.
- It should be robust to seasonal and spatial variability of the snow cover and land surface properties.
- It should maximize the number of pixels that are classified as snow or no-snow.
- It is always preferable to falsely classify a pixel as cloud than falsely classify a pixel as snow or no-snow.

---

<sup>1</sup>Barnett T. P., Adam J. C. and Lettenmaier D. P., Potential impacts of a warming climate on water availability in snow-dominated regions, *Nature* 438 (7066), 2005.

<sup>2</sup>GCOS Essential Climate Variables <http://www.wmo.int/pages/prog/gcos/index.php?name=EssentialClimateVariables>

<sup>3</sup>Global Climate Observing System (GCOS) Implementation Plan <http://remotesensing.usgs.gov/ecv/document/gcos-138.pdf>

<sup>4</sup>This was the objective of the CoreH2O mission but the project was not selected by the ESA for the Earth Explorer-7 program in 2015.

<sup>5</sup>Drusch M., Del Bello U., Carlier S. et al., Sentinel-2: ESA's Optical High-Resolution Mission for GMES Operational Services, *Remote Sensing of Environment* 120, 2012.

### 1.3 Development

The algorithm prototype was developed by Simon Gascoin with insights from Olivier Hagolle in June 2015. The snow detection function and a script to run this function with an example are given in appendices A and B as formatted documents that includes the original Matlab code, comments, and output. The LIS chain was designed to work on any high resolution multi-spectral images from satellite sensors that include at least a channel in the visible spectrum and a channel near 1.5  $\mu\text{m}$  (typically referred to as mid-infrared or “MIR”). This initial code was ported to Python 2.7 and C++ by Manuel Grizonnet in order to make it scalable to large images using Orfeo Toolbox and GDAL.

LIS currently supports SPOT-4, SPOT-5, Landsat-8 and Sentinel-2 level 2A products.

The LIS code, installation documentation and configuration file examples are available in the Cesbio’s gitlab: <http://tully.ups-tlse.fr/grizonnet/let-it-snow>.

The list of all contributors is available in the LIS source in the file README.md.

### 1.4 Limitations

The product is based on optical observations therefore it is not adapted to the detection of the snow cover:

- in polar regions when illumination is insufficient ;
- in dense forest areas where the ground is obstructed by the canopy, like in evergreen conifer forests.

The algorithm may also fail to detect the snow cover in steep shaded slopes if the solar elevation is very low (typically below 20°). This can occur in mid-latitude areas in winter. In this case the slope correction in the L2A product is generally not applied as indicated in the L2A mask.

The algorithm can only reduce the number of cloud pixels from the original L2A cloud mask. If a cloud was not detected by the previous cloud mask algorithm (MACCS) then it can only be classified as snow or no-snow.

The algorithm output depends on the scale of the input because the snowline elevation is computed at the scale of the image. In the case of the level 2A products this is 110 km by 110 km. The underlying assumption is that a large altitudinal variation of the snowline elevation is not likely at such a scale. Our impression is that this assumption is supported by regional analyses of the snowline in many mountain ranges, but this could be further assessed using mid-resolution snow products<sup>67</sup>.

## 2 Algorithm

### 2.1 Inputs

- From a level 2A product:
  - the cloud and cloud shadow mask (referred to as “L2A cloud mask” in the following),
  - the green, red and MIR bands from the flat surface reflectance product (Tab. 2.1). These images are corrected for atmospheric and terrain slope effects. The slope correction is important in mountain regions since it enables to use the same detection thresholds whatever the sun-slope geometry.
- A digital elevation model (DEM). The DEM is resampled from the SRTM seamless DEM<sup>8</sup>.
- The parameters of the algorithm:  $r_f$ ,  $d_z$ ,  $r_B$ ,  $r_D$ ,  $n_1$ ,  $n_2$ ,  $r_1$ ,  $r_2$ ,  $f_s$ ,  $f_t$  (written in red throughout the document).

<sup>6</sup>Gascoin, S., Hagolle, O., Huc, M., Jarlan, L., Dejoux, J.-F., Szczypta, C., Marti, R., and Sánchez, R.: A snow cover climatology for the Pyrenees from MODIS snow products, Hydrol. Earth Syst. Sci., 19, 2337-2351

<sup>7</sup>Krajčí P., Holko L. and Parajka J., Variability of snow line elevation, snow cover area and depletion in the main Slovak basins in winters 2001–2014, Journal of Hydrology and Hydromechanics 64(1), 2016

<sup>8</sup>Jarvis A., H.I. Reuter, A. Nelson, E. Guevara, 2008, Hole-filled seamless SRTM data V4, International Centre for Tropical Agriculture (CIAT), available from <http://srtm.csi.cgiar.org>.

Sensor	Band		
	Green	Red	MIR
SPOT-4 HRV	1 (20 m, 0.55 µm)	2 (20 m, 0.65 µm)	4 (20 m, 1.6 µm)
SPOT-5 HRG	1 (10 m, 0.55 µm)	2 (10 m, 0.65 µm)	4 (10 m, 1.6 µm)
Sentinel-2 MSI	2 (10 m, 0.56 µm)	3 (10 m, 0.66 µm)	5 (20 m, 1.6 µm)
Landsat-8 OLI	3 (30 m, 0.56 µm)	4 (30 m, 0.65 µm)	6 (30 m, 1.6 µm)

Table 1: Index of the spectral band in the L2A flat surface reflectance products used by LIS. In parentheses is also indicated the spatial resolution and the wavelength of the band center.

## 2.2 Outputs

The main output is a raster image (\*SEB.TIF) of the snow and cloud mask. It has the same projection and extent of the initial L2A product and the same resolution as the MIR band, i.e. 20 m for Sentinel-2 and SPOT-4, 30 m for Landsat-8 (Tab. 2.1). It coded as follows:

- 0: no-snow
- 100: snow
- 205: cloud including cloud shadow
- 254: no data

The same data are made available as polygons (ESRI Shapefile format) of the cloud and snow cover extent (\*SEB\_VEC\*). Two fields of information are embedded in this file:

- DN:
  - 0: no-snow
  - 100: snow
  - 205: cloud including cloud shadow
  - 254: no data
- field:
  - no-snow
  - snow
  - cloud
  - no-data

The other output files are rather useful for the expert evaluation and troubleshooting:

- an RGB color composite image of bands MIR/red/green also showing the snow and cloud mask boundaries (\*COMPO.TIF);
- a binary mask of snow and clouds (\*SEB\_ALL.TIF):
  - bit 1: snow (pass 1)
  - bit 2: snow (pass 2)
  - bit 3: clouds (pass 1)
  - bit 4: clouds (pass 2)
  - bit 5: clouds (initial all cloud)
  - bit 6: slope flag (optional bad slope correction flag)
- a metadata file (\*METADATA.XML)

## 2.3 Pre-processing

In the case of Sentinel-2 the red and green bands are first resampled with the cubic method to a pixel size of 20 m by 20 m to match the resolution of the SWIR band.

The DEM is also resampled to the resolution of the target product (30 m or 20 m, see Sect. 2.2) using the cubic spline method that is implemented in the GDAL library.

## 2.4 Snow detection

The snow detection is based on the Normalized Difference Snow Index (NDSI) and the reflectance in the red band. The NDSI is defined as<sup>9</sup>:

$$\text{NDSI} = \frac{\rho_{\text{green}} - \rho_{\text{MIR}}}{\rho_{\text{green}} + \rho_{\text{MIR}}} \quad (1)$$

where  $\rho_{\text{green}}$  (resp.  $\rho_{\text{MIR}}$ ) is the slope-corrected surface reflectance in the green band (resp. MIR at 1.6  $\mu\text{m}$ ). The NDSI is based on the fact that only snow surfaces are very bright in the visible but dark in the shortwave infrared. Some lake pixels may also have a high NDSI value so we add a criterion on the red reflectance to remove these. A pixel is classified as snow if the two following conditions are fulfilled:

- $\text{NDSI} > n_i$ ,
- $\rho_{\text{red}} > r_i$

where  $n_i$  and  $r_i$  are two parameters with  $i = \{1, 2\}$ . Otherwise the pixel is marked as no-snow.

## 2.5 Snowline elevation

The snow detection (Sect. 2.4) is performed a first time using thresholds  $n_1$  and  $r_1$ . The parameters are set to low values to minimize the false snow detections. As a consequence, many snow covered areas are not detected. However, this pass 1 enables to estimate a minimum snow cover elevation  $z_s$ . For that purpose the DEM is used to segment the image in elevation band of height  $d_z$ . The fraction of the cloud-free area of each band that is covered by snow is computed. We find the lowest elevation band  $b$  at which the the snow cover fraction is greater than  $f_s$ . Then,  $z_s$  is defined as the lower edge of the elevation band that is two elevation bands below band  $b$ . The snow cover fraction in each elevation band is determined using the pixels that are not marked as cloud in the pass 1 cloud mask (Sect. 2.6). To ensure that  $z_s$  is computed with a statistically significant sample of pixels, the snowline calculation is not activated if the total fraction of snow pixels in the image is lower than  $f_t$ . A detailed example of the determination of  $z_s$  is given in appendix A.

