Project Pansharpening Report

Remote Sensing

Paolo Mansi – 0622701542

Alessia Carbone – 0622701487

Nina Brolich – ERASMSIN02902

06/06/2022

0. Introduction

This project required the implementation of different methods of Component Substitution for Pansharpening.

0.1. Work Division

The division of tasks provided:

- The implementation of the details extraction step was developed by Alessia Carbone
- The implementation of the details injection step was developed by Paolo Mansi
- The implementation of the segmentation algorithm was developed by Nina Brolich

Finally, all the members carried out jointly the comparison of the results obtained.

0.2. Files Organisation

The package contains:

- A "*CS_methods*" folder containing the main procedures for each algorithm
- An "*utility_procedures*" folder that contains the utility procedures used for the implementation of the algorithms
- A "*main.pro*" IDL Script, containing the testing of each algorithm
- A "*menu_pansharpening.pro*" procedure, used to enable the use of the procedures from the ENVI context
- A "*envi.men*" file containing the menu bar of the ENVI software, customized for accepting the Pansharpening procedures
- The "*PAirMax*" folder containing example images from the PAirMax dataset
- The "*ResultAnalysis.m*" MATLAB file for comparing the results with the MATLAB Toolbox
- The "*output*" folder that contains the output images produced by the main.pro script

1. Preliminary definitions

1.1 Pansharpening

As there are trade-offs among the different resolutions of the images, the fusion between images with complementary resolutions seems to be the best way to obtain high-quality products. In particular, the fusion between a panchromatic high spatial resolution image and a high spectral resolution image is called pansharpening.

We can distinguish four main classes of algorithms for pansharpening. Two classical methods, in particular component substitution and multiresolution analysis, and emerging approaches based either on variational optimization or machine learning. Even if emerging approaches have a higher accuracy, they are overly complex. For this reason, in this project, we will show you the implementation of three algorithms for component substitution approach.

All classical approaches can be considered as a unique process divided into two sequential phases. Component substitution distinguishes the extraction phase, in which the spatial details are extracted from the panchromatic image as the difference between the panchromatic image itself and a combination of the channels of the multispectral one; and the injection phase, in which the extracted spatial details, multiplied to a vector of gains, are injected as a sum operation in the image obtained by upsampling the original multispectral image to the size of the panchromatic one.

Component substitution approaches are different from one another in how they compute the extraction and the injection phases. In this project three algorithms were implemented: Brovey Transform (BT), Gram-Schmidt (GS) and Gram-Schmidt Adaptive (GSA).

1.2 Segmentation

Segmentation algorithms are used to partition images in a specified number of groups of homogenous pixels. There are multiple algorithms that achieve this

One such algorithm is k-means clustering. It aims to partition n observations into k clusters. Each observation is assigned to the cluster with the nearest mean (cluster centre).

The principal idea of the algorithm is as following:

1. Initialize the algorithm by specifying an initial set of means
2. Assign each sample to the cluster whose mean is clusters
3. Update the cluster centres
4. Test for completion: Stop, if the assignments to clusters no longer (or only marginally) change, otherwise repeat from step 2.

In many pansharpening algorithms, the coefficient matrix G_k is constant for each band. They can be implemented in a context-adaptive form, based on a segmented image. The coefficients are then only allowed to vary between partition classes but remain the same for all pixels belonging to one class.

1.3 Quality indices

The fused data must satisfy two properties:

1. Consistency, for which we must compare the original multispectral image to a degraded version of the panchromatic image obtained through decimation filters (Wavelet transform, Gauss, etc).
2. Synthesis, for which the original multispectral image acts as the reference image to be compared to the fusion result, i.e. reduced resolution protocol.

The quality indexes used in this project for the quality assessment between the fused image and the reference image are:

1. Spectral Angle Mapper (SAM) index, which describes how similar the two compared images are in terms of their spectral resolution.
2. Erreur Relative Globale Adimensionnelle de Synthèse (ERGAS) is a non-dimensional error index to quantify the synthesis error.
3. The Q and the Q2n indexes measure how similar the images are in terms of their radiometric resolution.
4. The Spatial Correlation Coefficient (SCC) is a spatial similarity index that quantifies the correlation between near pixels of the compared images.

The quality indexes are computed through a MATLAB code that compares the Ground truth image of the dataset with the produced image.

2. Brovey Transform

1.1. Algorithm explanation

The Brovey Transform is a Pansharpening approach that extracts the details with a weighted mean of the Multi Spectral image, where the weights are calculated by computing the minimum Mean Square Error between the Multi Spectral image and a smoothed version of the PAN image. The injection gains are calculated for each band of the image by a pixelwise division between the multispectral and the details image.

An improved version of the algorithm contains also a Haze correction, so the correction of undesired effects due to the scattering of the atmosphere. This haze needs to be estimated, subtracted by each band before the computation and restored afterwards.

The final multiplicative injection scheme is

$$I_{L,BT} = \sum_{i=1}^{N} \frac{1}{N} * \tilde{MS}_i$$

$$G_k = \frac{\tilde{MS}_k}{I_{L,BT}}$$

$$\hat{MS}_k = (\tilde{MS}_k - L_H) + G_k * (P^I - I_{L,BTH}) = \frac{\tilde{MS}_k - L_H}{I_{L,BTH} - L_H} * (P^I - L_H)$$

1.2. API

The Brovey Transform method can be found inside the Brovey.pro procedure.

```
BROVEY, PAN, MS, BT, RATIO=ratio, HAZE=haze
```

- `PAN` is the panchromatic image. It is required and it must be `uintarr` type.
- `MS` is the multispectral image unsampled at the size of the panchromatic one. It is required and it must be a `uintarr` type.
- `BT` is the returned fused image. It is a `fltarr` type.
- `RATIO` is the ratio of the dimensions of the panchromatic image in relation to the multispectral one. If the parameter is not specified, the `RATIO = 4`.
- `HAZE` is an additional parameter that, if set, enables the Haze correction

1.3. Implementation

1.1.1 Initialisation workspace

As first step, the images are converted to float, the sizes and the number of channels are extracted and the ratio, if not present, is set to 4.

