Software of the seasonal crop type mapping prototype

Centre d'Etudes Spatiales de la BIOsphere (CESBIO),
UMR 5127 - CNES - CNRS - IRD - UPS,
18 avenue Edouard Belin, 31401 Toulouse, France.

Part of Deliverables	**5.4**
Version	1
Author	CESBIO, Toulouse

Contents

1 Introduction

The SenSagri Work Package 5 aims to develop robust automatic algorithms to detect crop areas. This documents is the user manual of the prototype chain developed by the CESBIO that performs the required task. The chain needs two different type of input data:

- Ensemble of images time series acquired by Sentinel-1 and Sentinel-2 satellites that covers the zone of interest. Each images time serie is in tiff format and corresponds to a *tiles* which is a region of 110km by 110km acquires by the Sentinel satellites.
- Ensemble of in situ land cover measurements files that overlaps the images time series zone. Those measurements are contained in vector shapefiles in ESRI format. One such a shapefile per image tiles is required. Each shapefile needs to be geolocalised according to the same coordinates system that its corresponding images time serie.

By using satellite and reference data, the chain applies a serie of treatments, especially a supervised classification. Different results are then provided as output data:

- The crop mask map and its confidence map (images files in tiff format).
- The seasonal crop type map and its confidence map (images files in tiff format).

More details concerning the methodology behind the chain can be found in Deliverable 5.3.

2 Installation

2.1 Minimal Configuration

The chain is meant to be used on a linux server that allows multi processor parallelization and important amount of memory. The chain can also be used on a personal computer with a modest configuration. The user should keep in mind that in that case computation time might increase strongly and memory overflow might occur. The use on a personal computer should be reserved for specific tests and the deployment preparation on a more scalable server. Note that the chain does not need specific graphic cards or GPU configuration. A summary of the needed resources is listed on the Table 1.

Ressources	Minimal Configuration	Advised Configuration
RAM	8GB	64GB
Disk	500GB	2TB
CPU	Dual Core 64	10-core

Table 1: Minimal and advised resources required by the classification chain.

2.2 Dependencies

The chain needs the following dependancies that are commonly installed on linux systems designs for scientific calculations:

- gcc 4.8.5
- cmake 3.9.1
- python 2.7.5 with numpy, scypyi, matplotlib and gdal libraries.
- mpirun 3.0.4
- pdflatex

Slighlty lower versions of these programs might still work but it has not been tested.

2.3 Compilation and installation of the prototype

It is assumed that the installation will be managed by the user called "user" on a linux machine called "machine" with the previous installed tools and libraries. The user does not need to have root privileges. The main installation steps are detailed in the following:

1. Download the chain archive SenSAgriChain.tar.gz in the user home directory[1]

2. Uncompress the archive with the command:

```
[user@machine]$ tar zxvf SenSAgriChain.tar.gz
```

   It will create a new directory called SenSAgriChain.

3. Go at SenSAgriChain/bin/, change the rights of the installation script then execute it:

```
[user@machine]$ cd SenSAgriChain/bin/
[user@machine]$ chmod 744 install.sh
[user@machine]$ ./install.sh
```

   The installation will take several minutes depending of the machine configuration. It is almost completed when the following message appears:

```
*** Installation Almost Done ***
To terminate the installation process,
please copy the following lines at the end
of the user .bashrc file then source it:
```

4. Follow the last instruction by opening the .bashrc file with a text editor, then copy/paste at the end of it the following 7 lines:

```
export SENSAGRICHAIN_HOME=~/SenSAgriChain
export SSOTB_HOME=$SENSAGRICHAIN_HOME/bin/OTB-6.2.0-Linux64
export PATH=${SSOTB_HOME}/bin:$PATH
export PATH=${SENSAGRICHAIN_HOME}/bin/chdb-master:$PATH
export PATH=${SENSAGRICHAIN_HOME}/bin/cpp/Executables:$PATH
export PATH=${SENSAGRICHAIN_HOME}/bin/scripts:$PATH
export LD_LIBRARY_PATH=${SSOTB_HOME}/lib:$LD_LIBRARY_PATH
```

   then save the change and quit the text editor.

5. The .bashrc file needs to be reloaded to take the change into account. It is done with the command:

```
[user@machine]$ cd
[user@machine]$ source .bashrc
```

6. To test if the chain is working, simply type the command sensagrichain. Depending of the version of the prototype, a message similar to the following one should appear:

```
[user@machine]$ sensagrichain

##########################################
#       SenSAgri Classification Chain    #
#       Last version: 23/04/2018         #
#             CESBIO 2017-2018           #
##########################################

usage: sensagrichain [-h] [-w WORKDIR] config
sensagrichain: error: too few arguments
```

   If the message appears, installation is completed.