## 2.6 Cloud mask processing

The L2A cloud mask is conservative because it is computed at a coarser resolution and also because it is developed for a large range of applications. However, the detection of the snow cover is robust to a thin, transparent, cloud cover. More importantly, the L2A cloud mask tends to falsely classify the edges of the snow cover as cloud. Hence, it is possible to recover many pixels from the L2A cloud mask and reclassify them as snow or no-snow. This step is important because it substantially increases the number of observations. A pixel from the L2A cloud mask cannot be reclassified as snow or no-snow if:

- it is coded as “cloud shadow” in L2A cloud mask. Note that it can be cloud shadows matched with a cloud or cloud shadows in the zone where clouds could be outside the image ;
- or: it is coded as “high altitude cloud” (or “cirrus”) in the L2A cloud mask;
- or: it is not a “dark” cloud (see below).

The cloud shadows are excluded because the signal-to-noise ratio is too low in these areas.

The “high clouds” are excluded because they can have a similar spectral signature as the snow cover (high reflectance in the visible and low reflectance in the MIR). This type of cloud is only detected in Landsat-8 and Sentinel-2 images because it is based on the spectral band centered on the 1.38  $\mu\text{m}$  wavelength<sup>10</sup>, therefore the test is not activated for SPOT images.

<sup>9</sup>Dozier, J.: Spectral signature of alpine snow cover from the Landsat Thematic Mapper, Remote sensing of Environment 28, 9–22, 1989

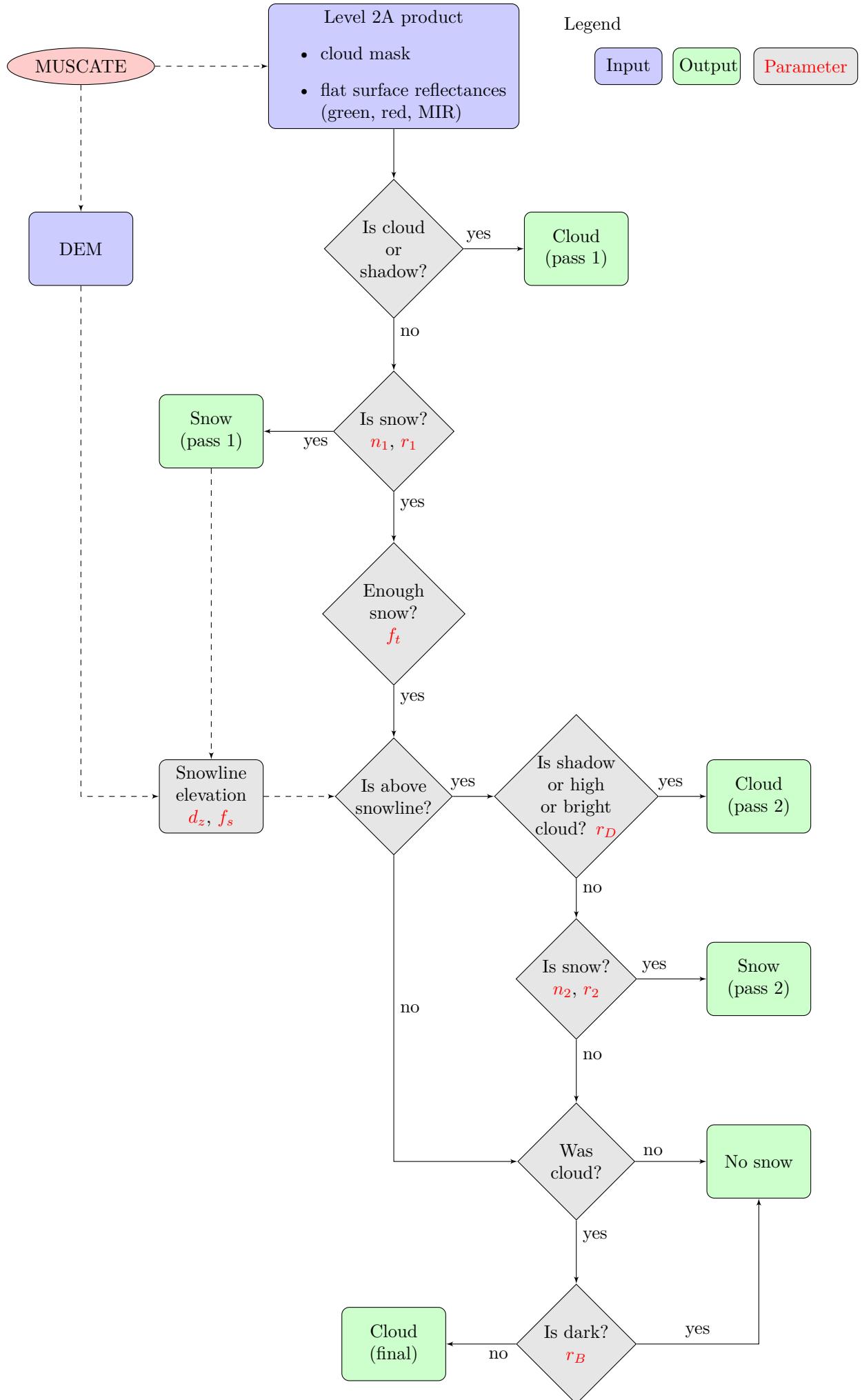
<sup>10</sup>Hagolle, O., High cloud detection using the cirrus band of LANDSAT 8 or Sentinel-2, <http://www.cesbio.ups-tlse.fr/multitemp/?p=4109>

We select only the “dark clouds” because the snow test is robust to the snow/cloud confusion in this case. The “dark” clouds are defined using a threshold in the red band after down-sampling the red band by a factor  $r_f$  using the bilinear method. This resampling is applied to smooth locally anomalous pixels<sup>11</sup>. Therefore, if a (non-shadow, non-high-cloud) cloud pixel has a red reflectance at this coarser resolution that is lower than  $r_D$  then it is temporarily removed from the cloud mask and proceeds to the snow test. The new cloud mask at this stage is the pass 1 cloud mask (Fig. 2.6).

After passing the pass 1 and 2 snow tests, some pixels that were originally marked as cloud will not be reclassified as snow. These pixels are marked as cloud if they have a reflectance in the red that is greater than  $r_B$ . Otherwise they are classified as no-snow. Here the full resolution red band is used. The resulting cloud mask is the pass 2 cloud mask.

---

<sup>11</sup>It was also inspired by the MACCS algorithm, which performs the cloud detection at 240 m for Landsat-8 L2A products.



## 2.7 Parameters description

### 2.7.1 Main algorithm parameters

The table below gives the description of the main parameters of the algorithm:

Parameter	Description	Name in the configuration file	Default value
$r_f$	Resize factor to produce the down-sampled red band	<code>rf</code>	8 for L8 (12 for S2)
$r_D$	Maximum value of the down-sampled red band reflectance to define a dark cloud pixel	<code>rRed_darkcloud</code>	0.300
$n_1$	Minimum value of the NDSI for the pass 1 snow test	<code>ndsi_pass1</code>	0.400
$n_2$	Minimum value of the NDSI for the pass 2 snow test	<code>ndsi_pass2</code>	0.150
$r_1$	Minimum value of the red band reflectance the pass 1 snow test	<code>rRed_pass1</code>	0.200
$r_2$	Minimum value of the red band reflectance the pass 2 snow test	<code>rRed_pass2</code>	0.040
$d_z$	Size of elevation band in the DEM used to define $z_s$	<code>dz</code>	0.100
$f_t$	Minimum snow fraction in an elevation band to define $z_s$	<code>fsnow_lim</code>	0.100
$f_{ct}$	Minimum clear pixels fraction (snow and no-snow) in an elevation band to define $z_s$	<code>fclear_lim</code>	0.100
$f_s$	Minimum snow fraction in the image to activate the pass 2 snow test	<code>fsnow_total_lim</code>	0.001
$r_B$	Minimum value of the red band reflectance to return a non-snow pixel to the cloud mask	<code>rRed_backtocaloud</code>	0.100

Table 2: LIS algorithm parameters description and default values.

Above default values related to reflectance are given as float values between 0 and 1. Threshold related to reflectance values follow the convention of considering milli-reflectance as input (values between 0 and 1000) in the json file. Some products can encode reflectance with other convention (floating values between 0 and 1 or reflectance between 0 and 10000), to handle those cases, there is a parameter 'multi' in the json configuration file which allows to scale reflectance parameters. For instance, for products with reflectance between 0 and 10000 you can use

## 2.7.2 JSON schema of configuration file

The JSON Schema here describes the parameter file format and provide a clear, human- and machine-readable documentation of all the algorithm parameters. JSON schema was generated on [with](#) the following options (with metadata and relative id).

