Solver for the CSP-Resolution : User Guide

1 Main Folder

In order to use the solver implemented for the resolution of the constraint satisfaction problem, the file **Solver** must be downloaded in its entirety. It contains :

- **Programming_Languages** : in this folder will be regrouped the **Python** modules required to run the solver, notably the module numpy; as well as the methods provided by IBM Ilog **Cplex** to implement and solve the integer linear programs.

- **Matlab_Scripts** : in this folder are the files.m necessary to create the characterization of homogeneous regions for a chosen hyperspectral scene. The output is the complete set of regions in the segmented image.

- **Python_Scripts** : in this folder are the files.py that have been put in place to use the **Cplex** solver and find a resolution for the set of regions obtained in **Matlab**.

- **Images** : several hyperspectral scenes have been stored in this folder to test the performances of the solver in various conditions.

- **Results** : after we obtained a resolution for a set of homogeneous regions, we store in this last folder a list containing the wavelengths that have been assigned to the beginning of all lines in the segmented image; a text-file which contains the parameters for the resolution (pre-processing, number of regions, number of sections) and some statistics; and finally an image which visually represents the satisfaction rate for the suggested wavelength-assignment and the complete set of homogeneous regions.

2 Programming_Languages

In the folder **Programming_Languages** will be regrouped the modules and methods required to run the solver with **Cplex** and its **Python API**. There will be 2 folders associated with the program *Cplex Optimization Studio* from IBM ILOG, and one folder which contains the modules for the **Python** version which has been chosen : **Python 3.7**. These folders must be created within the file **Solver** to be able to easily determine all paths and references necessary for future **Python** and **Matlab** functions, regardless of the computer being used.

3 Matlab_Scripts

In this section will be detailed the functionalities of the main matlab-scripts.

⋆ **Get_Image.m**

This function creates an image-object in **Matlab** from the folder **Images** defined in the previous section. In the folder, the scenes are regrouped by type and labeled with numbers. Consequently, the reference for an image is a set of 2 integers.

⋆ **anisodiff2D.m** and **segmentation.m**

Both these functions have been provided to perform the segmentation of the scene. For a random image, we have to set several parameters such as the *seuil_taille* : the minimum number of pixels in a region; or the *seuil_grad* : the higher this value is, the lesser the number of homogeneous regions in the image.

⋆ **Define_Segmentation_Parameters.m**

For a given image, the purpose of this script is to select the adequate *seuil_taille* and *seuil_grad*. In the submitted files, adequate parameters have been chosen for each of the hyperspectral scenes used as examples.

⋆ **Segmented_Image.m**

Thanks to the segmentation functions **anisodiff2D.m** and **segmentation.m**, we can, for any image-object obtain the segmented rendering where each pixel have been assigned a value with respect to the homogeneous region it belongs to.

⋆ **Save_Image.m**

With this function we save the panchromatic and segmented renderings of the hyperspectral scene in the folder **Images**.

⋆ **region.m** and **section.m**

We define *region*-objects and *section*-objects in **Matlab**. Typically, a region is characterized by its label and its set of sections; a section is characterized by the pixel at which it begins and at which it ends.

⋆ **Partitioning_Rows.m**

The purpose of this function is to go through all pixels located on one line of the segmented image, and define a set of sections for the line regardless of the homogeneous regions they belong to.

⋆ **Characterization_Regions_and_Sections.m**

We consider all the lines in the segmented image. Within this function, we define the homogeneous regions by adding successively the sections of each line. If a section belongs to a region which has already been defined, we add it to the set of sections for the region. Otherwise, we create a new region. We use the value assigned to each pixel of the segmented image to make the distinction between the regions we have already encountered and the ones we did not yet.

⋆ **From_Image_to_Text.m**

This final script creates, from the hyperspectral scene we wish to segment, the set of homogeneous regions in the image and stores it in a text file. The input for this function is the set of 2 integers which represent an image. With the image-object, the segmentation parameters and the functions provided by the LAAS, we create a segmented rendering of the hyperspectral scene. From this segmented image we deduce the homogeneous regions to store in the text-file.

4 Python_Scripts

4.1 Classes.py

The module **Classes** is comprised of 2 class-objects : *region* and *section*.