[1] The chain can be found on the CNR repository res.ba.issia.cnr.it at the location /IncommingQueue/ups/Deliverables/WP5/SenSAgriChain.tar.gz

3 Usage

3.1 Working Directory

The first important and compulsory step before running the classification chain is to create the so-called "working directory". It is the directory that will contain the input data, the output products as well as configuration files. This working directory need to follow a specific tree structure. That is why it should be created automatically by running the sensagrichain command with the –workdir switch on. As an example, one can imagine the user wants to create a working directory call "Example-France2016" in the home directory of the chain: To do so, the user can run the following commands:

```
[user@machine]$ cd ~/SenSAgriChain
[user@machine]$ sensagrichain -w Example-France2016
```

The user can then go in the newly created directory and have a look at the tree structure thanks to the `ls *` command. The output should show:

```
[user@machine]$ cd Example-France2016
[user@machine]$ ls *
joborder.cfg

WorkFiles:
DatesFiles  Images  Shapefiles
```

Here some details concerning the file and directories that are created automatically inside the working directory:

- **joborder.cfg:** Template of the configuration file that allows the whole setting of the chain. More details will be exposed in the Section 3.3.
- **WorkFiles/DatesFiles/:** Directory that should contain the text files that list the chosen dates of the Sentinel-1 and Sentinel-2 images time series that are going to be used for the classification. Two such files should be provided by the user: one corresponding to the S1 time series, one corresponding to the S2 time series. The format of these text files is straintforward: there is one date per line and it is written as a YYYMMDD number. For instance, for 4 dates between January the 1st and January the 16th 2016 with a time step of 5 days, such a date file has the shape:

  ```
  20160101
  20160106
  20160111
  20160116
  ```

- **WorkFiles/Images/:** Directory that should contain the Sentinel-1 and Sentinel-2 image satellites data. Each images file should be in a tiff format where each pixel values is a multdimenssionnal vector that contain a concatenation of all bands at all acquisition dates (see more details in Deliverables 5.1 and 5.3).
- **WorkFiles/Shapefiles/:** Directory that should contains the shapefiles of the field data. There should be one shapefile per images tile, in ESRI format and set on the same coordinates system that the corresponding images. The data assigns to each polygons in the shapefiles should follow the column structure of the form:

LC	CODE	CROP
Maize	12	1
Soybean	41	1
Build Up	7001	0
Forest	5001	0

  where the columns type are:

- **LC**: Character string that indicates the polygon land cover class in English.
- **CODE**: 16-bit integer that corresponds to the land cover class code according to the extended JECAM nomenclature[2].
- **CROP**: 16-bit integer that indicates if the polygon contains an annual crop (1 if is a crop, 0 otherwise).

Note that as soon as the previous columns are included in the shapefile describing the reference data, other columns can be added to the file without any problem, assuming of course they do not have the reserved names "LC", "CODE" and "CROP".

3.2 Preparation of the input data

Once the working directory is created, the next step is to fill the relevant directories with the relevant files. As an exemple, the user can for instance copy two dates files in the **DatesFiles**, four images in the **Images** directory and one shapefile (with its auxiliary files) in the **Shapefiles** directory. Then its files tree might look like:

```
[user@machine]$ cd ~/SenSAgriChain/Example-France2016/WorkFiles/
[user@machine]$ ls *
DatesFiles:
S1_Dates.txt  S2_Dates.txt

Images:
S1_T30TYP.tif  S1_T31TCJ.tif  S2_T30TYP.tif  S2_T31TCJ.tif

Shapefiles:
Ref2016_T30TYP.dbf  Ref2106_T30TYP.prj  Ref2016_T30TYP.qpj
Ref2016_T30TYP.shp  Ref2016_T30TYP.shx  Ref2016_T31TCJ.dbf
Ref2016_T31TCJ.prj  Ref2016_T31TCJ.qpj  Ref2016_T31TCJ.shp
Ref2106_T31TCJ.shx
```

Note that for conveniency, it is perfectly possible to use symbolic links that point to a remote location. To create such a symbolic link, the following command can be used:

```
[user@machine]$ ln -s OtherLocation/S1_Images.tif ~/SenSAgriChain/test
   /WorkFiles/Images/
```

3.3 Configuration file

Once the input data are included to the WorkFiles directory and subdirectories, the next step is to edit the joborder configuration file. This file contains the different settings needed by the chain to work properly. A template version of this file is created automatically in the working directory under the default name joborder.cfg. Because this file is a template, it is already felt with some default values that should be changed according to the chain specific usage. Its name can also be changed if needed. Note that if several configurations of the chain need to be used, several such configuration files can be created in the working directory.