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "id": "http://culy.ups-tlse.fr/grizonnet/let-it-snow/blob/master/test/param_test_s2_template.json",
  "properties": {
    "cloud": {
      "id": "cloud",
      "properties": {
        "all_cloud_mask": {
          "default": 1,
          "description": "Threshold apply to Theia cloud mask to retrieve a strict cloud mask (greater than all_cloud_mask).",
          "id": "all_cloud_mask",
          "title": "The All_cloud_mask schema.",
          "type": "integer"
        },
        "high_cloud_mask": {
          "default": 128,
          "description": "bitmask apply to retrieve high clouds for input mask",
          "id": "high_cloud_mask",
          "title": "The High_cloud_mask schema.",
          "type": "integer"
        },
        "red_backtocaloud": {
          "default": 100,
          "description": "Minimum value of the red band reflectance to return a non-snow pixel to the cloud mask.",
          "id": "red_backtocaloud",
          "title": "The Red_backtocaloud schema.",
          "type": "integer"
        },
        "red_darkcloud": {
          "default": 300,
          "description": "Maximum value of the down-sampled red band reflectance to define a dark cloud pixel.",
          "id": "red_darkcloud",
          "title": "The Red_darkcloud schema.",
          "type": "integer"
        },
        "rf": {
          "default": 12,
          "description": "Resize factor to produce the down-sampled red band (use for cloud refinement).",
          "id": "rf",
          "title": "The Rf schema.",
          "type": "integer"
        },
        "shadow_in_mask": {
          "default": 32,
          "description": "bitmask apply to retrieve cloud shadows (for cloud inside the image).",
          "id": "shadow_in_mask",
          "title": "The Shadow_in_mask schema.",
          "type": "integer"
        },
        "shadow_out_mask": {
          "default": 64,
          "description": "bitmask apply to retrieve cloud shadows (for cloud outside the image).",
          "id": "shadow_out_mask",
          "title": "The Shadow_out_mask schema.",
          "type": "integer"
        },
        "strict_cloud_mask": {
          "default": false,
          "description": "Option that prevent any snow detection within the initial cloud mask. (experimental)",
          "id": "strict_cloud_mask",
          "title": "The Strict_cloud_mask schema.",
          "type": "boolean"
        },
        "rm_snow_inside_cloud": {
          "default": false,
          "description": "Trigger the experimental function discarding snow area that are inside in cloud mask.",
          "id": "rm_snow_inside_cloud",
          "title": "The Rm_snow_inside_cloud schema.",
          "type": "boolean"
        },
        "rm_snow_inside_cloud_dilation_radius": {
          "default": 1,
          "description": "Size in pixel of the dilation radius around the snow area. (experimental)",
          "id": "rm_snow_inside_cloud_dilation_radius",
          "title": "The Rm_snow_inside_cloud_dilation_radius schema.",
          "type": "integer"
        },
        "rm_snow_inside_cloud_threshold": {
          "default": 0.85,
          "description": "Minimum fraction of cloudy pixel in the dilated area to discard the snow area. (experimental)",
          "id": "rm_snow_inside_cloud_threshold",
          "title": "The rm_snow_inside_cloud_threshold schema.",
          "type": "float"
        },
        "rm_snow_inside_cloud_min_area": {
          "default": 25000,
          "description": "Minimum area (in pixels) for snow areas to execute the cold cloud removal. (experimental)",
          "id": "rm_snow_inside_cloud_min_area",
          "title": "The rm_snow_inside_cloud_min_area schema.",
          "type": "int"
        }
      },
      "type": "object"
    },
    "general": {
      "id": "general",
      "properties": {
        "log": {
          "default": true,
          "description": "Log output and error to files (std***.log).",
          "id": "log",
          "title": "The Log schema.",
          "type": "boolean"
        },
        "multi": {
          "default": 10,
          "description": "Scale input parameters to map reflectance interval (parameters are provided in milli-reflectance but L2A S2 Theia product are between 0 and 10000).",
          "id": "multi",
          "title": "The Multi schema."
        }
      }
    }
  }
}
```

```

        "id": "multi",
        "title": "The Multi schema.",
        "type": "integer"
    },
    "nb_threads": {
        "default": 1,
        "description": "Maximum number of threads use by the program.",
        "id": "nb_threads",
        "title": "The Nb_threads schema.",
        "type": "integer"
    },
    "nodata": {
        "default": -10000,
        "description": "No-data value in the input L2A product.",
        "id": "nodata",
        "title": "The Nodata schema.",
        "type": "integer"
    },
    "pout": {
        "description": "Path to output directory.",
        "id": "pout",
        "title": "The Pout schema.",
        "type": "string"
    },
    "preprocessing": {
        "default": false,
        "description": "Activate the extraction and resampling of the DEM.",
        "id": "preprocessing",
        "title": "The Preprocessing schema.",
        "type": "boolean"
    },
    "ram": {
        "default": 1024,
        "description": "Maximum number of RAM memory used by the program.",
        "id": "ram",
        "title": "The Ram schema.",
        "type": "integer"
    },
    "target_resolution": {
        "default": -1,
        "description": "Resolution of the output SNOW products in meter (automatically use the input product resolution in case target_resolution=-1).",
        "id": "target_resolution",
        "title": "The target_resolution schema.",
        "type": "float"
    }
},
    "type": "object"
},
"vector": {
    "id": "vector",
    "properties": {
        "generate_vector": {
            "default": true,
            "description": "Generate vector with snow and cloud masks of the final detection.",
            "id": "generate_vector",
            "title": "The Generate_vector schema.",
            "type": "boolean"
        },
        "generate_intermediate_vectors": {
            "default": false,
            "description": "Generate vector masks of the detection (pass 1 and 2 and final result).",
            "id": "generate_intermediate_vectors",
            "title": "The generate_intermediate_vectors schema.",
            "type": "boolean"
        },
        "use_gdal_trace_outline": {
            "default": true,
            "description": "Generate vector mask using gdal_trace_outline from gina_tools.",
            "id": "use_gdal_trace_outline",
            "title": "The use_gdal_trace_outline schema.",
            "type": "boolean"
        },
        "gdal_trace_outline_min_area": {
            "default": 0,
            "description": "Minimum area to keep in vector mask when using gdal_trace_outline.",
            "id": "gdal_trace_outline_min_area",
            "title": "The gdal_trace_outline_min_area schema.",
            "type": "int"
        },
        "gdal_trace_outline_dp_toler": {
            "default": 0,
            "description": "Tolered pixel approximation in vector mask when using gdal_trace_outline (0 no approximation).",
            "id": "gdal_trace_outline_dp_toler",
            "title": "The gdal_trace_outline_dp_toler schema.",
            "type": "int"
        }
    },
    "type": "object"
},
"inputs": {
    "id": "inputs",
    "properties": {
        "cloud_mask": {
            "description": "Input mask image (in MASK directory for Theia product).",
            "id": "cloud_mask",
            "title": "The Cloud_mask schema.",
            "type": "string"
        },
        "dem": {
            "description": "Input DEM with the same resolution and extent as the input image if preprocessing is deactivated.",
            "id": "dem",
            "title": "The Dem schema.",
            "type": "string"
        },
        "div_mask": {
            "description": "Input other mask image for slope correction flag (optional in MASK directory for Theia product *DIV*.TIF).",
            "id": "div_mask",
            "title": "The div_mask schema.",
            "type": "string"
        },
        "div_slope_thres": {
            "description": "Input threshold of div_mask image to determine the flag for slope correction (optional for Theia product).",
            "id": "div_slope_thres",
            "title": "The Div_slope_thres schema.",
            "type": "int"
        }
    },
    "green_band": {

```

```

        "id": "green_band",
        "properties": {
            "noBand": {
                "default": 1,
                "description": "Green band number.",
                "id": "noBand",
                "title": "The Noband schema.",
                "type": "integer"
            },
            "path": {
                "description": "Path to input L2A image or L2A green band.",
                "id": "path",
                "title": "The Path schema.",
                "type": "string"
            }
        },
        "type": "object"
    },
    "red_band": {
        "id": "red_band",
        "properties": {
            "noBand": {
                "default": 1,
                "description": "Red band number.",
                "id": "noBand",
                "title": "The Noband schema.",
                "type": "integer"
            },
            "path": {
                "description": "Path to input L2A image or L2A red band.",
                "id": "path",
                "title": "The Path schema.",
                "type": "string"
            }
        },
        "type": "object"
    },
    "swir_band": {
        "id": "swir_band",
        "properties": {
            "noBand": {
                "default": 1,
                "description": "SWIR band number.",
                "id": "noBand",
                "title": "The Noband schema.",
                "type": "integer"
            },
            "path": {
                "description": "Path to input L2A image or L2A swir band.",
                "id": "path",
                "title": "The Path schema.",
                "type": "string"
            }
        },
        "type": "object"
    },
    "type": "object"
},
"snow": {
    "id": "snow",
    "properties": {
        "dz": {
            "default": 100,
            "description": "Minimum snow fraction in an elevation band to define zs.",
            "id": "dz",
            "title": "The Dz schema.",
            "type": "integer"
        },
        "fsnow_lim": {
            "default": 0.1,
            "description": "Minimum snow fraction in an elevation band to define zs.",
            "id": "fsnow_lim",
            "title": "The Fsnow_lim schema.",
            "type": "number"
        },
        "fclear_lim": {
            "default": 0.1,
            "description": "Minimum clear pixel fraction in an elevation band to define zs.",
            "id": "fclear_lim",
            "title": "The Fclear_lim schema.",
            "type": "number"
        },
        "fsnow_total_lim": {
            "default": 0.001,
            "description": "Minimum snow fraction in the image to activate the pass 2 snow test.",
            "id": "fsnow_total_lim",
            "title": "The Fsnow_total_lim schema.",
            "type": "number"
        },
        "ndsi_pass1": {
            "default": 0.4,
            "description": "Minimum value of the NDSI for the pass 1 snow test.",
            "id": "ndsi_pass1",
            "title": "The Ndsi_pass1 schema.",
            "type": "number"
        },
        "ndsi_pass2": {
            "default": 0.15,
            "description": "Minimum value of the NDSI for the pass 2 snow test.",
            "id": "ndsi_pass2",
            "title": "The Ndsi_pass2 schema.",
            "type": "number"
        },
        "red_pass1": {
            "default": 200,
            "description": "Minimum value of the red band reflectance the pass 1 snow test.",
            "id": "red_pass1",
            "title": "The Red_pass1 schema.",
            "type": "integer"
        },
        "red_pass2": {
            "default": 40,
            "description": "Minimum value of the red band reflectance the pass 2 snow test.",
            "id": "red_pass2",
            "title": "The Red_pass2 schema."
        }
    }
}

```

