Manual of `fusionImage`, an R package for pan-sharpening images in open-source software

Fulgencio Cánovas-García, Francisco Alonso-Sarría and Paúl Pesántez-Cobos

July 2, 2020

1 Introduction

This document is the user's manual for the R package `fusionImage`. It is included, along with some test images, in the supplementary materials of the paper "`fusionImage`: An R package for pan-sharpening images in open-source software".

It is intended for the pan-sharpening of multispectral and panchromatic images coded as Digital Numbers. It has not been tested to merge multispectral images with radiance or reflectivity values. If atmospheric, illumination or topographic correction are intended, it is recommended that the images are pan-sharpened first.

The package has been tested under GNU/Linux (Xubuntu 16.04 64-bit) and under Windows 7 with version 3.4.2 of R. All the images described in the aforementioned article were tested. If problems appear with other images, please let us know and provide us with the necessary data to reproduce and fix the problem. Check out the Github page http://github.com/pacoalonso/fusionImage and CRAN repository for new versions.

2 Algorithms

2.1 High Pass Filter (HPF) algorithm

The proposal of Gangkofner et al. (2008) [5] has been implemented. The conceptual model (figure 1) consists on applying a high-pass filter to the panchromatic image (PAN) and combining this information with the multispectral images (MS) using a map algebra operation. This can be summarized in three steps:

1. Applying a high-pass filter to the panchromatic image with the aim of preserving the high frequency components and attenuate the low frequencies; it implies an enhancement of the contours of the objects or linear features in the image. This operation is achieved using a High Pass Kernel (HPK) as filtering matrix. The size of this HPK is a function of the ratio R (table 1), defined as:

   $$R = \frac{r_{ms}}{r_{pan}} \tag{1}$$

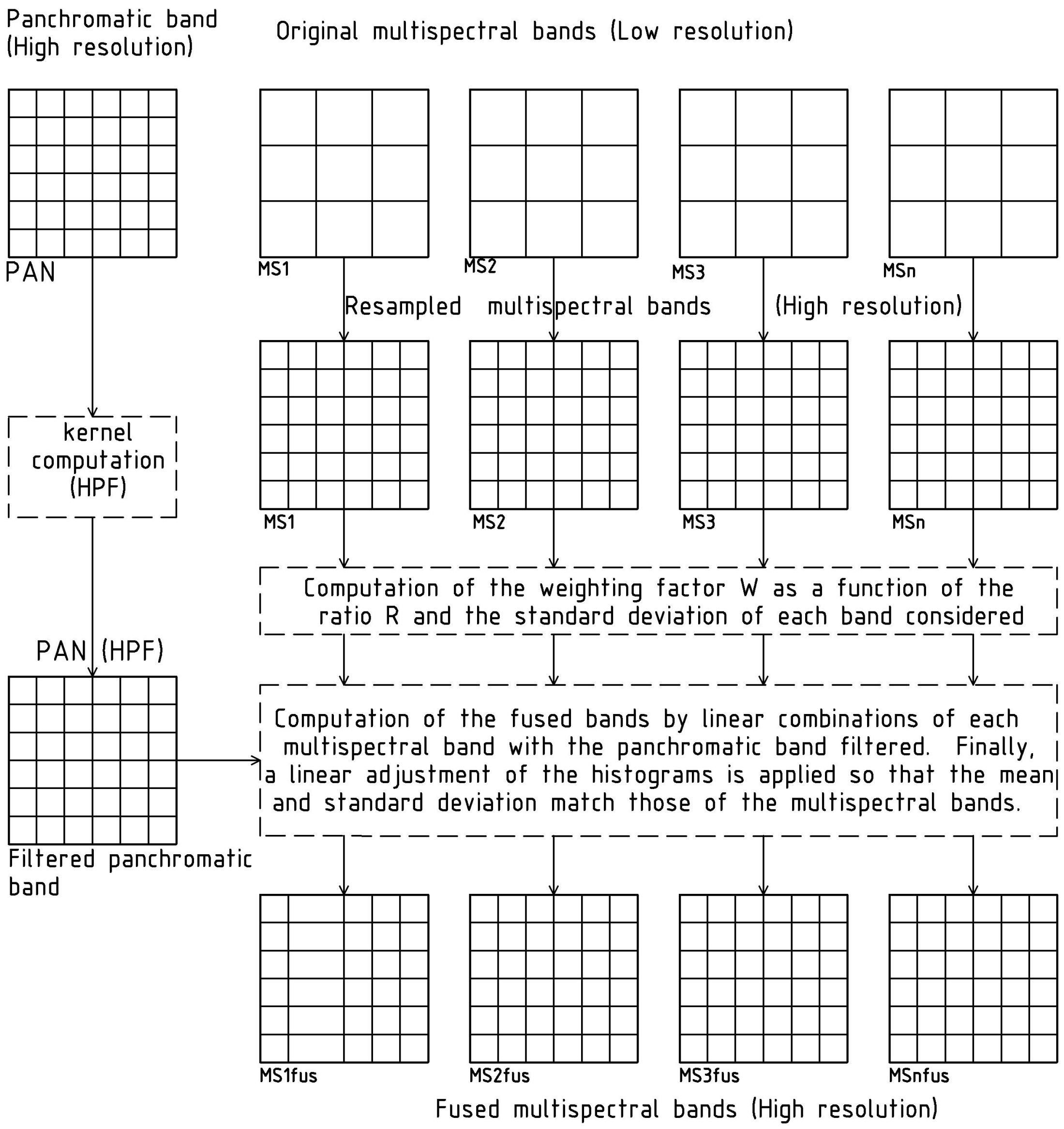
   where r_{ms} is the spatial resolution of the multispectral band and r_{pan} is the spatial resolution of the panchromatic band.