3.3.1 [Parameters]

The first section of this file is the **[Parameters]** section. All the items of this section are global parameters the chain need to work properly. They are *a priori* fixed one time for good. All those item are listed in the Table 2 with their corresponding details and examples:

[2] http://www.jecam.org/JECAM_Guidelines_for_Field_Data_Collection_v1_0.pdf

Item	Type	Comments and Examples
JobName:	Character string	When the chain is used with a queue manager such as PBS or SLURM, JobName is the name of the job sent to the queue. This is also the name given to the output script produce by the sensagrichain command (see Section 3.4). **Ex:** "France2016".
ReservedTime:	Character string	Time reserved for the job when a queue manager such as PBS or SLURM is used. Without this kind of managers, this paramater will be ignore and can just be fixed to a default value such as "0-00:00:00". **Ex:** For a run time of 12h, the value should be "0-12:00:00".
WorkingDirectory:	Character string	Indicate the absolute path of the working directory where the input and output data are managed. This path is automatically felt at the creation of the working directory. **Ex:** "/home/user/SenSAgriChain/Example-France2016".
RadarDates:	Character string	Relative name of the date file containing S1 acquisition dates that is stored in the (workdir)/WorkFiles/DatesFiles directory. Each vector entry is in correspondance with the selected tiles provided by the Tiles items. **Ex:** "S1_Dates.txt"
OpticalDates:	Character string	Relative name of the date file containing S2 acquisition dates that is stored in the (workdir)/WorkFiles/DatesFiles directory.Each vector entry is in correspondance with the selected tiles provided by the Tiles items. **Ex:** "S2_Dates.txt"
NbRun:	Integer	Number of statistical runs. **Ex:** 10.
NbSamples:	Integer	Number of pixels par classes used during the supervised training. **Ex:** 2000 (default value).
RegularizationRadius:	Integer	Radius in pixels of the image spatial regularization of the output maps if such regularization is used. **Ex:** 2 (default value)
Tiles:	Vector of character strings	List of tiles that will be treated by the chain. To each tile corresponds a S1 image time serie, a S2 images time serie and a shapefile. **Ex:** ["T30TYP","T31TCJ"]
RadarImages:	Vector of character strings	Relative names of the S1 radar images time series that are stored in the (workdir)/WorkFiles/Images directory. Each vector entry is in correspondance with the selected tiles provided by the Tiles items. **Ex:** ["S1_T30TYP.tif","S1_T31TCJ.tif"]
OpticalImages:	Vector of character strings	Relative names of the S2 optical images time series that are stored in the (workdir)/WorkFiles/Images directory. Each vector entry is in correspondance with the selected tiles provided by the Tiles items. **Ex:** ["S2_T30TYP.tif","S2_T31TCJ.tif"]
ProdRadarDates	Vector of integer	List of radar dates that are used to construct the products maps. Each dates is expressed as an integer number (between 1 and the maximum number of radar dates) that corresponds to the position of the dates listed in the *RadarDates* files. Typically, 3 or 4 such dates are chosen during the year. **Ex:** [2,50,80,121]
ProdOpticalDates	Vector of integer	List of optical dates that are used to construct the products maps. Each dates is expressed as an integer number (between 1 and the number of optical dates) that corresponds to the position of the dates listed in the *OpticalDates* files. Typically, 3 or 4 such dates are chosen during the year. **Ex:** [1,33,50,61]
Shapefiles:	Vector of Character strings	List of the shapefiles names that are stored in the (workdir)/WorkFiles/Shapefile directory. Each vector entry is in correspondance with the selected tiles provided by the Tiles items. **Ex:** ["Ref2016_T30TYP.shp","Ref2016_T31TCJ.shp"]

Table 2: List of items of the **[Parameters]** section of the job order file. (workdir) stands for the path of the working directory.

3.3.2 [Optical]

The next section is called **[Optical]** and its items are flags that allow to control the flow of treatment of the optical side of the treatment chain. Those flags listed in Table 3 are all booleans that can take the values "Yes" or "No". If some of these flags are set to the "Yes" value, their corresponding action will be performed by the chain. By default all the values are set on "No" which means that the chain will not do anything.