1.1.2 Haze Correction

If the HAZE keyword is set, the haze correction is performed. It consists in an array of the same dimension as the multispectral image, where the coefficients for each band are calculated from the first percentile of each band, multiplied by a specific constant for each band. Since haze majorly affects the low bands, only the first four bands are considered for this sort of correction.

1.1.3 Extraction

- I_L calculation

The calculation of the I_L array is made by generating a gaussian smoothed PAN image, through the specific utility function. Then, the Least Squares approach is used for minimising the MSE between the smoothed

PAN and the Multi spectral image and thus calculating the weights. Finally, the weighted sum is performed on the haze corrected Multispectral image.

- P_i calculation

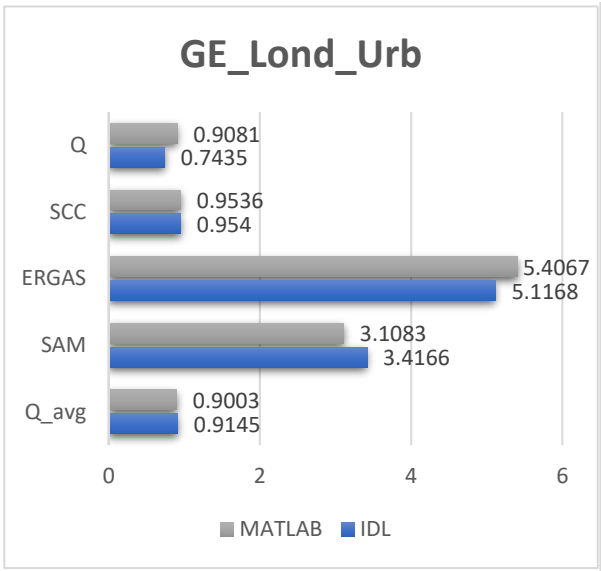
The P_i is calculated by histogram matching the original PAN to the I_L

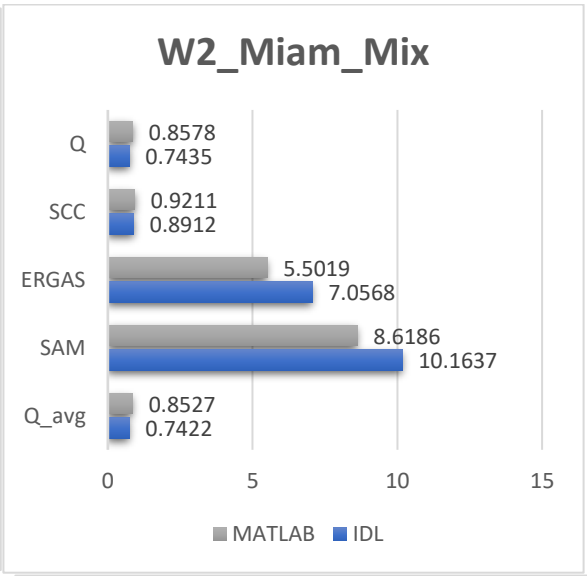
1.1.4 Injection

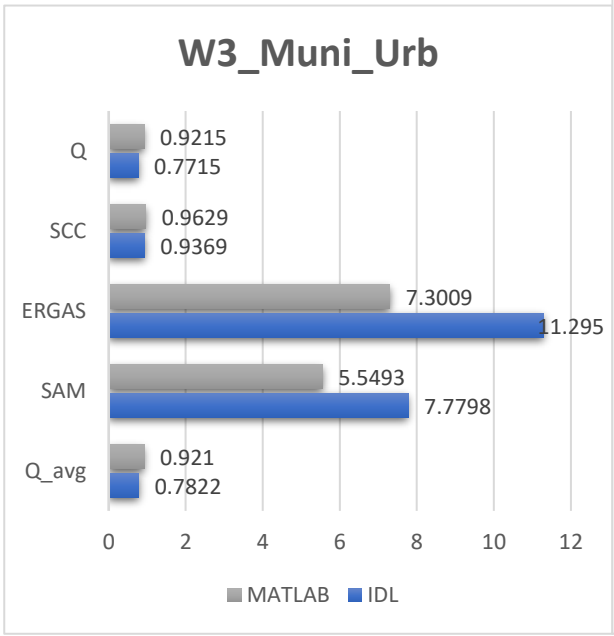
The final fusion is performed by multiplying each band of the Multi Spectral image by the Details thus extracted. In the end, the Haze factor is reinserted to restore the unbiased sharpened image.

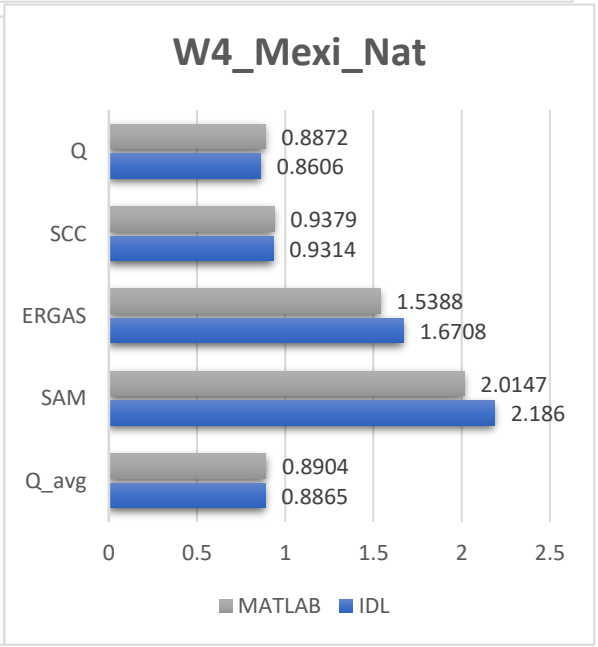
1.4. Comparison with MATLAB results

In this section, the results obtained by the Brovey transform applied both with the IDL procedures described in this file and with the MATLAB Toolbox are shown. The indices are extracted with the MATLAB script "*ResultAnalysis.m*" with four distinct set of images of the PAirMax dataset.