   All kernel coefficients equal -1 with the exception of the central value, which can take three values. The smallest of these is the default one, as shown in the table 1.

   It should be noted that the filtered image will be reduced in size depending on the size of the HPK, as the edges of the image are not processed.

2. Gangkofner et al. (2008) [5] proposes to add the filtered image (step 1) to each multispectral band previously resampled to the spatial resolution of the panchromatic image ($DsMS$). The resampling options included in the `raster` package are nearest neighbor and bilinear interpolation. A W weighting factor is introduced

   $$W_n = \frac{sd.MS_n}{sd.HPF} \times M \tag{2}$$



Panchromatic band (High resolution)
Original multispectral bands (Low resolution)
PAN
MS1
MS2
MS3
MSn
Resampled multispectral bands (High resolution)
kernel computation (HPF)
MS1
MS2
MS3
MSn
Computation of the weighting factor W as a function of the ratio R and the standard deviation of each band considered
PAN (HPF)
Computation of the fused bands by linear combinations of each multispectral band with the panchromatic band filtered. Finally, a linear adjustment of the histograms is applied so that the mean and standard deviation match those of the multispectral bands.
Filtered panchromatic band
MS1fus
MS2fus
MS3fus
MSnfus
Fused multispectral bands (High resolution)

Figure 1: Conceptual model of the High Pass Filter algorithm.

Table 1: Size and central value of the High Pass Kernel as a function of the ratio R (ERDAS IMAGINE, 2011)

R value	**HPK size**	**Central Value**		
		Default value	**Optional values**	
$1 < R < 2.5$	5×5	24	28	32
$2.5 \leq R < 3.5$	7×7	48	56	64
$3.5 \leq R < 5.5$	9×9	80	93	106
$5.5 \leq R < 7.5$	11×11	120	150	180
$7.5 \leq R < 9.5$	13×13	168	210	252
$R \geq 9.5$	15×15	336	392	448

Table 2: Range of M values depending on the ratio R (ERDAS IMAGINE, 2011)

HPK size	**Maximum**	**Default value**	**Minimum**
$1 < R < 2,5$	0,30	0,25	0,20
$2,5 \leq R < 3,5$	0,65	0,50	0,35
$3,5 \leq R < 5,5$	0,65	0,50	0,35
$5,5 \leq R < 7,5$	1,00	0,65	0,50
$7,5 \leq R < 9,5$	1,40	1,00	0,65
$R \geq 9,5$	2,00	1,35	1,00

where W_n is the weighting factor of the n band, MS_n is the standard deviation of each multispectral band, $sd.HPF$ is the standard deviation of the filtered image and M is a parameter that determines the intensity in the application of the filter, which is a function of the ratio of the images (table 2).

The calculation of each pan-sharpened band is given by a linear combination:

$$FMS_n = Ds.MS_n + (HPF \times W_n) \tag{3}$$

where FMS_n is each pixel of the merged image, $Ds.MS_n$ is each pixel of the resampled multispectral image, and the second term is a scalar product with HPF as the filtered panchromatic image and W_n as the weighting factor described in the equation 2.