⋆ A *region* is characterized, as in **Matlab**, by its label and its set of sections. Two methods were created for this class : **add_Section** and **add_Section_2**. The first method trivially adds a section to the set of sections of the region and updates the size of the region. The second method has been defined for the relocation of sections, and only serves for this purpose. It allows us to create a second set of sections where each one is located in the first columns of the image.

⋆ A *section* is characterized by its label, the line on which it is located, the columns at which it begins and at which it ends. For the class *section* no additional method was required.

4.2 Functions.py

Several functions were designed to configure the Constraint Satisfaction Problem and find a satisfactory wavelength-assignment. The main functions have been detailed just below.

⋆ **Data_Image**

Input = *Type*, *Number*.

As in **Matlab**, an example image is defined by the set of 2 integers (*Type*, *Number*). Thanks to both these values, we can extract from a specific file in the folder the text-file with the complete characterization of homogeneous regions. We also extract, within this function, the *Height* and *Width* of the segmented image.

⋆ **Removing_Smallest_Regions**

Input = *Regions*, *Minimum_Number_of_Pixels* (*M_N_o_P*).

For a set of homogeneous regions *Regions*, we will remove from the problem the areas for which there is not at least a certain amount of pixels. The output for this function is a set in which remain the regions with more than *M_N_o_P* pixels. It is the very 1st pre-processing step.

⋆ **Grouping_of_Pixels2**

Input = *Regions*, *Grouping_Factor*.

The purpose of this function is to fuse the pixels in the image in groups of *Grouping_Factor*. This corresponds to creating a new set of homogeneous regions from *Regions*, in which each section has been sized down with respect to the *Grouping_Factor* and the pessimistic assumptions associated with this method. It is the 2nd pre-processing step.

⋆ **Subset_of_Sections**

Input = *Regions*, *Limiter*.

The purpose of this function is to keep the longest sections in each region of the set *Regions* so that a minimum amount of pixels is left within each of the regions. The parameter *Limiter* we choose for this function is an integer : we will keep the longest sections so that we have at least $Limiter \times Nb\_of\_Wavelengths$ pixels in each region. It is the 3rd pre-processing step.

$\star$ **Sort_Regions**

Input = *Regions*, *Nb_of_Wavelengths*.

Within this function we distinguish regions in 3 types depending on the value of N_o_W: regions which cannot be satisfied (not enough pixels); regions with more than N_o_W consecutive pixels on the same line; and the remaining regions for which we must find a satisfactory wavelength-assignment. This pre-processing step is to always be performed : as the only step, at the last position, or just before the relocation of sections.

$\star$ **Pre_Processing**

Input = *Regions*, *Pre_Processing*, *Pre_Processing_Parameters*, *Nb_of_Wavelengths* (N_o_W).

Regions is the complete set of homogeneous regions. *Pre_Processing* is a Boolean which is equal to 1 when we want to apply pre-processing methods to *Regions*, otherwise equal to 0. In *Pre_Processing-Parameters* are regrouped the *Minimum_Number_of_Pixels*, the *Grouping_Factor* and the *Limiter*. Within this function, for a given N_o_W and a set of regions, we apply first the 3 pre-processing steps defined earlier, and we then sort the regions with respect to the N_o_W.

$\star$ **Relocation_of_Sections**

Input = *Regions*, *Nb_of_Wavelengths*.

For a set of homogeneous regions, we run this function to obtain an equivalent set of regions where each section has been moved to the N_o_W first columns of this image. As a reminder, this pre-processing step is necessary for the 2^{nd} linear characterization and can be associated with the other methods. In case we indeed use 5 pre-processing steps for the configuration of the resolution, the **Relocation_of_Sections** is to be performed last.

$\star$ **Boundaries** and **Boundaries_of_I**

Input = *Regions*, *Nb_of_Wavelengths*.

For a λ in $\{0,...,N\_o\_W-1\}$, for a section in *Regions*, we compute thanks to **Boundaries** the two specific values $\lambda_{l,1}$ and $\lambda_{l,2}$ required for the 3^{rd} linear characterization. The purpose of the function **Boundaries_of_I** is to return two set of indices where each set contains either $I(\lambda, S)$ or $J(\lambda, S)$ for all sections in *Regions* and all λ in $\{0,...,N\_o\_W-1\}$.