The images show that Brovey Transform with the IDL procedures gives results comparable with the MATLAB one.

3. Gram Schmidt

1.1. Algorithm explanation

The Gram Schmidt Pansharpening approach applies the Gram Schmidt orthogonalization process algorithm to the image, finding the projection of each band of the Multi Spectral image on the plane defined by the previously found orthogonal vectors. The intensity components are used as first vector of the new basis.

The injection gains are calculated for each band of the image by a division between the covariance of the Multispectral image in respect to the intensity components and the variance of the intensity components. In this version, the gains are static for each band.

The final multiplicative injection scheme is

$$I_{L,GS} = \sum_{i=1}^{N} \frac{1}{N} \tilde{MS}_i$$

$$G_k = \frac{cov(\tilde{MS}_k, I_{L,GS})}{var(I_{L,GS})}$$

$$\hat{MS}_k = \tilde{MS}_k + G_k * (P^I - I_{L,GS})$$

1.2. API

The Gram Schmidt approach method can be found inside the GS.pro procedure.

GS, PAN, MS, I_GS

- `PAN` is the panchromatic image. It is required and it has to be `uintarr` type.
- `MS` is the multispectral image unsampled at the size of the panchromatic one. It is required and it has to be a `uintarr` type.
- `I_GS` is the returned fused image. It is a `fltarr` type.

1.3. Implementation

1.1.1 Initialisation Workspace

As first step, the images are converted to float, the sizes and the number of channels are extracted and the Multispectral image is normalized by removing the mean values.

1.1.2 Extraction

- I_L calculation

The calculation of the I_L array is made by averaging the values of the original Multispectral image along the channels. The result is then normalized by removing the mean.

- P_i calculation

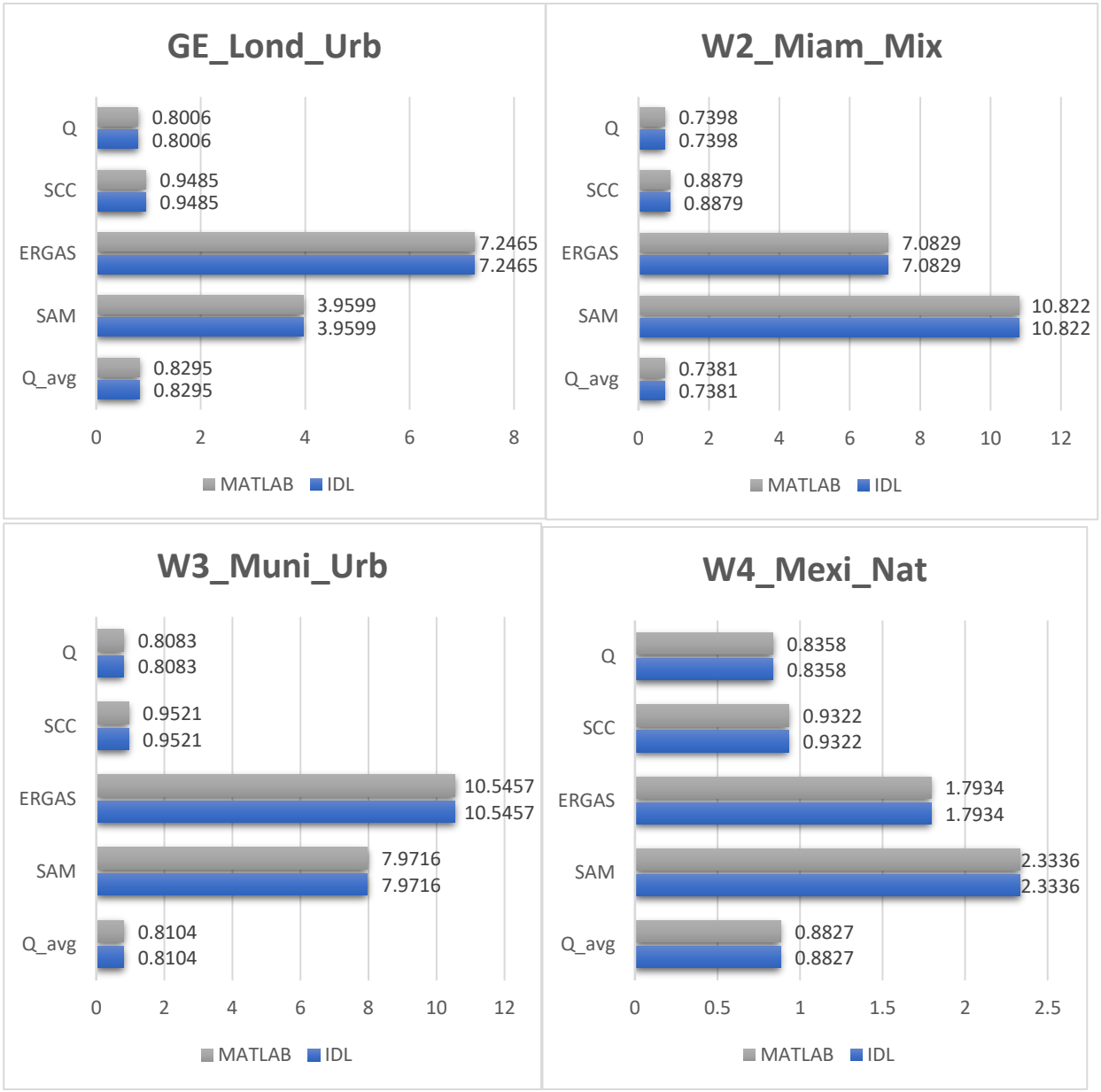
The P_i is calculated by histogram matching the original PAN to the I_L

1.1.3 Injection

The final fusion is performed by adding to each band of the Multi Spectral image the spatial resolution obtained by multiplying the injection coefficients and the related channel of the detail array. Finally, a last normalisation is performed.