3. Linear expansion of the histogram adjusts the image resulting from step 2 so that the mean and standard deviation match those of the original image. This allows a direct visual comparison between the pan-sharpened image and the original. The following equations are used:

$$FMS2_n = (FMS_n \times Ga_n) + Bi_n \tag{4}$$

where

$$Ga_n = \frac{sd.MS_n}{sd.FMS_n} \tag{5}$$

and

$$Bi = \overline{MS}_n - (Ga_n \times \overline{FMS}_n) \tag{6}$$

where $FMS2_n$ is the merged image applied linear expansion; FMS_n is the merged image in step 2. Ga_n (Gain) and Bi_n (Bias) are adjustment parameters calculated from $\overline{MS}_n$ and $\overline{FMS}_n$, which are the average of the original multispectral bands (MS_n) and FMS_n, respectively. $sd.MS_n$ and $sd.FMS_n$, standard deviation of MS_n and FMS_n, respectively.

As mentioned above, the methodology adopted to program the function was proposed by Gangkofner et al. (2008) [5] and implemented in ERDAS IMAGINE 2011. However, this methodology indicates that for the calculation of the W_n weighting factor, equation 2, the standard deviation of the multispectral band being merged will be used. Thus, it is not specified whether it corresponds to the standard deviation of the original multispectral band or the downsampled band. After testing the function with both options, it has been seen that the second option gives, in general, better results, and is the one that has been implemented in the function.