$\star$ **Solution_from_1st_Column**

Input = (λ_{line}), *Nb_of_Wavelengths*.

For a set of homogeneous regions and a sample of wavelength Λ, a wavelength-assignment has been suggested by the linear solver. This wavelength-assignment is represented by the (λ_{line}) which have been allocated to the beginning of each line in the segmented image.

From the list (λ_{line}), the N_o_W and the congruence relation between pixels located on the same row, we can create a wavelength-assignment for the whole segmented image with respect to the 1^{st} requirement.

⋆ **Adequate_Assignment_Regions**

Input = *Regions*, *Solution*, *Nb_of_Wavelengths*.

For the complete set of homogeneous region *Regions* (without any pre-processing), and a wavelength-assignment *Solution* suggested by the solver, we verify which homogeneous regions are satisfied by the **WA**. The output is 2 sets of indices : the labels of the regions which have been satisfied, and the labels of the regions which were not.

Various other functions were written and used with this solver, in order to compute statistics, create text-files to store the suggested wavelength-assignment or obtain a visual rendering of the satisfaction percentage. However, they are not detailed here.

4.3 Functions_Visual.py

The python-script **Functions_Visual.py** contains the same functions as the one detailed in the section earlier. However, all visual methods have been regrouped in this script. The distinction has been made because importing the module numpy is necessary to obtain a visual rendering of the satisfaction percentage. Given that the procedure for importing numpy can be specific to the computer, some issues may arise with installing the module. If numpy cannot be easily installed on a device, the script **Functions.py** will be used in order to obtain results without numpy, but also without visuals renderings for the solution.

⋆ **Draw_Figure**

Input = *Satisfied_Regions*, *Unsatisfied_Regions*, *Height*, *Width*.

After resolution with the solver, two sets of homogeneous regions are created. The purpose of this function is to display a visual rendering of the solution where each region which has been satisfied by the suggested **WA** is in green, and each region which has not been satisfied in red. The function also allows for the image to be saved in a separate folder for further uses.

4.4 Solvers.py

In the module **Solvers** are defined 4 linear characterizations :

$\star$ **Solver_1**

Input = *Regions*, *N*, *Height*, *Width*.

Output = (λ_{line}), *problem*.

The **Solver_1** corresponds to the first linear characterization which was implemented. All pre-processing steps have already been performed on the set of homogeneous regions and we use *Height* and *Width* to assign a variable to each pixel in the image.

We create a *problem* in **Cplex**, define our variables and constraints with respect to the number of wavelenghts N and search for a satisfactory **WA**.

We will recreate the complete wavelength-assignment from the (λ_{line}) and retain the *problem* for statistics such as the number of variables, number of constraints and execution time.

$\star$ **Solver_2** and **Solver_2_Prim**

Input = *Regions*, *N*, *Height*.

Output = (λ_{line}), *problem*.

We no longer need the width of the image. Indeed, for the 2^{nd} linear characterization the width of the segmented image is equal to N.

The relocation of sections have been performed on the set of homogeneous regions already. An alternative version of the 2^{nd} characterization has been provided to compare the computing efficiency of the model with or without the use of primitives.

As for the 1^{st} characterization, we will recreate the complete wavelength-assignment from the (λ_{line}) and retain the *problem* for statistics.

$\star$ **Solver_3**

Input = *Regions*, *N*, *Height*.

Output = (λ_{line}), *problem*.

The third solver corresponds to the final model, the one which we implemented to solve images of earth-observation scene.

For this model, we need first to define the intervals $I(\lambda, S)$ or $J(\lambda, S)$ for each *section* in the set of homogeneous regions and each λ in the sample of wavelengths. There is a novelty that within the solver, we can choose whether the 2^{nd} requirement is ensured by a hard constraint or an objective. The default value being an objective.

As for the two previous characterizations, we will recreate the complete wavelength-assignment from the (λ_{line}) and retain the *problem* for statistics.

4.5 SysPath.py

The module **SysPath** has been created to define, for any computer, the adequate *sys.path* so that we can import the right modules for **Python** and **Cplex**, and easily get the example images and their characterization in homogeneous regions.