1.4. Comparison with MATLAB results

In this section, the results obtained by the Brovey transform applied both with the IDL procedures described in this file and with the MATLAB Toolbox are shown. The indices are extracted with the MATLAB script “ResultAnalysis.m” with four different set of images of the PAirMax dataset



We can see that the Gram Schmidt results on both platforms are the same.

4. Adaptive Gram Schmidt

4.1 Algorithm explanation

Gram-Schmidt Adaptive algorithm is a component substitution method used for multispectral pansharpening, i.e. the fusion of a multispectral image with high spectral resolution and low spatial resolution and a panchromatic image with low spectral resolution and high spatial resolution, to obtain a fused image with high spectral resolution and high spatial resolution.

As all component substitution approaches, Gram-Schmidt Adaptive is a process divided into two sequential phases.

The **extraction** of the spatial details from the panchromatic image with high spatial resolution as the subtraction operation between the panchromatic image itself and a combination of the multispectral channels of the multispectral image interpolated at the size of the panchromatic one.

$$D = P - f(MS)$$

Gram-Schmidt Adaptive defines the function $f(MS)$ as the intensity $I = \Sigma_{i=1}^{N} w_i MS_i$ where the weights are computed as the solution of the following least-squares problem $w_i = \arg\min_{w_i} \left| P_i - \Sigma_{i=1}^{N} w_i MS_i \right|^2$.

The **injection** of the spatial details retrieved from the panchromatic image as the sum between the multispectral image unsampled at the size of the panchromatic one and the product between the spatial details themselves and a vector of gains.

$$Result = MS + G \cdot D$$

Gram-Schmidt Adaptive Algorithm defines the vector of gains as $G_k = \frac{cov(MS_k, I_{L,GSA})}{var(I_{L,GSA})}$.

4.2 API

Gram-Schmidt Adaptive algorithm can be found in the *GSA.pro* file and it can be run through the dedicated section in the *main.pro* file.

```
GSA, PAN, MS, MS_LR, I_GSA [, RATIO = ratio]
```

where:

- `PAN` is the panchromatic image. It is required and it has to be `uintarr` type.
- `MS` is the multispectral image unsampled at the size of the panchromatic one. It is required and it has to be a `uintarr` type.
- `MS_LR` is the original multispectral image. It is required and it has to be `uintarr` type.
- `I_GSA` is the returned fused image. It is a `fltarr` type.
  `RATIO` is the ratio of the dimensions of the panchromatic image in relation to the multispectral one. If the parameter is not specified, the `RATIO = 4`.

4.3 Implementation

4.3.1 Initialization Workspace

All the images are converted into `float` type and their size is retrieved:

```
size_XX = size(imageXX, /DIMENSIONS)
```

Notice that IDL matrices are saved as `[n_channels, [dimension_1 x dimension_2]]`.

In particular, multispectral images have four colour channels for red, green, blue and nir, while the panchromatic image has just a colour channel, i.e. it is a 2-dimensional matrix.

The images are then normalized by subtracting the mean value per channel computed over all the image through the `remove_mean()` utility function described in the dedicated paragraph:

```
imageXX0 = remove_mean(imageXX)
```

The panchromatic image is then smoothed according to the consistency property of the reduced resolution quality assessment used in this project. The PAN image is transformed through the wavelet transform function provided by IDL `WT = wtn(imageHR0, ratio, /OVERWRITE)`, the high frequency coefficients, i.e. HL, LH, HH, are set to zero to delete the image details, and then the result is inversely transformed `imageHR0 = wtn(smoothed, ratio, /INVERSE, /OVERWRITE)`.

4.3.2 Extraction phase

The weights `alpha` are computed as the solution of the least-squares problem between all the pixels values in the panchromatic image and the multispectral low-resolution image concatenated to a one-values matrix on the first dimension, i.e. the channels.

```
alpha[0,0, *] = la_least_squares(ILRc, IHc)
```

At the end `channels+1` weights are computed.

The intensity `I` is computed as the total sum of the products between the multispectral channels and their corresponding weight.

```
I = total(concatenation * new_alpha, 3)
```

Notice that `concatenation` here refers to the multispectral image at PAN size concatenated to a one-values matrix on the first dimension, i.e. the channels, while `new_alpha` refers to the weights repeated on the whole 2-dimensional matrix corresponding to each of their channel.

The intensity was normalized too through the `remove_mean()` utility function

```
I0 = remove_mean(I)
```

At the end of the extraction phase, the spatial details `delta` are extracted from the panchromatic image

```
delta = imageHR – I0
```

4.3.3 Injection phase

The gains `coeff` are computed for each channel of the multispectral image as the definition given above:

```
coeff[i+1] = correlate(I0, imageLR0[i,*,*], /COVARIANCE)/variance(I0)
```

To inject the spatial details extracted from the PAN image into the MS one, they have to be multiplied to the gains and the added to the multispectral image unsampled at the size of the panchromatic one.

```
V_hat = V + (new_delta*gm)
```

Notice that: `V` is the two-dimensional matrix containing for each channel, i.e. the first dimension, all the corresponding pixels values, i.e. the second dimension; `new_delta` is the 2-dimensional repetition of the spatial details extracted on each channel; `gm` is the 2-dimensional matrix containing the repetition of the coefficients `coeff` on each channel.