```

        "type": "integer"
    },
    "type": "object"
}
,
"type": "object"
}

```

### 3 Validation

The snow maps derived from Landsat data were generally considered as “ground truth” to validate and calibrate lower resolution snow cover products<sup>12</sup>. There is no space-borne sensor with a MIR channel that provides higher resolution imagery than Sentinel-2 or Landsat-8. As a consequence it is difficult to conduct a quantitative assessment. Hence the validation of the algorithm and the adjustment of the parameters was primarily done by visual inspection of the snow cover mask boundaries on the color composite images in Moroccan Atlas, Pyrenees, and the French Alps. First tests were done on subsets of SPOT-4 Take 5 images. The Python/C++ implementation of LIS allowed the processing and inspection of larger datasets such as a series of full 57 Landsat-8 scenes over the Pyrenees available from THEIA.

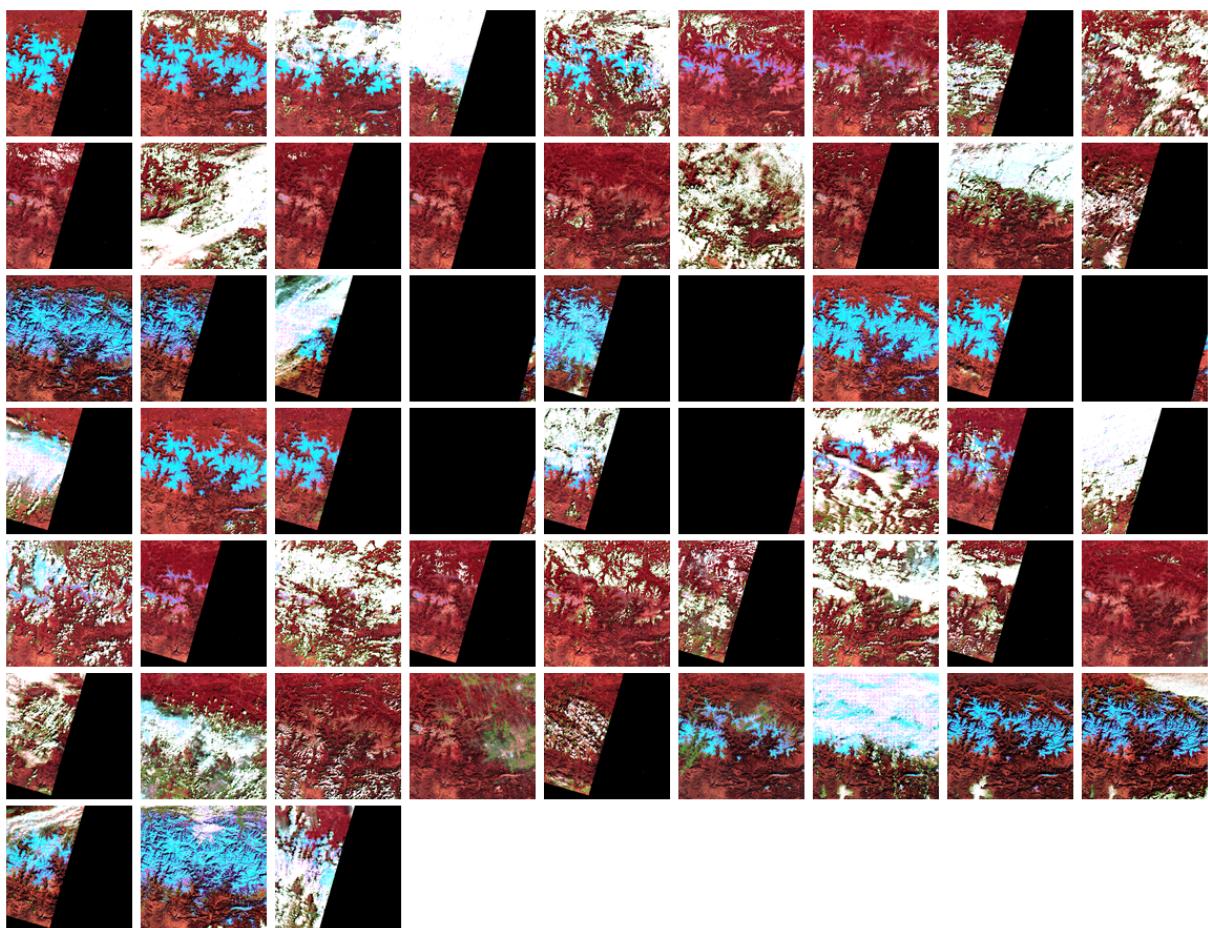


Figure 2: Color composites of a Landsat-8 tile D0005H0001 time series over the Pyrenees processed by LIS. The snow mask is drawn in magenta and cloud mask in green. Each image is 110 km by 110 km.

The implementation of the Sentinel-2 configuration was tested on the Sentinel-2A image of 06-July-2015 tile 30TYN. The output snow mask was compared with an aerial photograph that was taken at a similar period of the year available from the Institut National Information Géographique Forestière. Both images were not acquired in the same year but the snow patterns at the end of the melt season tend to reproduce from one year to the other. The LIS snow mask matches very well the snow cover that is visible on the aerial photograph<sup>13</sup>

<sup>12</sup>Hall, D.K., Riggs, G.A.: Accuracy assessment of the MODIS snow products, *Hydrological Processes* 21(12), 1534–1547, 2007

<sup>13</sup>Gascoin, S. First Sentinel-2 snow map <http://www.cesbio.ups-tlse.fr/multitemp/?p=7014>

The output of LIS was also examined by comparing the output snow mask from two images acquired on the same day by two different sensors. The example shown in Fig. 5 illustrates that both snow masks are consistent, although further inspection revealed that the SPOT-4 snow mask tends to underestimate the snow cover area in this case. This is probably due to the lower radiometric resolution of SPOT-4 sensor.

The output of LIS was also compared to the output of the fmask algorithm<sup>14</sup> available in Google Earth Engine (Landsat TOA with Fmask collections). The snow mask are similar, because the method of detection is also based on the NDSI. We found that fmask falsely detects water area in snow-free shaded slopes (an example is given in Fig 6). The cloud mask is also more conservative than LIS, which is normal given that fmask is a general-purpose algorithm like MACCS.

---

<sup>14</sup>Zhu, Z., Wang, S. and Woodcock, C.E., 2015. Improvement and expansion of the Fmask algorithm: cloud, cloud shadow, and snow detection for Landsats 4–7, 8, and Sentinel 2 images. *Remote Sensing of Environment*, 159, pp.269–277.

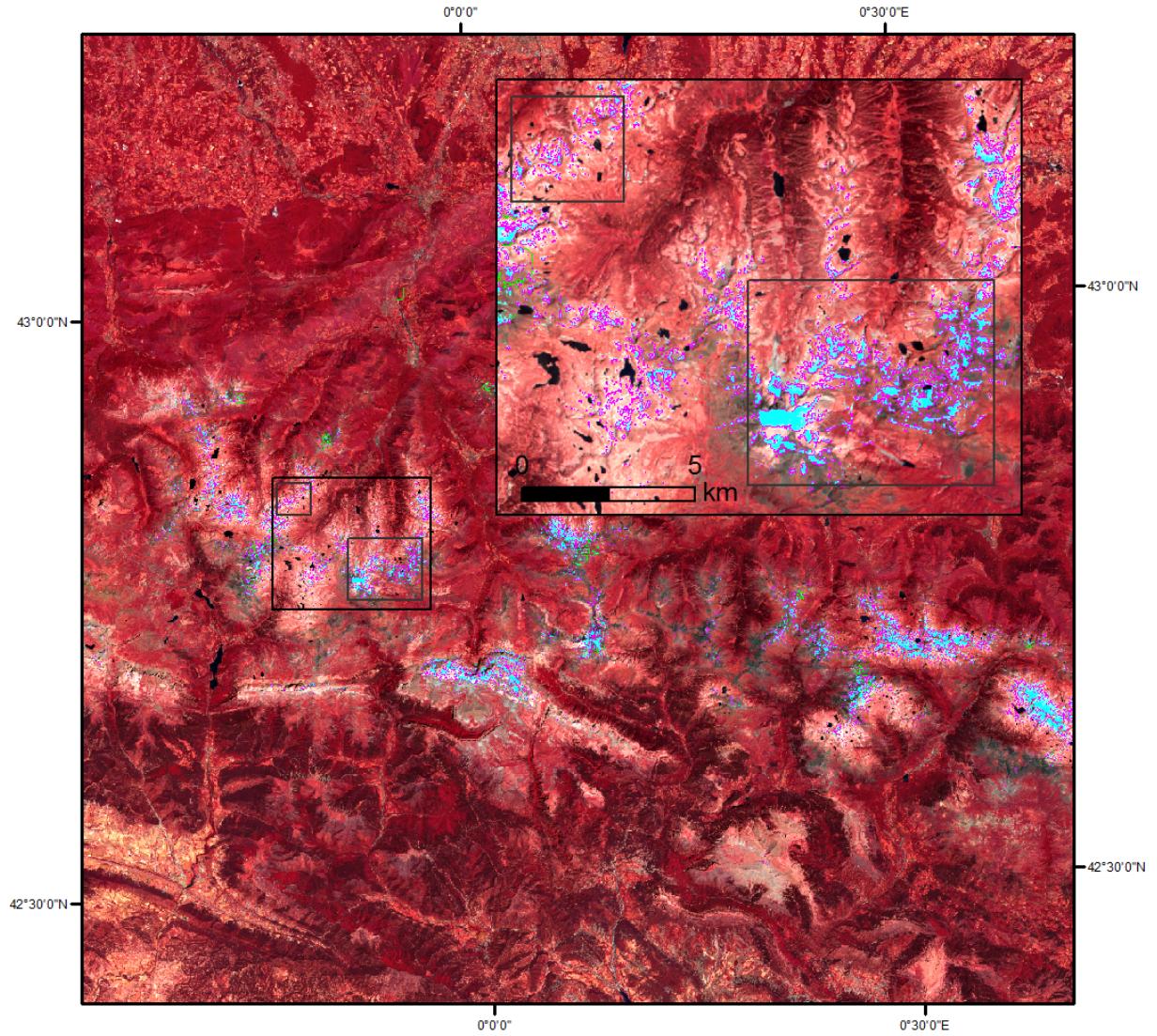


Figure 3: The Sentinel-2A image of 06-July-2015 (level 2A, tile 30TYN) and the snow mask generated by LIS. The snow mask is in magenta and the background image is a color composite RGB NIR/Red/Green. The inset is a zoom in the Vignemale area (Fig. 4).

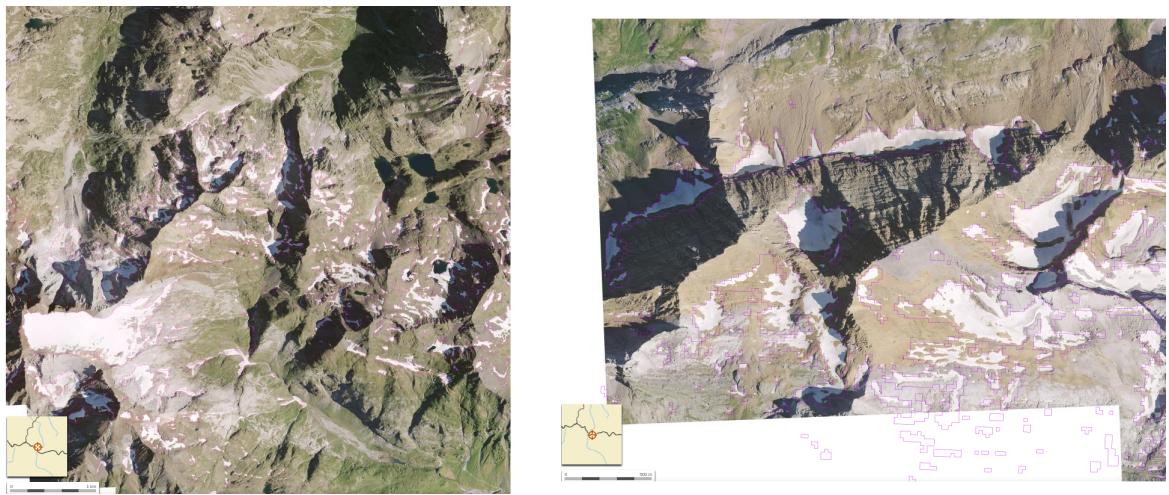