4.6 main.py

The module **main** regroups all the functions defined to solve the Constraint Satisfaction Problem.

First, we select an image for which we want an assignment of the wavelengths. This is done by extracting the set of homogeneous regions from the corresponding text-file.

We choose the parameters for the resolution, namely the number of wavelengths in Λ, the parameters for the pre-processing steps and the linear characterization for the resolution.

We obtain a list of (λ_l) from which we deduce the wavelength-assignment for the whole segmented image.

Finally we verify the satisfaction percentage of the suggested wavelength-assignment for the complete set of homogeneous regions. We compute statistics and store the results into text-files.

5 Images

As explained earlier, an image is characterized by a set of 2 integers : its *Type* and its *Number*. In the folder **Images**, there are three types :

- **Test_Easy** : images of *Type* = 1,
- **Test_Medium** : images of *Type* = 2,
- **Aerial_Scene** : images of *Type* = 3,

In each sub-folder, images are referenced with a *Number*. The total number of images within each sub-folder varies between 2 and 4. Below are the images that have been chosen as examples and saved for tests. Under each image we also provide the reference (*Type*, *Number*).

5.1 Test_Easy

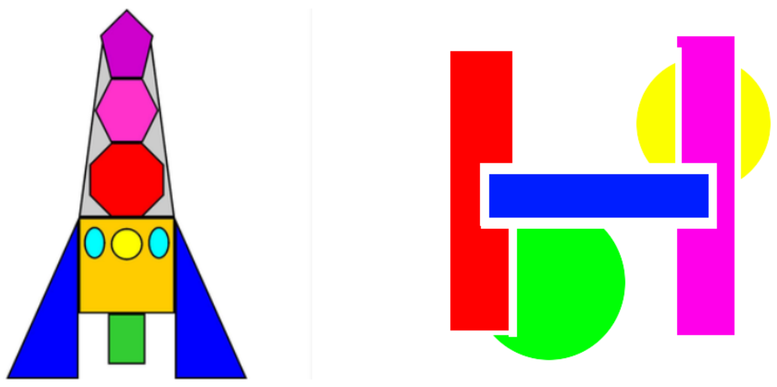


Figure 1: Left : Ref = (1,1) / Right : Ref = (1,2)

5.2 Test_Medium

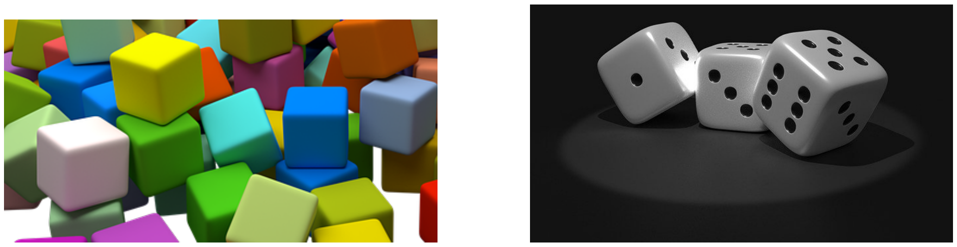


Figure 2: Left : Ref = (2,1) / Right : Ref = (2,2)

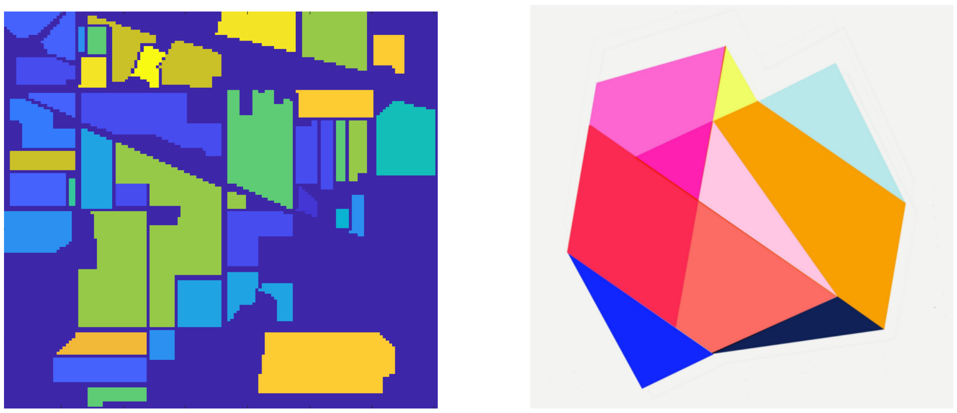


Figure 3: Left : Ref = (2,3) / Right : Ref = (2,4)