4.3.4 Comparison with MATLAB results

The fused image I_fus_GSA is reshaped at the size of the panchromatic one

```
I_fus_GSA[i, *,*]=reform(V_hat[i+1,*],[size_MS[1], size_MS[2]])
```

and normalized

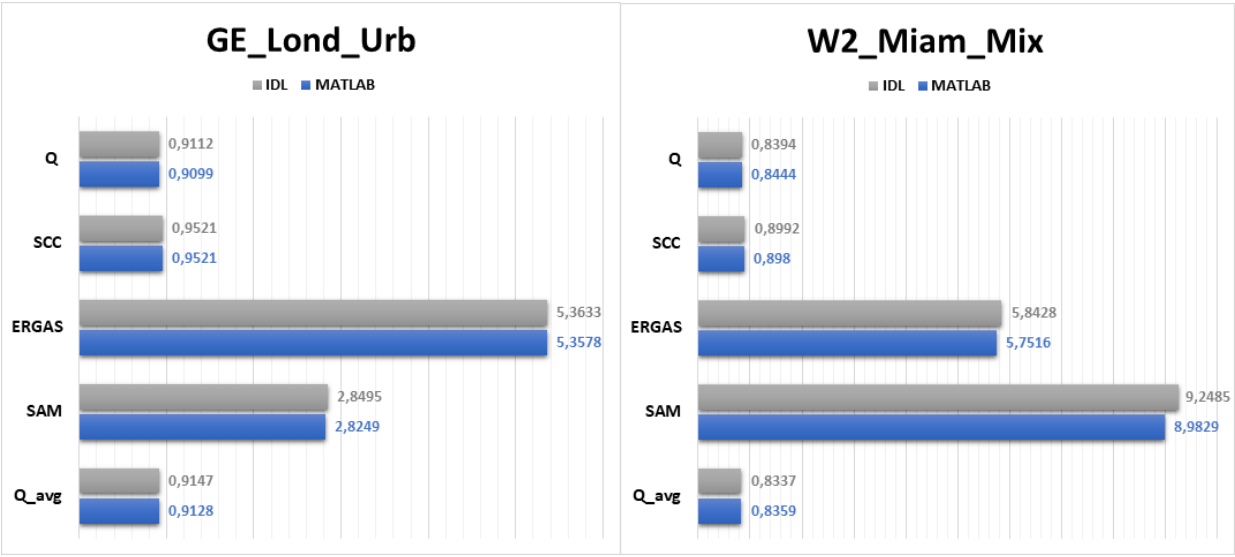
```
I_Fus_GSA[i, *,*] = h[i,*,*] - mean(h[i,*,*]) + mean(imageLR[i,*,*])
```

In the *main.pro* file the returned `I_GSA` image is saved in the *"./output"* directory as the tiff image *GSA.tif*.

The image can be easily displayed through the ENVI default menu *File, Open External File, Generic Formats, TIFF/GeoTIFF* and then selecting the *GSA.tif* from the file system. Notice that the image is a BGR image, so the *R* colour must be displayed in the *Band 3* and the colour *B* in the *Band 1*.

It is also possible to use the customized menu we created *Pansharpening, Gram-Schmidt, Adaptive* in which you can easily select the images the algorithm needs, i.e. the MS, the MS_LR and the PAN images in the PAirMax dataset, and the bands for displaying the result as discussed above.

In the project you can find the MATLAB file *ResultAnalysis.m* for the analysis of the results obtained though the computation of the quality indexes.



The plots of the quality indices for *"./PAirMax/GE_Lond_Urb"* directory and *"./PAirMax/W4_Mexi_Nat"* directory show that the results obtained with MATLAB and IDL are almost equal to each other, while the plots of the quality indices for *"./PAirMax/W2_Miami_Max"* directory and *"./PAirMax/W3_Muni_Urb"* directory show that the results obtained with MATLAB and IDL are a bit different from each other, but still similar. The only value that changes more is the ERGAS error for the *"./PAirMax/W3_Muni_Urb"* directory.

5. Adaptive Gram-Schmidt with Segmentation

The Adaptive Gram-Schmidt algorithm with segmentation is based on the previously described Adaptive Gram-Schmidt algorithm. It takes a segmented image as an input parameter.

It is implemented by the following procedure, which is in the file *GS_Segm.pro* and can be run via the dedicated section in the *main.pro*-file.

```
pro gs_segm, PAN, I_MS, I_LR_input, S, I_GS_Segm
```

As arguments, the procedure takes as input parameters

- the panchromatic image `PAN`
- the multispectral image `MS`, which is upscaled to `PAN` dimensions
- the low-resolution version of the PAN image, `I_LR_input`
- the segmented image `S`, which was segmented using the *k_means*-procedure.

All those arguments are required to have the type `uintarr`. Furthermore, the procedure takes an output argument `I_GS_Segm` of the type `fltarr`, which is later used to store the result of the procedure. All the arguments are strictly required for the algorithm.

5.1 Initialization Workspace

The values of the `I_MS, PAN,` and `I_LR_input` images are converted to double. We save the size of `I_MS` and `I_LR_input` in the variables `size_I_MS` and `size_I_LR`, as well as the number of bands of `I_MS` in the variable `channels`. The array `I_PAN` is created by replicating the `PAN`-image for each channel, with the number of channels being the number of bands in the `I_MS` image.

5.2 Adjustment of Dimensions and Transformation of Single Band Images

If `I_LR_input` is an image with a single band, it is transformed into a multiband image by replicating the single band for each channel. Then, we check if `I_LR_input` has the same number of bands as `I_PAN`. If it does not, we print an error statement.