Item	Type	Comment
Extraction:	Boolean	Extracts all the pixels from the images that overlap all the polygons in the shapefile as well as their class labels. Depending of the size of the shapefile and the processor used, this step can take several hours per tiles and per run.
Sampling:	Boolean	Split the extracted pixels in two sets: A learning set and a validation set. Then randomly sample `NbSamples` pixels from the "Learning set". Those steps are preparation to the classification step. See Deliverable 5.3 for more details.
Classification:	Boolean	Performs supervised learning on the sampling sets of pixels then produces classification models and validation quantities.
Map:	Boolean	Produces maps from the images listed in the "OpticalImages" variables thanks to the classification models. Two type of maps is produced per tiles: a map that contains the predicted class label for each pixels and a map that contains the class probability for each pixel. Thoses maps are used later on to produce the Crop Mask product. See Deliverable 5.3 for more details.
CropClassification:	Boolean	Performs supervised learning on the sampling sets of pixels **that are only of the annual crop kind** then produce classification models and validation quantities. This classification allows the construction of the Crop Type product. See Deliverable 5.3 for more details.
CropMap:	Boolean	Produces maps from the images listed in the "OpticalImages" variables thanks to the classification models obtain during the `CropClassification` step. Two type of maps is produced per tiles: a map that contains the predicted class label for each pixels and a map that contains the class probability for each pixel. Thoses maps are used later on to produce Crop Type product. See Deliverable 5.3 for more details.

Table 3: List of items of the **[Optical]** and **[Radar]** sections of the joborder.

3.3.3 [Radar]

The third section is called **[Radar]** and plays the same role than the **[Optical]** section but for the radar side of the chain. It contains the same exact flags than the **[Optical]** section and are also listed in the Table 3. Note that in term of the chain flow, the steps listed in this section are executed after the ones listed in the **[Optical]** section.

3.3.4 [OSO], [RedEdge] and [OSORedEdge]

The next three sections are three independent options of the optical side of the chain that allows a little increase in performance of the optical classification. It is based on the use of additional spectral indices. The flags of the three sections are similar to the **[Optical]**. However, the "Extraction" and "Sampling" steps are replaced here by a "CalculateIndices" steps that produce the require spectral indices from the pixels already extracted during the optical extraction (see Deliverable 5.3 for more technical details). Here some detail about the methods:

- OSO: Stands for the french "Occupation du Sol Operationnel" (Operation Soil Usage). It uses the following spectral indices: the NDVI, the NDWI and the Brightness. The definitions of these indices can be found in the ATBD.
- RedEdge: It uses the following spectral indices: the PSRI and the CHL. The definitions of these indices can be found in the ATBD.
- OSORedEdge: Combine the OSO and Red Edge approaches.

3.3.5 [RadarOptical]

The configuration of this section allows the user to performs the classification by using both radar and optical data. The flags are similar to the flags of the **[Optical]** section excepted that the "Extraction" and "Sampling" steps are replaced here by a "Join" steps that perform the concatenation of the radar and optical pixels samples generated by the steps in the **[Optical]** and **[Radar]** sections. Note that performing classification with this method is really time and memory consuming. It should be use only for test and research purpose. In order to use both radar and optical data, the Fusion method described in the next section is much more faster and perform better.

3.3.6 [Fusion]

The last section called **[Fusion]** is where the chain can be set to performs a fusion stage task at the decision level to merge single optical and radar classification results as explained in Deliverable 5.3. It is also in this section that the creation of the products can be turned on. The fusion method is at the same time really efficent and performs significantly better than the combined optical and radar classification method. The flags of this section are detailed on table 4. An important parameter is the string called "OpticalType". It indicates to the chain which optical type of classification results the fusion stage should be based on. This parameter can take one of the following four values:

- "Optical": Classification that only uses the 10 bands of Sentinel-2
- "OSO": Classification that uses the 10 bands of Sentinel-2 + the OSO indices
- "RE" Classification that uses the 10 bands of Sentinel-2 + the Red Edge indices
- "OSORE" Classification that uses the 10 bands of Sentinel-2 + the OSO indices + the Red Edge indices

Note that it is assumed that the OSO + Red Edge method is the strategy that obtains to the best performance. Therefore, it is the default value.

The flags "CreateCropMask" and "CreateCropType" are used to activate the creation of the crop mask and crop type products with their respective confidence maps. The flags "CropMaskRegularization" and "CropTypeRegularization" activate the treatment of the crop mask and crop type with spatial regularization as explain in Deliverable 5.3. All those products can be found at the location (workdir)/Products/

Item	Type	Comment
OpticalType:	String	String parameter that indicates to the chain which optical type of classification results the fusion stage should be based on. This parameter can take one of the following four values: "Optical", "OSO", "RE" and "OSORE".
CreateCropMask:	Boolean	Create the crop mask product and its confidence map thanks to the fusion method.
CropMaskRegularization:	Boolean	Treats the crop mask product during the previous step with pixel spatial regularization.
CreateCropType:	Boolean	Create the crop type product and its confidence map thanks to the fusion method.
CropTypeRegularization:	Boolean	Treats the crop type product during the previous step with pixel spatial regularization.