5.3 Aerial_Scene

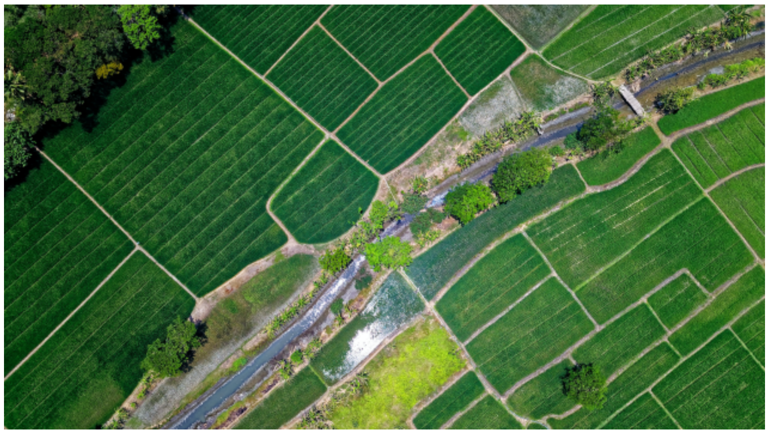
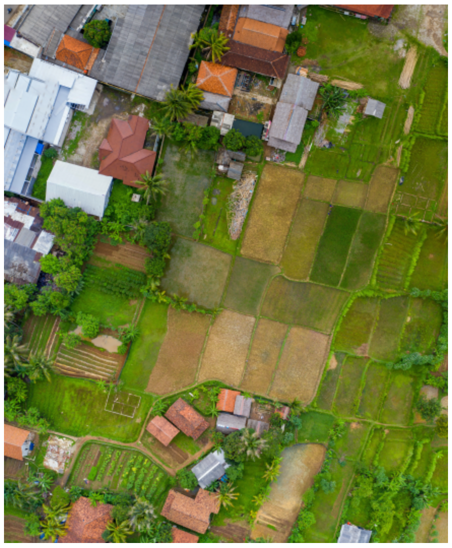


Figure 4: Ref = (3,1)



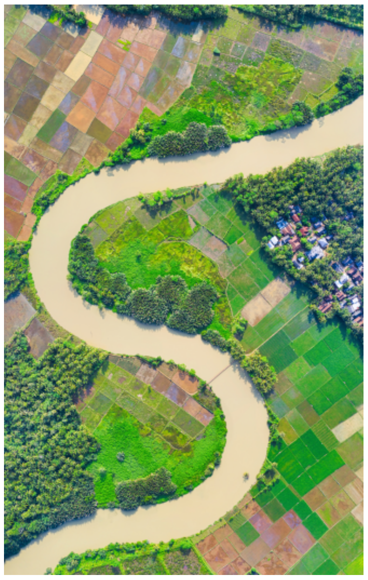


Figure 5: Left : Ref = (3,2) / Right : Ref = (3,3)

Furthermore, for the resolution of an image, the outputs for the segmentation part will be saved in the same folder next to the original picture. These outputs are :

- The panchromatic rendering of the scene,
- The segmented rendering of the scene,
- The text-file with the geometry of the image ($Height$, $Width$),
- The text-file with the characterization of homogeneous regions,

For the reference (*Type*, *Number*) of an image, both text-files will be used as input for the resolution of the CSP in **Python**.

6 Use of the Solver

In order to operate the solver, one must first download the file called **Solver** from the cloud-laas. This file contains the **Python** scripts, the **Matlab** scripts, the test-images, and two empty folders : **Results** and **Programming_Languages**.

In the folder **Programming_Languages** we must add the modules for **Python** and **Cplex** we will use for the resolution. As explained earlier, we need to download and install 3 separate files for the resolution to work.