5.3 Calculation of the Injection Matrix

The injection matrix `DetailsHRPan` is calculated by deducting `I_LR_input` from `I_PAN`.

5.4 Calculation of the Coefficients

First, we create an array of zeros for storing the coefficients. We also save an array of labels for the different partition classes of the segmented image.

For each channel, we delete singleton dimension of the corresponding `I_MS` and `I_LR_input` bands. Then, we create an array of zeros for storing the coefficients for the band.

For each of the partition classes, we get the coefficients by calculating the covariance of the pixels belonging to the partition class of the `I_LR_input` band and the `I_MS` band, and then dividing the results by the variance of the pixels belonging to the partition class of the `I_LR_input` band.

The coefficient matrix is consecutively filled with the bands of coefficients calculated in the previous step.

5.5 Fusion

Now, we can calculate the pansharpened image by multiplying the coefficient matrix with the `DetailsHRPan` matrix and adding those details onto the `I_MS` image.

5.6 The *k_means*-procedure

As stated before, the Adaptive Gram-Schmidt algorithm with segmentation takes as an input an image that was previously segmented using the *k_means*-procedure.

The *k_means*-procedure is implemented in the *k_means.pro*-file, can be run via the dedicated section of *main.pro* and has the following signature.

```
pro k_means, I_MS, segm, N_SEGM=n_segm
```

The arguments are:

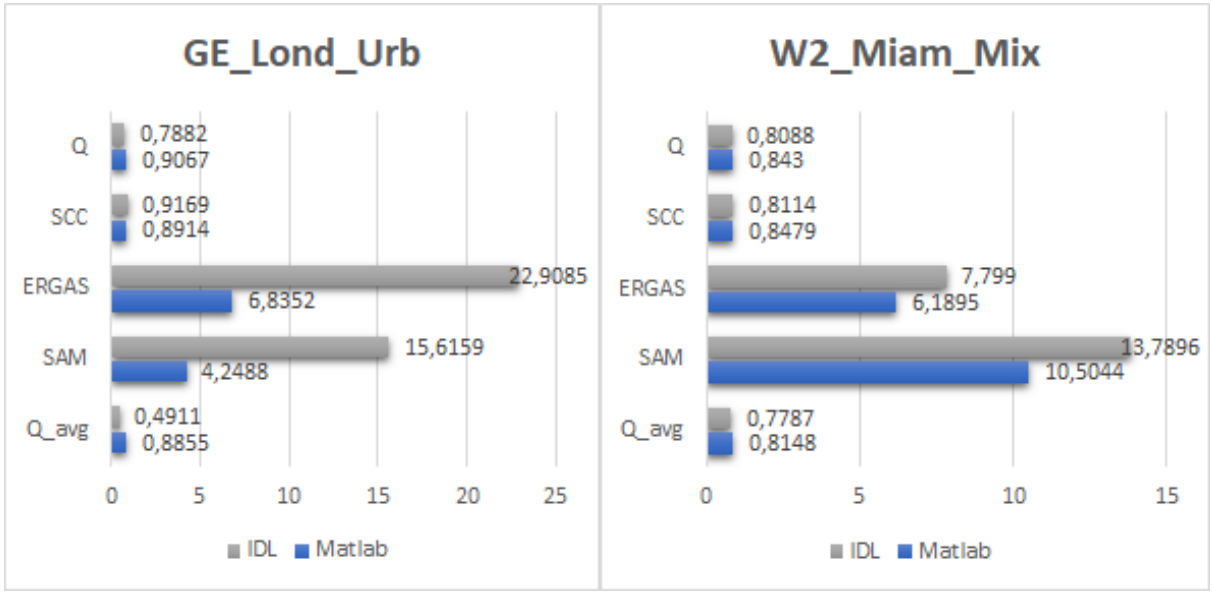
- `I_MS`: the multispectral image to segment, of type `uintarr`
- `segm`: the output image of type `fltarr`, to store the result
- `N_SEGM`: the number of partition classes of the segmentation

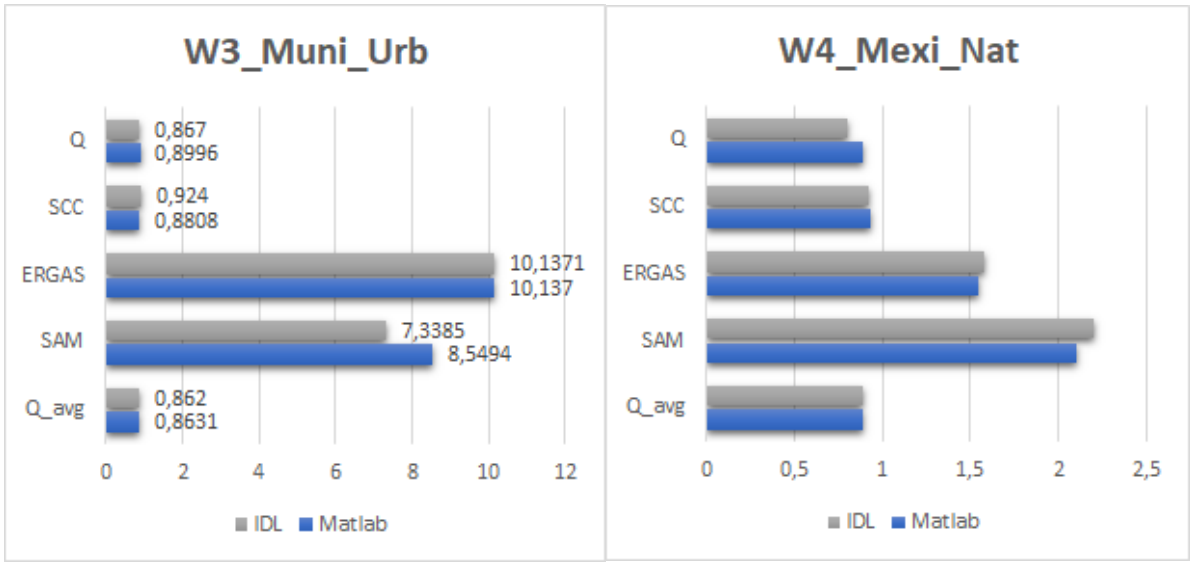
Both `I_MS` and `segm` are strictly required for the algorithm. If no `N_SEGM` is given, it will be set to a default value of 5.