Figure 4: The LIS snow mask from the Sentinel-2A image of 06-July-2015 (Fig. 3) is superposed to an aerial image taken in August 2013 and distributed by the Institut National Information Géographique Forestière.

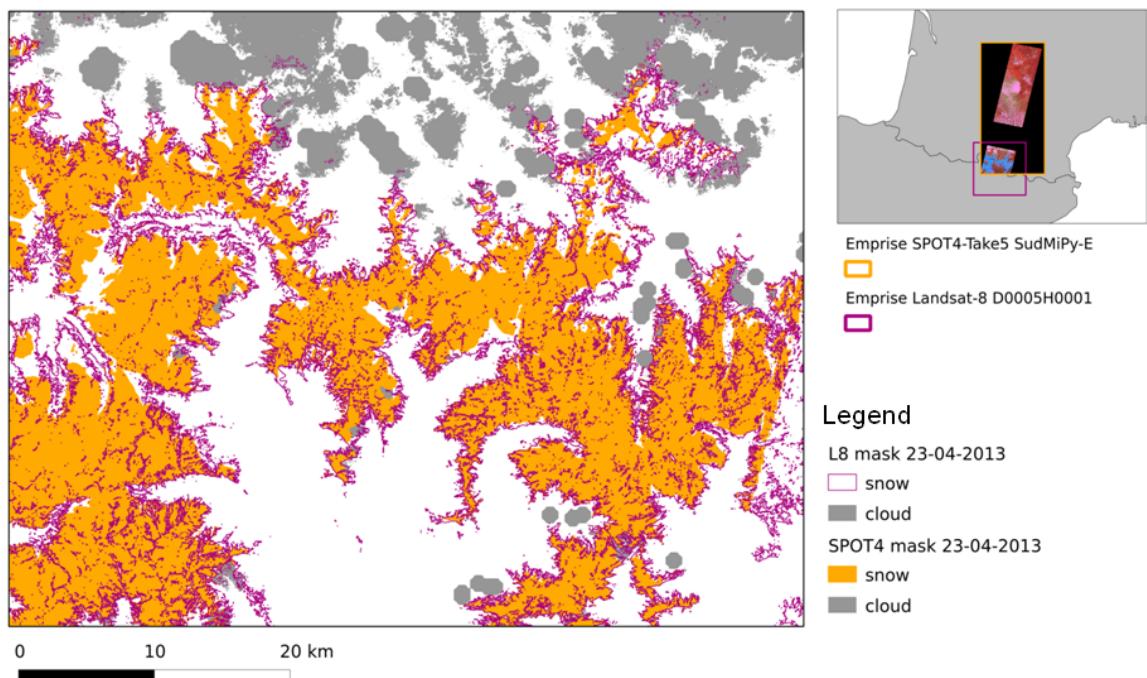


Figure 5: Comparison of the output of LIS from two a Landsat-8 and SPOT-4 product acquired on the same day.

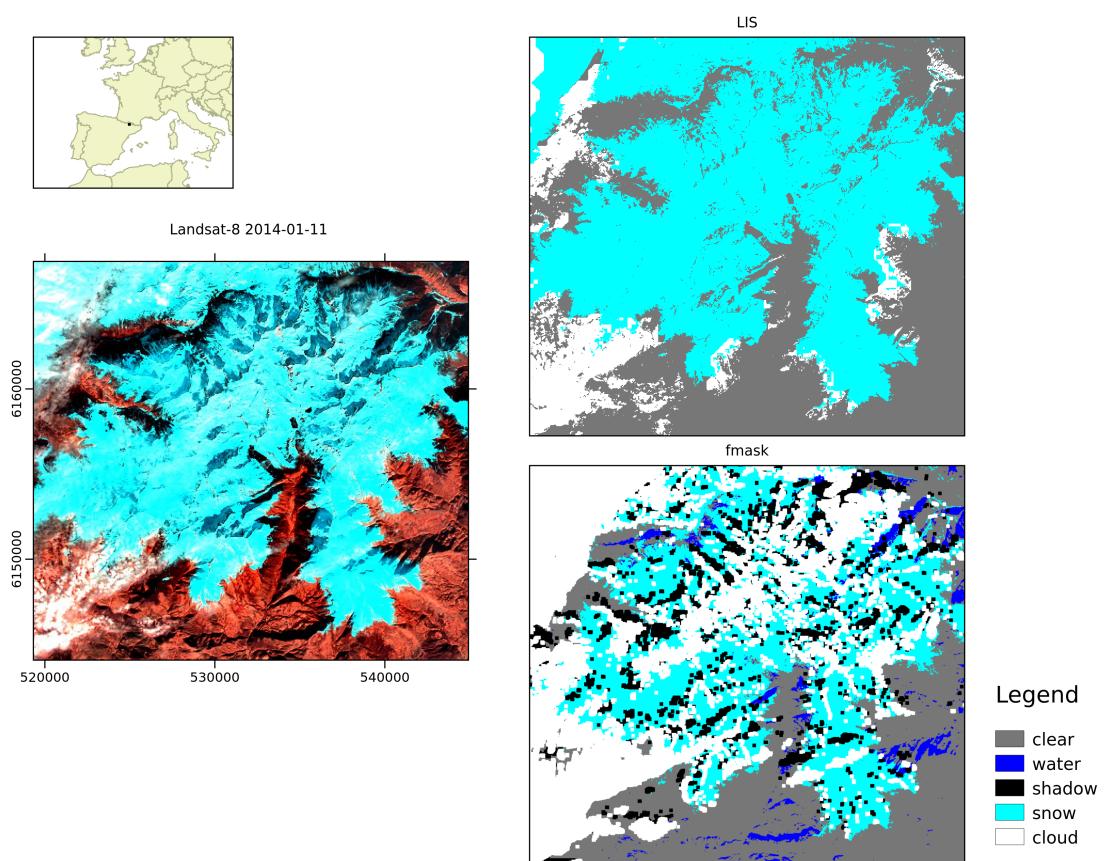


Figure 6: Comparison of the output of the LIS and fmask algorithm on 2014-01-11 in the Pyrenees.

## 4 Conclusion and perspectives

The LIS processing chain is a robust and numerically efficient tool to generate a new, high-resolution snow cover product for Theia Land data center. The LIS snow mask is an improvement from the L2A snow mask (Fig. 7), but its accuracy is largely due to the quality of the slope-corrected L2A product. After the launch of Sentinel-2B the frequency of observations will increase. This will improve the atmospheric corrections by the multi-temporal algorithm MACCS and all the products that are derived from the L2A product in general, including this snow cover extent product.

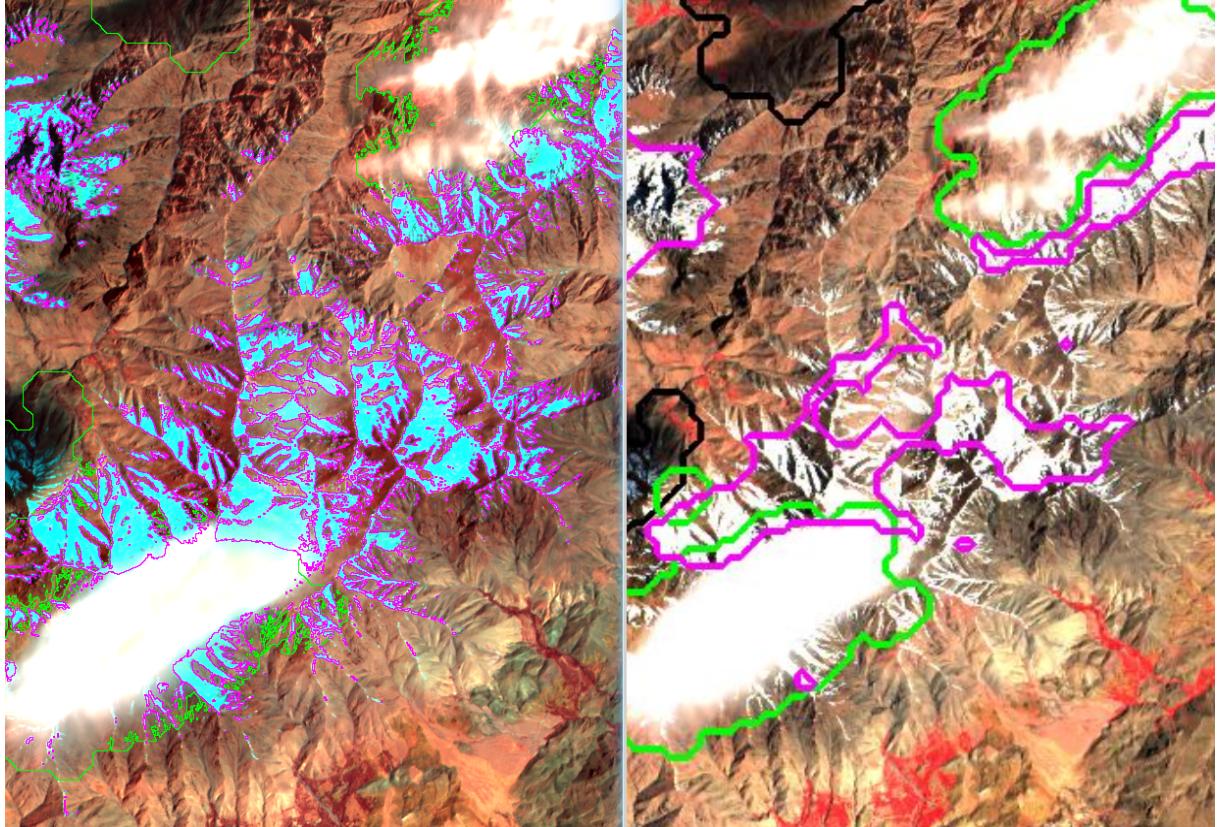


Figure 7: Snow and cloud mask after processing by LIS (left) vs. L2A original cloud and snow mask (right). Clouds are marked in green, cloud shadows in black, snow in magenta. Revisiting the cloud mask enables to increase the area of snow/no snow detection.

In the meantime we would like to further validate LIS using terrestrial time lapse images in the next year. This could also allow a calibration of the parameters (e.g.  $n_2, r_2$ ).

A cloud-free snow cover extent product would facilitate the exploitation of the data by end-users. We plan to work on the development of a cloud-free snow cover product (i.e. a level 3 product). The cloud removal or “gap-filling” algorithm will rely on a series of spatio-temporal filters to reclassify the cloud pixels that are output by LIS. This type of gap-filling algorithm was already developed for MODIS snow cover products<sup>15</sup> but must be further assessed on Sentinel-2 time series. We also plan to evaluate the combination of Sentinel-2 and Landsat-8 snow maps to increase the number of observations used by the gap-filling algorithm.

---

<sup>15</sup>Gascoin, S., Hagolle, O., Huc, M., Jarlan, L., Dejoux, J.-F., Szczypta, C., Marti, R., and Sánchez, R.: A snow cover climatology for the Pyrenees from MODIS snow products, Hydrol. Earth Syst. Sci., 19, 2337-2351.

## A Test script (`castest_CESneige.m`)

Author: Simon Gascoin (CNRS/CESBIO)

June 2015

This code is a demonstrator of the snow detection algorithm for Sentinel-2 images. It calls the function S2snow (appendix B) using subsets of SPOT-4 or Landsat-8 level 2A images.

The input files were generated from L2 images downloaded from Theia Land and pre-processed by three shell scripts:

1. `decompress_* .sh`, to unzip the files
2. `decoupe_* .sh`, to extract a rectangle AOI from L2A data using gdal\_translate with projection window defined in the ascii file `AOI_test_CESNeige.csv`
3. `projette_mnt_* .sh`, to project the SRTM DEM and resample at 30m or 20m (Landsat8 or Take5) over the same AOI. It uses gdalwarp with the cubicspline option

### Contents

- Configuration
- Cloud mask processing parameters
- Snow detection parameters
- Definition of the input images
- Data loading
- Snow detection
- Figures

Check Matlab version