6.1 Programming_Languages

6.1.1 Cplex

First we must obtain the optimization software : *IBM ILOG Cplex Optimization Studio*. Downloading this software comes with some difficulties. In order to be able to run the resolution without limitations on the number of variables or constraints, one must download the complete version of the software. I was able to get this version thanks to the academic initiative, and the information from the engineering school I attended. Another user would maybe have to find a complete version without the help of the academic initiative. In any case, with an IBM login and following the link just under, a user will have to download and install the complete suite : *IBM ILOG CPLEX Optimization Studio V12.10.0 Multiplatform*.

https://www.ibm.com/products/ilog-cplex-optimization-studio

Once the software has been installed, there are 2 specific folders to search for within the main folder associated with **Cplex**. On my computer, the full path of this main folder was:

/opt/ibm/ILOG/CPLEX_Studio1210/

And the paths for the 2 specific folders we need to copy and paste within **Programming_Languages** are :

/opt/ibm/ILOG/CPLEX_Studio1210/cplex/

/opt/ibm/ILOG/CPLEX_Studio1210/python/

Both files must be copied within **Programming_Languages** so that **sys.path.py** can easily search in this folder the modules required for the resolution of the CSP.

6.1.2 Python3.7

The following procedure is suggested to download the right version of **Python3.7**, which is the only one with **Python3.6** which supports **CPLEX_Studio12.10**. If a user has already **Python3.7** on his computer, he only needs to ensure that the supplementary packages *Pillow*, *six* and *numpy* are installed on the device.

In order to download **Python3.7**, I recommend the file *Gzipped source tarball* which can be downloaded from the link :

https://www.python.org/downloads/release/python-379/

All contents of the *Gzipped source tarball* must be extracted within a single file. In said file, several commands must be executed to install **Python3.7**. The complete procedure is defined in *README.srt* but for a quick installation, one only has to open a command terminal within the decompressed file and execute the following commands :

- ./configure
- make
- make test
- sudo make install

After which **Python3.7** has normally been installed on the computer.

Additional modules are required to run the solver : *Pillow*, *six* and *numpy*. Consequently, we must execute supplementary commands to install these modules. In a command terminal we will run the following lines :

- sudo python3 -m pip install –upgrade pip
- sudo python3 -m pip install Pillow
- sudo python3 -m pip install six
- sudo python3 -m pip install numpy

Once these commands have been executed, we possess the right version of **Python3.7** for a resolution with the CSP we implemented. We look for the file in the system in which all these modules have been regrouped. On my computer, the corresponding file was :

/usr/local/lib/python3.7/

We must also copy-paste this file : python3.7 in **Programming_Languages**. At the end, the folder **Programming_Languages** is comprised of three folders :

- **cplex** and **python** from */opt/ibm/ILOG/CPLEX_Studio1210/*
- **python3.7** from */usr/local/lib/*

These 3 folders are referenced and used within **SysPath.py** which is the reason why they must be duplicated the right way in **Programming_Languages**. However, this duplication can be avoided by changing in **SysPath.py** the file-paths and referencing the original folders directly

6.1.3 Matlab

The last programming language we need for the resolution is **Matlab**, with which we will perform the segmentation of the image. The procedure for downloading **Matlab** is relatively straightforward. Furthermore, no additional modules must be installed, or specific file-paths determined. Consequently, the only requirement for the use of **Matlab** is having a base version installed on the computer.

6.2 Segmentation of the Image

The first part of the resolution is obtaining the segmented rendering of the hyperspectral scene. For an image we consider in the set of Images, we determine its reference (*Type*, *Number*). As shown earlier, the reference for the typical aerial scene we used as example during the project is (3,1). Type = 3 because the image is an aerial scene, Number = 1 because the image is the first aerial scene we consider.

In order to segment the image, the user opens a command terminal within the main folder **Solver** and runs the following command :

```
matlab -nodisplay -r "cd('Matlab_Scripts');From_Image_to_Text(3,1);exit;" — tail -n +11
```

- "-nodisplay" and "exit" : it allows to enter and exit **Matlab** without the user interface,
- "-r cd('Matlab_Scripts');From_Image_to_Text" : we will run the script From_Image_to_Text located within the folder **Matlab_Scripts**,
- (3,1) : reference (*Type*, *Number*) of the image we want to segment in the folder **Images**,
- "tail -n +11" : it allows to remove from the terminal pieces of information on the **Matlab** version currently used.