The procedure resizes and normalizes the I_MS image by, for each band, dividing every value by the corresponding maximum value in the band. It then calls the built-in functions of `CLUST_WTS` and `CLUSTER`, which apply the k-means algorithm to segment the given image. Then, the procedure resizes the output of `CLUSTER` and saves it to the `segm`-array.

5.7 The results

The results are stored and analysed as previously explained. Consequently, we will compare the indices of the MATLAB results with the IDL results for four images.





./PAirMax/GE_Lond_Urb:

As we can see, for two of the indices the results differ quite extensively from the MATLAB results. The other three indices are quite similar.

./PAirMax/W2_Miam_Mix:

Again, the ERGAS and SAM indices for MATLAB and IDL are quite different, whereas the other indices obtain similar results.

./PAirMax/W3_Muni_Urb:

For the W3_Muni_Urb image, all indices except for one are almost equal. The SAM indices differ from each other, but not as much as with the previous two pictures.

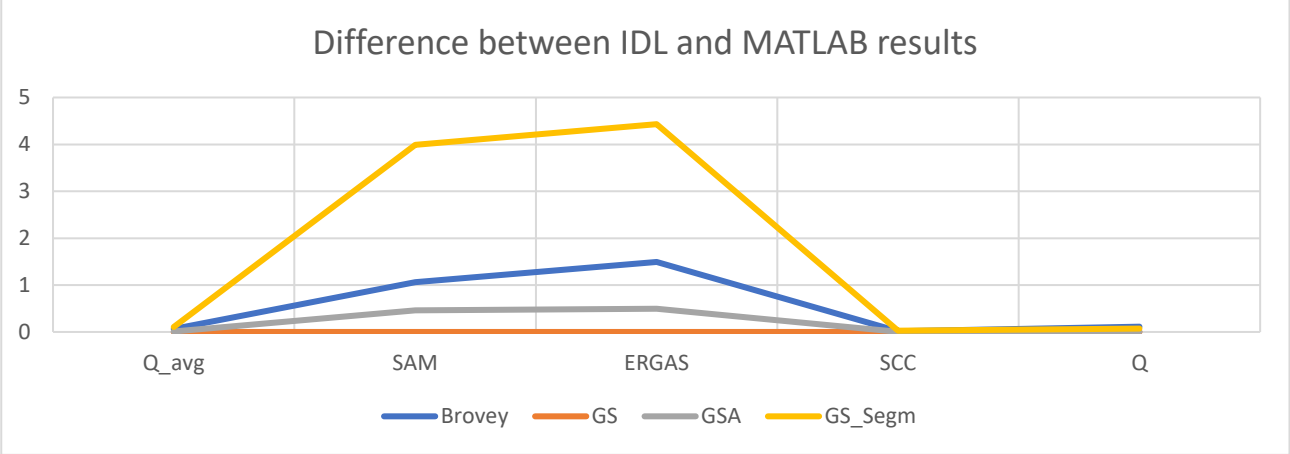
./PAirMax/W4_Mexi_Nat:

With the W4_Mexi_Nat image, the algorithm obtains satisfactory results. The indices do not differ much.

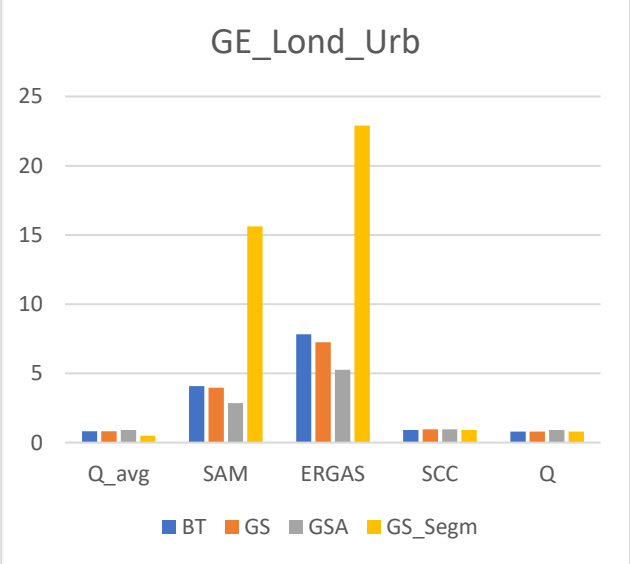
From these different tests we can infer that the performance of the GS_Segm algorithm depends on the image used.