Table 4: List of items of the **[Fusion]** section of the joborder.

3.4 Running

Once the job order file is well configured, the chain can be run. As a first try, it is advised to run a test chain with the minimum amount of flag turned on. It would allow the user to keep track of the ressources used by the very first step of the chain which is the extraction of the optical pixels.

To do so, its is assumed the user creates a job order files called ExampleChain.cfg and configure the **[Parameters]** section properly as seen in the prevous parts. For this example, it is assumed that the "JobName" item has the value "ExampleChain". Then the user can put all flags to "No" excepted the **[Optical]:Extraction** one that should be set to "Yes". When the file is created and save in the appropriate working directory, it can be run with the following command:

```
[user@machine]$ sensagrichain ExampleChain.cfg
```

This should produce a new file called ExampleChain.sh which is a bash script file that contain the entire instructions to run the chain. Finally the chain is started by simply executing this bash file:

```
[user@machine]$ bash ExampleChain.sh >> ExampleChain.out
```

Note the redirection to a log file. Because the chain produces a lot of debugging information, it is convenient to use such a redirection file. However this is optional.

The chain can also be run with a queue manager such as SLURM or PBS. However, it requires a slight editing of the bash file to adapt it to the user specific configuration. By default, the chain produces SLURM compatible script files but modification of the script to PBS is straightforward. In the special case of a SLURM system, the script should be then run with the command:

```
[user@machine]$ sbash ExampleChain.sh
```

No redirection file is needed in this case because the standard and error output are handled by the queue system itself.

3.5 Output products

After the chain has run with a complete set of configuration flags, a directory called "Products" will be created in the working directory. Depending on the tiles and selected dates configuration chosen during the setting, this directory will contains a group of three tiff images per processed tile:

- The crop mask map.
- The crop type map.
- The crop mask confidence map.

In the next future, a crop mask confidence map will also be produced.

The pixel legend of those products are the following:

- **For the crop mask map**: Each pixel takes the integer value 1 if it corresponds to a crop class, 0 otherwise.
- **For the crop type map**: Each pixel takes an integer value that corresponds to the class code defined in the field data shapefile. During the processing, the chain creates a file at the location (workdir)/WorkFiles/Classes-CROP.txt that maps the class code to its definition in english. This mapping is taken from the shapefile.
- **For the crop mask confidence map**: Each confidence pixel is an integer between 0 (null confidence) and 1000 (perfect confidence).

4 Troubleshooting

4.1 During installation

If an error message appears during the installation process, several things should be check

- Check if all the dependancies and requirement listed in the section 2.2 are fulfilled.
- Check if the installation was done in the user home directory.
- Check if explicit error message appears and follow them.

4.2 When using the sensagrichain command

If an error message appears when using the sensagrichain command, several things should be check

- Check if the environment variables set up in the .bashrc file are defined (see section 2.3).
- Check if the job order file is well formatted as it was at its creation. Especially, check if there is no colon symbol (":") missing between items and value parameters. Also check there is no missing sections, variables or flags.

4.3 When running the chain script

The redirection file, where the standard output of the chain script is sent, might be useful to track down errors at this stage.

Common errors at this stage are often due to path files that do not point to a correct location.

Potential memory overflow can also be detected by looking at this file. In that case, using less core that the maximum possible might keep the total memory bellow a critical value. Indeed, this change will increases the computation time.

At the moment, this version of the chain is a prototype. Therefore, it does not handle errors as well and as complete as an operating chain would do. Especially, the user should take extra care in setting the joborder flags in a consistant manner. For instant, it is possible to first run the chain with only the [Optical]:Extraction flag turned on, then run it again with only the [Optical]:Sampling flag turned on. That is consistant and should not rise any error. But if instead, the user first run the chain with only the [Optical]:Sampling flag turned on, this will create an error because the "Sampling" step will not be able to use the ouput data of the "Extraction" step that has never run. In general, if an error occurs and it is difficult to track down, the user should always be sure that the flags setting have been done in a consistant way.

4.4 Contact

Specific questions and issues can be addressed to `arnaudl@cesbio.cnes.fr`.