When this command is being executed, 4 files will be created in the folder **Images** next to the image they refer to : the panchromatic and segmented renderings of the scene, the text-file with the characterization of the homogeneous regions for the image, and the two pieces of information $Height_{Image}$ and $Width_{Image}$.

In order to consider a custom hyperspectral scene, the image must be placed in one of the 3 folders : Test_Easy, Test_Medium or Aerial_Scene; and renamed with a number and a ".png" extension. Segmentation parameters have been defined for images in general depending on the folder they belong to. For a custom scene however, a new set of segmentation parameters can be defined for the reference (*Type*, *Number*) in the matlab-script **Define_Segmentation_Parameters.m**.

Remark: it takes up to 2 minutes to obtain the characterization in homogeneous regions for the images *Aerial_Scene_2* and *Aerial_Scene_3*. For others image, the segmentation of the scene is generally performed in under 30 seconds.

6.3 Selection of a Satisfactory Wavelength-Assignment

We consider a set of homogeneous regions in the hyperspectral scene, the remaining programming scripts have been designed in **Python3.7** so as to obtain an adequate wavelength-assignment for the set of regions.

6.3.1 Visual Representation

As explained previously, importing numpy for **Python3.7** can prove difficult. Consequently, a parameter *Visual* has been defined in the script **main.py**. If $Visual = 0$, then we refer to the algorithms written in **Functions.py** : we can find a resolution without the use of numpy, but we cannot display visual renderings with this method. If $Visual = 1$, numpy has successfully been installed on the computer and we can display the satisfaction percentage of the regions in an image. The value of *Visual* must be chosen by the user.

6.3.2 Initialisation

In order to use one of the linear characterizations designed for the resolution of the CSP, one must open a command terminal in the **Solver** directory and execute the following line:

```
python3.7 Python_Scripts/main.py 3 1
```

- ”Python_Scripts/main.py” to execute the **main.py** file in **Python_Scripts**
- (3,1) : reference (*Type*, *Number*) of the set of homogeneous regions in the image for which we want to find a satisfactory wavelength-assignment.

Once the command is executed, 3 questions will be asked to the user :

1. How many wavelengths to be considered in the sample of wavelengths (N)?
2. Which linear characterization to use for the resolution of the CSP?
3. Is it necessary to perform pre-processing steps?

For the first question, the expected answer is an integer corresponding to the size N of the sample of wavelengths.

For the second question, the answer must be an integer equal to : **1**, for a resolution with the initial model; **2**, for a resolution with the initial model and the relocation of sections; **2.5**, alternative version of the second characterization with use of primitives; **3**, for a resolution with the final model.

Finally, the last expected answer is either **1** (if Yes, pre-processing) or **0** (if No, no pre-processing steps necessary). If indeed we want to perform the 3 steps of the pre-processing before the beginning of the resolution, we will have to set values for the pre-processing factors, namely the *Minimum_Number_of_Pixels* in a region, the *Grouping_Factor*, and the *Limiter*. For this last parameter, the input must be an integer : the parameter for this step will be defined as $integer \times N$.

6.3.3 Results

After a satisfactory wavelength-assignment has been determined for the set of homogeneous regions in the hyperspectral scene, parameters for the resolution and statistics computed for this assignment will be displayed in the terminal and saved in a text-file. Depending on the value set for *Visual*, we will also have a visual rendering of the satisfaction percentage of the regions.

Remark: For the versions **2** and **2.5** of the solver, we consider a smaller image where each section has been relocated to the first N columns. In order to better understand the methodology behind this pre-processing step, a text-file *Relocation_of_Sections* is created each time the version **2** or **2.5** is used for the resolution. In this text-file are figures of size Height $\times$ N, a figure for each homogeneous region in the image. Thanks to this text-file, it is easier to represent the relocation of sections for each homogeneous region individually.

Remark: It is recommended, for a random image, not to try to solve the Constraint Satisfaction Problem while considering more than 200000 variables.