```
matlabversion=version;
assert(str2double(matlabversion(end-5:end-2))>=2014, ...
    'Needs Matlab version 2014 and later')
```

### Configuration

This demo can run for only one image (option 1) or process several images series from different sites and sensors (option 2)

```
demotype=1;
```

If this flag QL is true the code will output three figures for each image

1. a bar graph showing the snow fraction per elevation band,
2. a bar graph showing the snow, cloud and elevation area in m<sup>2</sup>,
3. a false color composite image overlaid by snow and cloud masks.

```
QL=true;
```

Define the output path for the figures

```
pout='../../figures/demo_CESneige';
```

Define a label to add to the figure file names

```
label='ndsipass2_015_cloudpass1_bicubic';
```

## Cloud mask processing parameters

Resampling factor to determine the "dark clouds" in the initial cloud mask. Here rf=8 corresponds to a target resolution of 240m for Landsat.

```
rf=8;
```

Dark cloud threshold (reflectance unit: per mil). The clouds pixels with a (coarse) reflectance lower than rRed\_darkcloud are selected to pass the snow test

```
rRed_darkcloud=500;
```

Back to cloud threshold. Pixels which are not detected as snow then they go back to the cloud mask only if their reflectance at full resolution is greater than rRed\_backtocaloud. Here we use full resolution reflectances because resampled cloud pixels reflectance drops rapidly along the cloud edges due to the mixing with the land reflectance.

```
rRed_backtocaloud=100;
```

## Snow detection parameters

Elevation band height in m.

```
dz=100;
```

NDSI threshold for pass 1

```
ndsi_pass1=0.4;
```

Threshold in the red reflectance for pass 1

```
rRed_pass1=200;
```

NDSI threshold for pass 2

```
ndsi_pass2=0.15; % Note: Zhu's fmask algorithm uses 0.15
```

Threshold in the red reflectance for pass 2

```
rRed_pass2=120;
```

Minimum snow fraction from which to compute the snowline elevation (zs)

```
fsnow_lim=0.1;
```

Snow fraction limit to go to pass 2. The total snow fraction after pass 1 is computed to decide whether or not proceed to pass 2. If the snow fraction is below fsnow\_total\_lim then pass 2 is skipped because the number of snow pixels is not sufficient to properly determine the snow line elevation. Here we use 1000 pixels.

```
fsnow_total_lim=0.001; % Warning! it depends on the image size
```

## Definition of the input images

```
switch demotype
    case 1

        use only 1 image

        satlist={'Take5'};
        sitelist{1}={'Maroc'};
        datelist{1}{1}={'20130327'}; % format yyyyymmdd

    case 2
```

process a batch of 2 x Landsat8 and 4 x Take5 images series

```
[satlist,sitelist,datelist]=load_demo_input('all');

otherwise
    error('demotype should be 1 or 2')
end
```

## Data loading

Begin the sensor loop

```
nsat=length(satlist);
for isat=1:nsat

    sat=satlist{isat};

    nsite=length(sitelist{isat});
```

Get the sensor band numbers and no-data value

```
switch sat
    case 'Take5'
        nGreen=1; % Index of green band
        nMIR=4; % Index of SWIR band (1 to 3 μm) = band 11 (1.6 μm) in S2
        nRed=2; % Index of red band
        nodata=-10000; % no-data value
    case 'Landsat8'
        nGreen=3;
        nMIR=6;
        nRed=4;
        nodata=-10000;
    otherwise
        error('sat should be Take5 or Landsat8')
end
```

Begin the site loop

```
for isite=1:nsite;

    site=sitelist{isat}{isite};
    ndate=length(datelist{isat}{isite});
```

read SRTM DEM for this site

```
fdem=['../' sat '/AOI_test_CESNeige/SRTM/' site '/' site '.tif'];
[Zdem,R]=geotiffread(fdem);
```

Begin the date loop

```
for idate=1:ndate
    date=datalist{isat}{isite}{idate};
```

Read the L2 reflectance data (search the image filename based on the date)

```
pL2=['../' sat '/AOI_test_CESNeige/LEVEL2A/' site]; % path
fL2=dir([pL2 '/* date '*PENTE*.TIF']); % file list
assert(~isempty(fL2),'The L2 image is not present in the input directory')
f=[pL2 '/ fL2.name];
[ZL2,~]=geotiffread(f);
```

The data are converted to double because it enables to work with NaN, but we could probably avoid this to optimize memory use..