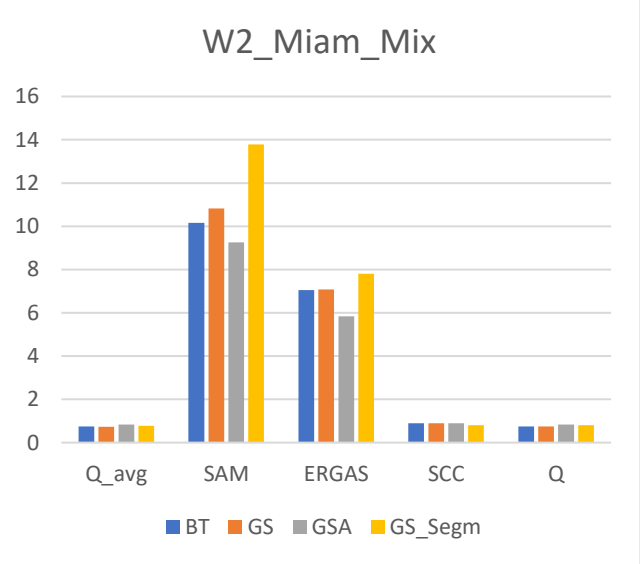
6. Result Analysis

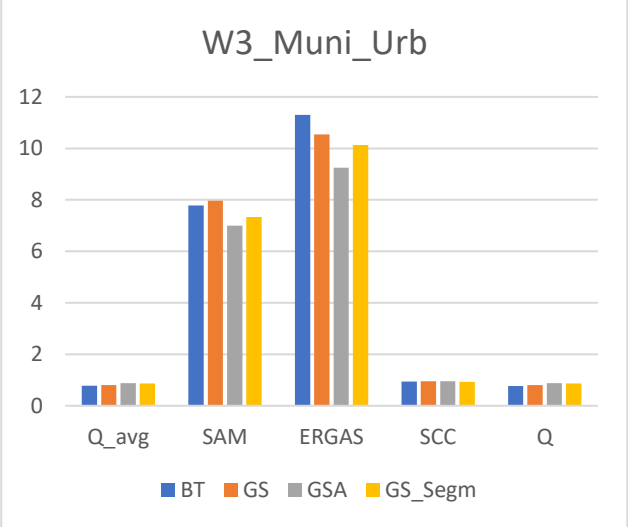
In the following figure, the average difference of the quality indexes obtained in IDL and MATLAB on the four images is shown. From the graph we can infer that the GS algorithm obtains the same quality indexes as MATLAB while the GS_Segm obtains the most different results.

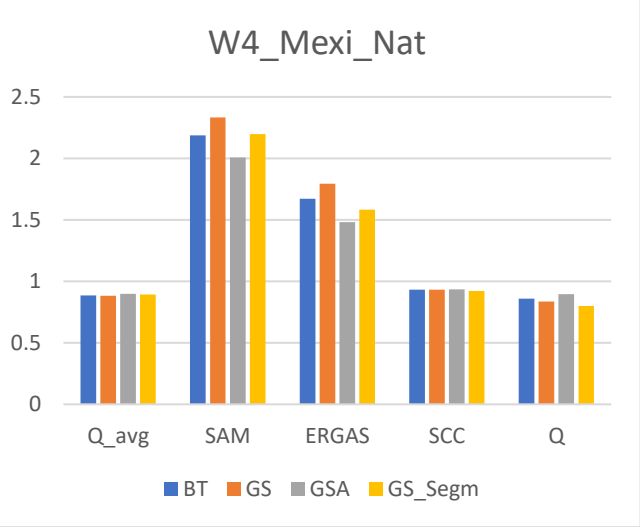


The next four figures show the quality indexes obtained with the IDL procedures on four different images of the dataset.









Analysing the results, we can point out that in urban scenarios, the algorithms perform worse because of the high number of details while in natural scenes, they all have better performances.

In particular, the GSA algorithm is the one that achieves the best performances in all the dataset used for testing.

Testing Images

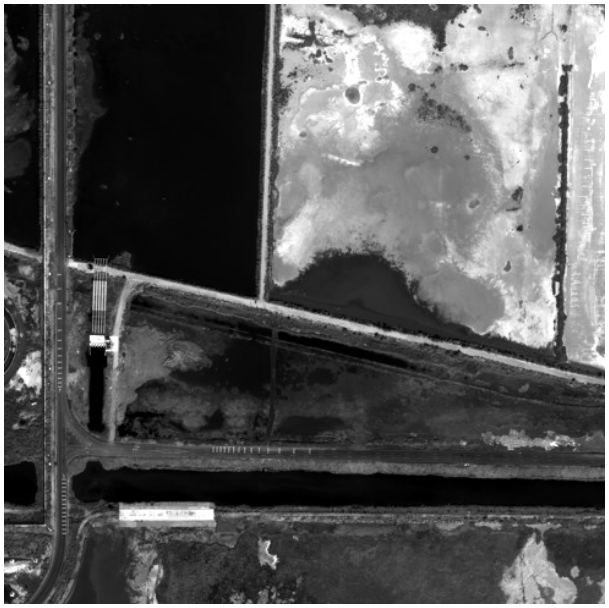


Figure 1. Panchromatic image

Figure 2. Multispectral image upsampled to PAN size

Figure 3. Brovey Pansharpening

Figure 4. GS Pansharpening

Figure 5. GS_Segm Pansharpening

Figure 6. GSA Pansharpening

7. How To Run

All the presented code is available at the following GitHub Repository: https://github.com/pmns5/Pansharpening.git

- `IDL`
  - Download the Coyote Library at the following link: http://www.idlcoyote.com/programs/zip_files/coyoteprograms.zip
  - Launch the command

```
!PATH = Expand_Path('+C:\idlfiles\coyote\') + ';' + !PATH
```

  - Launch the _main.pro_ script from an IDL Terminal

- `ENVI`
  - Copy the _envi.men_ file inside the **menu** folder of the ENVI installation files
  - Copy all the IDL Procedures inside the **save** folder of the ENVI installation files

- `Matlab`
  - Download the Toolbox at the following link: https://openremotesensing.net/knowledgebase/a-new-benchmark-based-on-recent-advances-in-multispectral-pansharpening-revisiting-pansharpening-with-classical-and-emerging-pansharpening-methods/
  - Add the toolbox Path to the MATLAB Search Paths
  - Launch the ResultAnalysis.m file