```
ZL2=double(ZL2);
ZL2(ZL2==nodata)=NaN;
```

Read the cloud mask

```
fc=dir([pL2 '/* date '*NUA.TIF']);
assert(~isempty(fc),'The cloud mask is not present in the input directory')
f=[pL2 '/ fc.name];
[Zc,~]=geotiffread(f);
```

## Snow detection

Calls the function S2snow (appendix [B](#))

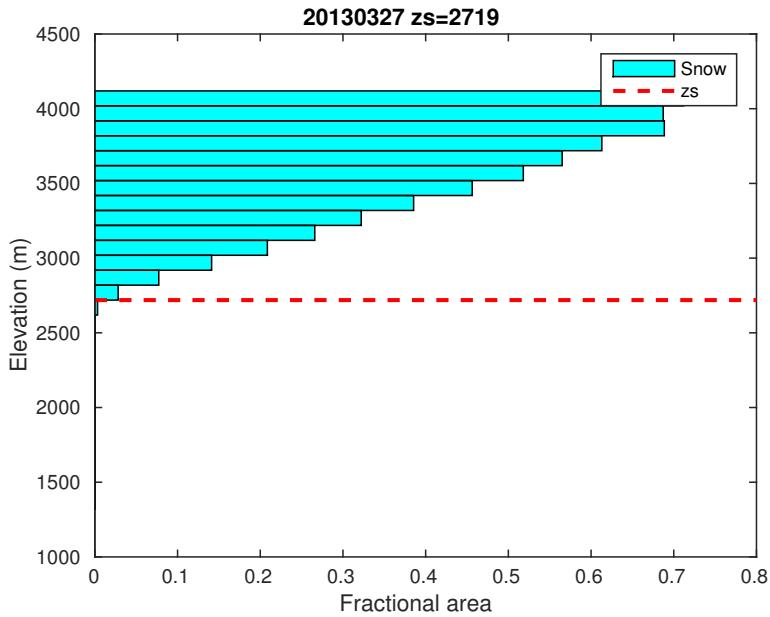
```
[snowtest,cloudtestfinal,z_center,B1,B2,fsnow,zs,%
    fsnow_z,fcloud_z,N,z_edges]...
    = S2snow...
    ZL2,Zc,Zdem,nGreen,nRed,nMIR,rf,QL,rRed_darkcloud,%
    ndsi_pass1,ndsi_pass2,rRed_pass1,rRed_pass2,dz,fsnow_lim,%
    fsnow_total_lim,rRed_backtocaloud);
```

## Figures

```
if QL
```

Draw a bar plot of the snow fraction per elevation band

```
figure(1),clf
barh(z_center,fsnow_z,1,'facecolor','c');
hold on
if ~isempty(zs)
    hrf=refline(0,zs);
    set(hrf,'Color','r','LineStyle','--','LineWidth',2)
end
xlabel('Fractional area')
ylabel('Elevation (m)')
legend('Snow','zs')
title(sprintf('%s zs=%g',date,zs))
```

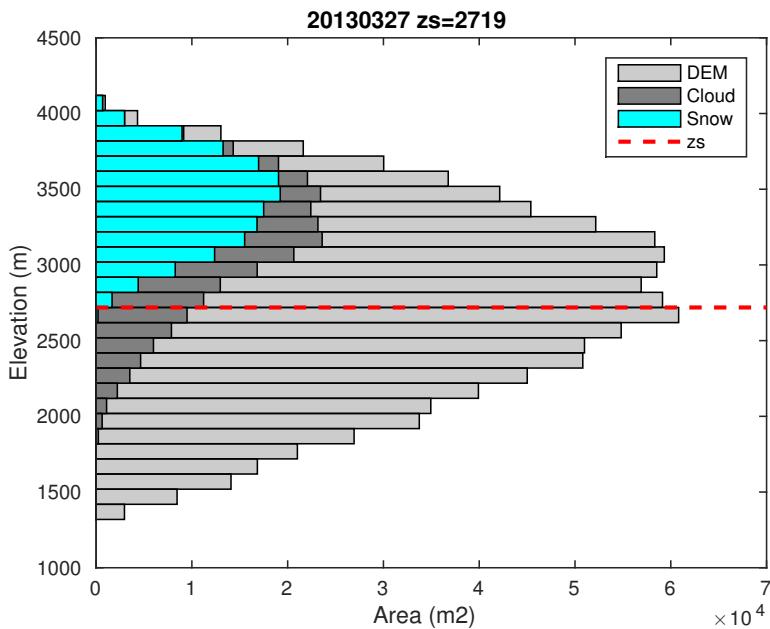


Draw a bar plot of the snow area and elevation band area

```

figure(2),clf
[Nz,~,~] = histcounts(Zdem(:,z_edges));
barh(z_center,Nz,1,'facecolor',.8*[1 1 1]);
hold on
barh(z_center,(fcloud_z+fsnow_z).*Nz',1,'facecolor',.5*[1 1 1]);
barh(z_center,fsnow_z.*Nz',1,'facecolor','c');
if ~isempty(zs)
    hrf=refline(0,zs);
    set(hrf,'Color','r','LineStyle','--','LineWidth',2)
end
xlabel('Area (m2)')
ylabel('Elevation (m)')
legend('DEM','Cloud','Snow','zs')
title(sprintf('%s zs=%g',date,zs))

```

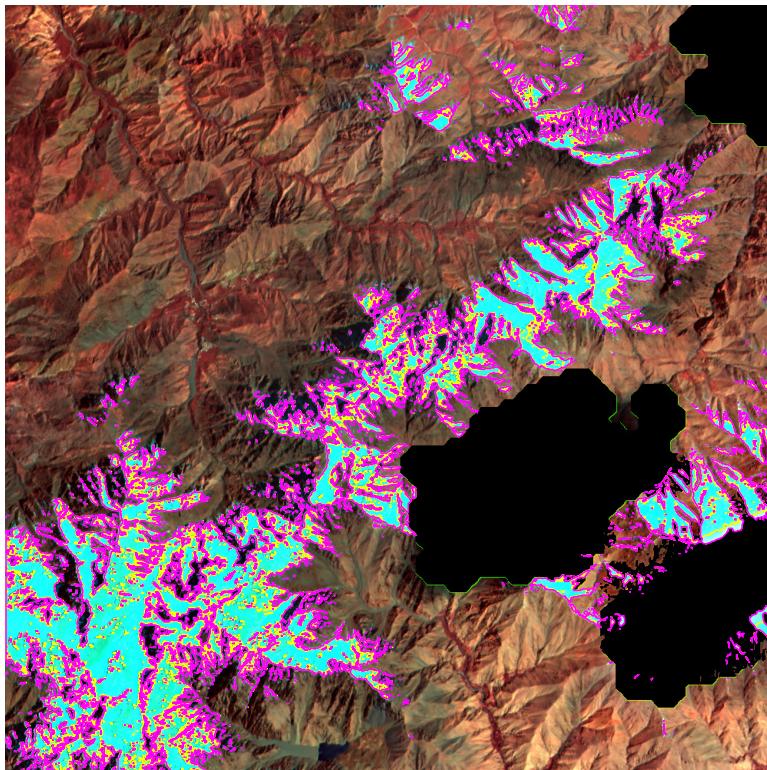


Show an RGB composition overlaid with pass 1 and 2 snow cover polygons. The clouds pixels are in black.

```

figure(3),clf
z=ZL2(:,:, [nMIR nRed nGreen]);
max_b=[300 300 300];
z2=zeros(size(z), 'uint8');
for b=1:3
    z2(:,: ,b)=~cloudtestfinal.*double(z(:,: ,b))/max_b(b)*255;
end
imshow(uint8(z2), 'Border', 'tight')
hold on
for k = 1:length(B1)
    boundary = B1{k};
    %           if length(boundary)>100
    plot(boundary(:,2), boundary(:,1), 'y', 'LineWidth', 1)
    %
end
if ~isempty(zs)
    for k = 1:length(B2)
        boundary = B2{k};
        %           if length(boundary)>100
        plot(boundary(:,2), boundary(:,1), 'm', 'LineWidth', 1)
        %
    end
end

```



Print figures

```

set(1:3,'PaperPositionMode','auto','Visible','off');
pfig=[pout '/ sat '/ site];
system(sprintf('mkdir -p %s/zs1',pfig));
system(sprintf('mkdir -p %s/zs2',pfig));
system(sprintf('mkdir -p %s/QL',pfig));
f1=sprintf('%s/zs1/%s_%s_%s_%s.png',pfig,sat,site,date,label);
f2=sprintf('%s/zs2/%s_%s_%s_%s.png',pfig,sat,site,date,label);
f3=sprintf('%s/QL/%s_%s_%s_%s.png',pfig,sat,site,date,label);

```

```

    print(1,f1,'-dpng')
    print(2,f2,'-dpng')
    print(3,f3,'-dpng')

end

end

end

end

```

## B Snow detection function (S2snow.m)

**Author:** Simon Gascoin (CNRS/CESBIO)

**June 2015**

This function computes the snow presence and absence (snow/no-snow) from a high-resolution optical multispectral image like Sentinel-2 (eg Landsat-8, SPOT-4). The input are the full resolution reflectances corrected from atmospheric and slope effects (Level 2A), the associated cloud mask and a digital elevation model. The output is a mask with three classes: snow/no-snow/cloud. The output cloud mask is different from the input cloud mask.

The snow detection algorithm works in two passes: first the most evident snow is detected using a set of conservative thresholds, then this snow pixels are used to determine the lower elevation of the snow cover. A second pass is performed for the pixels above this elevation with a new set of less conservative thresholds

The snow detection is made on a pixel-basis using the Normalized Difference Snow Index (NDSI):

$$NDSI = \frac{\rho_{Green} - \rho_{MIR}}{\rho_{Green} + \rho_{MIR}}$$

where  $\rho_{Green}$  is the reflectance in the green channel and  $\rho_{MIR}$  in the shortwave infrared.

## Contents

- Function call
- Inputs
- Outputs
- Initial cloud mask processing
- Pass 1 : first snow test
- Pass 2 : second snow test
- Final update of the cloud mask

## Function call

```

function [snowtest,cloudtestfinal,z_center,B1,B2,fsnow,zs, ...
    fsnow_z,fcloud_z,N,z_edges]...
= S2snow(... ...
ZL2,Zc,Zdem,nGreen,nRed,nMIR,rf,QL,rRed_darkcloud, ...
ndsi_pass1,ndsi_pass2,rRed_pass1,rRed_pass2,dz,fsnow_lim, ...
fsnow_total_lim,rRed_backtocaloud)

```

## Inputs

- ZL2: L2A reflectances (NxMxB double array, where B is the number of band)
- Zc: cloud mask ( $Zc > 0$  is cloud) (NxM array)
- Zdem: digital elevation model (NxM array)
- nGreen, nRed, nMIR: index of green, red, SWIR band in ZL2
- rf: resampling factor for cloud reflectance
- QL: quicklooks? (boolean)
- ndsi\_pass1,ndsi\_pass2,rRed\_pass1,rRed\_pass2,rRed\_backtocalc: parameters for reflectance-based tests (see the documentation in appendix A)
- dz: elevation band width
- fsnow\_lim: minimum snow fraction in an elevation band to define zs
- fsnow\_total\_lim: minimum snow fraction in the whole image after pass1 to go to pass2

## Outputs

- snowtest: snow presence/absence (NxM boolean)
- cloudtestfinal: cloud presence/absence (NxM boolean)
- z\_center: centers of the elevation bands (vector)
- B1, B2: snow cover contours after pass 1 and pass 2 (cell)
- fsnow: total snow fraction in the image after pass 1
- zs: snow elevation for pass 2
- fsnow\_z: snow fraction in each band
- N: number of clear pixels in each elevation band
- z\_edges: upper and lower limits of the elevation bands

## Initial cloud mask processing

The L2A cloud mask is too conservative and much useful information for snow cover mapping is lost.

We allow the reclassification of some cloud pixels in snow or no-snow only if they have a rather low reflectance. We select only these "dark clouds" because the NDSI test is robust to the snow/cloud confusion in this case

Initial cloud mask (incl. cloud shadows)

```
cloudtestinit = Zc>0;  
  
Get the cloud shadows  
  
code_shad = bin2dec(fliplr('00000011'));  
cloudshadow = bitand(Zc,code_shad)>0;
```

We exclude high clouds detected from with 1.38  $\mu\text{m}$  band (S2 and L8 only)

```
code_highcloud = bin2dec(fliplr('00000100'));  
highcloud = bitand(Zc,code_highcloud)>0;
```

The "dark cloud" are found with a threshold rRed\_darkcloud in the red band after bilinear resampling of the original image to match the MACCS algorithm philosophy but the benefit of this was not yet evaluated.

```
rRedcoarse = imresize(ZL2(:,:,nRed), 1/rf,'bicubic');
```

Then we oversample to the initial resolution using nearest neighbour to have the same image size.

```
rRedcoarse_oversamp1 = imresize(rRedcoarse,rf,'nearest');
```

clear rRedcoarse to free up memory

```
clear rRedcoarse
```

These pixels are removed from the intial cloud mask unless they were flagged as a cloud shadow or high cloud. The snow detection will not be applied where cloudtesttmp is true.

```
cloudtesttmp = ...
    (cloudtestinit & rRedcoarse_oversampl>rRed_darkcloud)...
    | cloudshadow | highcloud;
```

## Pass 1 : first snow test

The test is based on the Normalized Difference Snow Index (ndsi0) and the reflectance value in the red channel

```
ndsi0 = (ZL2(:,:,nGreen)-ZL2(:,:,nMIR))./(ZL2(:,:,nGreen)+ZL2(:,:,nMIR));
```

A pixel is marked as snow covered if NDSI is higher than ndsi\_pass1 and if the red reflectance is higher than Red\_pass1

```
snowtest = ~cloudtesttmp & ndsi0>ndsi_pass1 & ZL2(:,:,nRed)>rRed_pass1;
```

Now we can update the cloud mask Some pixels were originally marked as cloud but were not reclassified as snow after pass 1. These pixels are marked as cloud if they have a high reflectance ("back to black"). Otherwise we keep them as "no-snow" ("stayin alive").

```
cloudtestpass1 = cloudtesttmp ...
    | (~snowtest & cloudtestinit & ZL2(:,:,nRed)>rRed_backtocaloud);
```

For the quicklooks we compute the boundary of the snow cover after pass 1.

*Warning this uses much computer time.*

```
if QL
    [B1, ~] = bwboundaries(snowtest); % requires Matlab Image Processing Toolbox
else
    B1 = [];
end
```

Initialize some output variables as empty array in case they is not reached in pass 2

```
B2 = [];
zs = [];
N = [];
bin = [];
```

## Pass 2 : second snow test

Based on the DEM, the scene is discretized in elevation band of height dz. The elevation bands start from the minimal elevation found in the DEM resampled at the image resolution. The edges of elevation band are :

```
z_edges = double(min(Zdem(:)):dz:max(Zdem(:)));
% NB) the colon operator j:i:k is the same as [j,j+i,j+2i, ...,j+m*i],
% where m = fix((k-j)/i).
```

The number of bins is :

```
nbins = length(z_edges)-1;
```

The mean elevation of each band is computed for the graphics but not used by the algorithm

```
z_center = mean([z_edges(2:end);z_edges(1:end-1)]);
```

The lower edge of each bin is used to define zs

```
z_loweredges=z_edges(1:end-1);
```

We get the number of pixels (N) which are cloud-free (at this step) in each bin, and the index array (bin) to identify the elevation band corresponding to a pixel in the cloud-free portion of the image

```
if nbins>0
    [N,~,bin] = histcounts(Zdem(~cloudtestpass1),z_edges);
end
```

Compute the fraction of snow pixels in each elevation band

```
fsnow_z = zeros(nbins,1);
fcloud_z = zeros(nbins,1);
if ~isempty(bin)
```

We collect the snow pixels only in the cloud free areas

```
M = snowtest(~cloudtestpass1(:)); % this also reshapes snowtest from a
% 2D array to 1D array to match the bin index array dimension
```

Then we start to loop over the each elevation band

```
for i = 1:nbins
```

We sum the snow pixels and divide by the number of cloud-free pixels

```
if N(i)>0
    fsnow_z(i) = sum(M(bin==i))/N(i);
    fcloud_z(i) = sum(cloudtestpass1(bin==i))/N(i);
else
    fsnow_z(i) = NaN;
    fcloud_z(i) = NaN;
end

end
```

We compute the total fraction of snow pixels in the image

```
fsnow = nnz(snowtest)/numel(snowtest);
```

The pass 2 snow test is not performed if there is not enough snow detected in pass 1.

```
if fsnow>fsnow_total_lim
```

We get zs, the minimum snow elevation above which we apply pass 2. zs is **two elevation bands** below the band at which fsnow > fsnow\_lim

```
izs = find(fsnow_z>fsnow_lim,1,'first');
zs = z_loweredges(max(izs-2,1));
```

```
    end
```

```
end
```

if zs was found then we apply the second snow test : A pixel is marked as snow covered if NDSI is higher than ndsi\_pass2 and if the red reflectance is higher than Red\_pass1... and if it is above zs!

```
if ~isempty(zs)

snowtest2 = ~cloudtesttmp ... % we use cloudtesttmp again
& ndsi0>ndsi_pass2 ...
& Zdem>zs ...
& ZL2(:,:,nRed)>rRed_pass2;
```

We add these snow pixels in the snow mask from pass 1

```
snowtest = snowtest2 | snowtest;
```

For the quicklooks we compute the boundary of the snow cover after pass 2 *Warning this uses much computer time.*

```
if QL
    [B2, ~] = bwboundaries(snowtest);
end
```

```
end
```

## Final update of the cloud mask

Some pixels were originally marked as cloud but were not reclassified as snow after pass 1. These pixels are marked as cloud if they have a high reflectance ("back to black"). Otherwise we keep them as "no-snow" ("stayin alive").

```
cloudtestfinal = cloudtesttmp ...
| (~snowtest & cloudtestinit & ZL2(:,:,nRed)>rRed_backtocaloud);
```