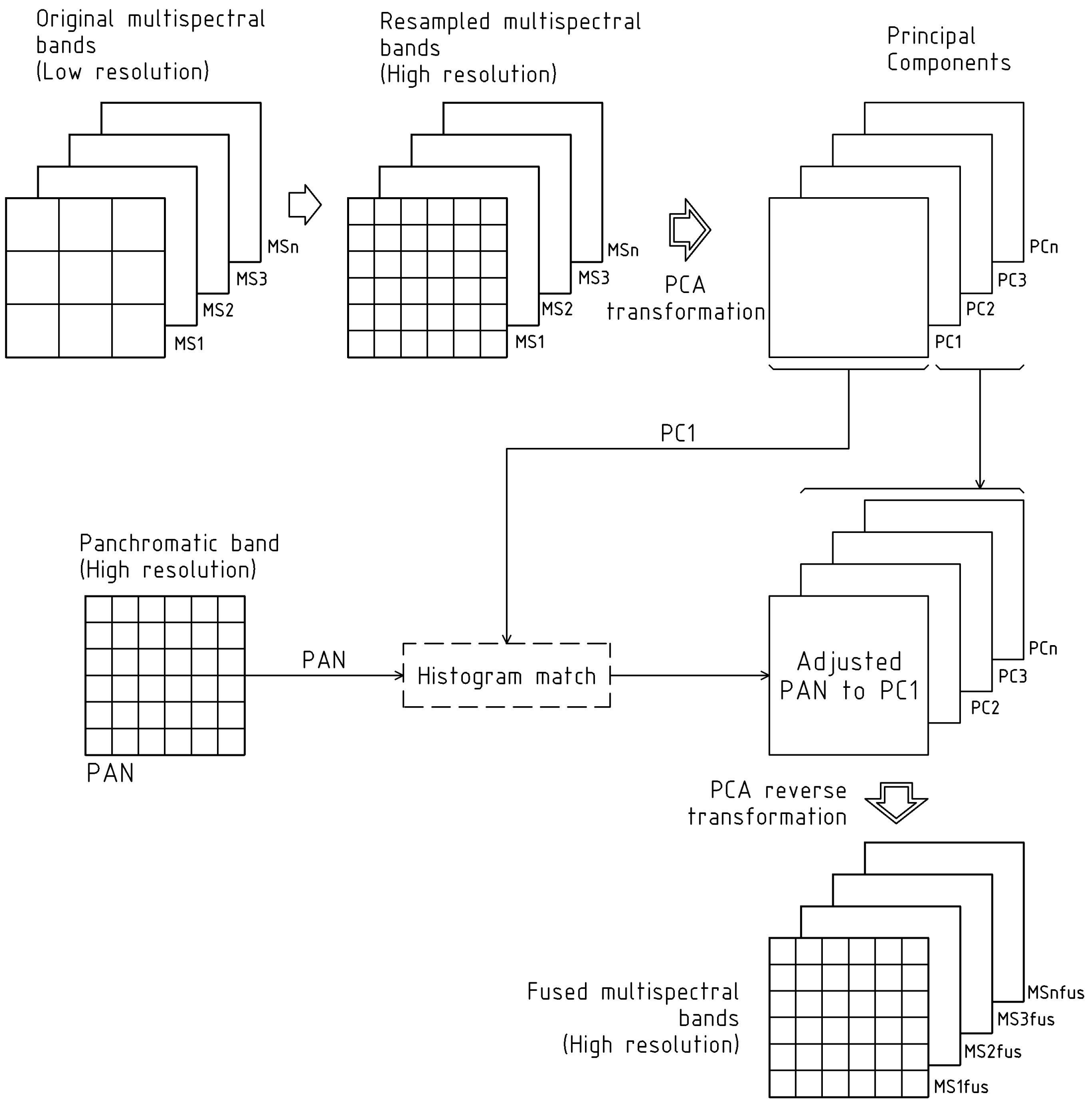

Original multispectral bands (Low resolution)
MS1
MS2
MS3
MSn
Resampled multispectral bands (High resolution)
MS1
MS2
MS3
MSn
PCA transformation
Principal Components
PC1
PC2
PC3
PCn
PC1
Panchromatic band (High resolution)
PAN
PAN
Histogram match
Adjusted PAN to PC1
PC2
PC3
PCn
PCA reverse transformation
Fused multispectral bands (High resolution)
MS1fus
MS2fus
MS3fus
MSnfus

Figure 2: Conceptual model of Principal Components Analysis fusion algorithm.

2.2 Principal Components Analysis (PCA) algorithm

PCA involves a linear transformation of the multispectral bands, the substitution of a variable in the transformed space, and the inverse transformation to the original space [13]. The process diagram is shown in figure 2.

PCA is a statistical technique that transforms a multivariate data set (in this case a set of multispectral bands) into a new set of uncorrelated components called Principal Components [15]. The number of components equals the number of multispectral original bands, and the first component (PC_1) captures the greatest variance of the dataset [11].

The high-resolution panchromatic image is adjusted to have the same variance and average as PC_1. The adjusted panchromatic image replaces the first component before performing a reverse transformation to the original space. The reason for this substitution is that the panchromatic image is approximately equal to PC_1, as this component contains information that is common to all bands, while the unique spectral information of each band is represented in the other components [3]. It is considered that this substitution maximize the effect of the high-resolution panchromatic band on the resulting pan-sharpened bands [13].

In short, the calculation of the PCA merger comprises the following steps [4]:

1. Resampling of multispectral bands to the spatial resolution of the panchromatic band.
2. Application of PCA to resampled bands.
3. Adjustment of the panchromatic band according to the mean and standard deviation of PC_1.
4. Replacement of PC_1 with the adjusted panchromatic band and inverse transformation to obtain the pan-sharpened multispectral bands into high resolution.

The function implemented in this package performs the Principal Component Analysis transformation, which includes the calculation of the covariance matrix and the matrix of eigenvectors. In some cases, the results were inconsistent, so the matrix of eigenvectors was calculated from other programs (ENVI and ERDAS), the values of certain eigenvectors differed in the sign maintaining the same absolute values. An example of the problem and how its solution has been addressed is given in the section on 3.3.

2.3 Gram-Schmidt (GS) algorithm

The GS transformation is a common technique in linear algebra and multivariate analysis whose purpose is to eliminate redundant information generally present in nearby bands of the electromagnetic spectrum. The procedure is described below [7] and the flow process diagram is shown in figure 3.

1. To calculate of a low resolution simulated panchromatic band. Two options are proposed: 1) a weighting average according of the n multispectral bands:

   $$PAN_{sim} = \sum_{n=1}^{N} w_n \times MS_n \tag{7}$$

   where w_n is a weighting factor based on sensor calibration parameters and is different for each band. 2) resampling the high-resolution panchromatic band.

   Other studies [1, 9, 6, 16, 12], propose to construct this simulated band from the mean of all multispectral bands used (equation 8). This last option offers better visual results [2] and is the one that has been implemented.

   $$PAN_{sim} = \frac{\sum_{n=1}^{N} MS_n}{N} \tag{8}$$

   The simulated panchromatic band, PAN_{sim}, is now used as a first band to be added to the low resolution multispectral data set and is used as input for the Gram-Schmidt transformation.

   $$GS_1(i,j) = PAN_{sim}(i,j) \tag{9}$$

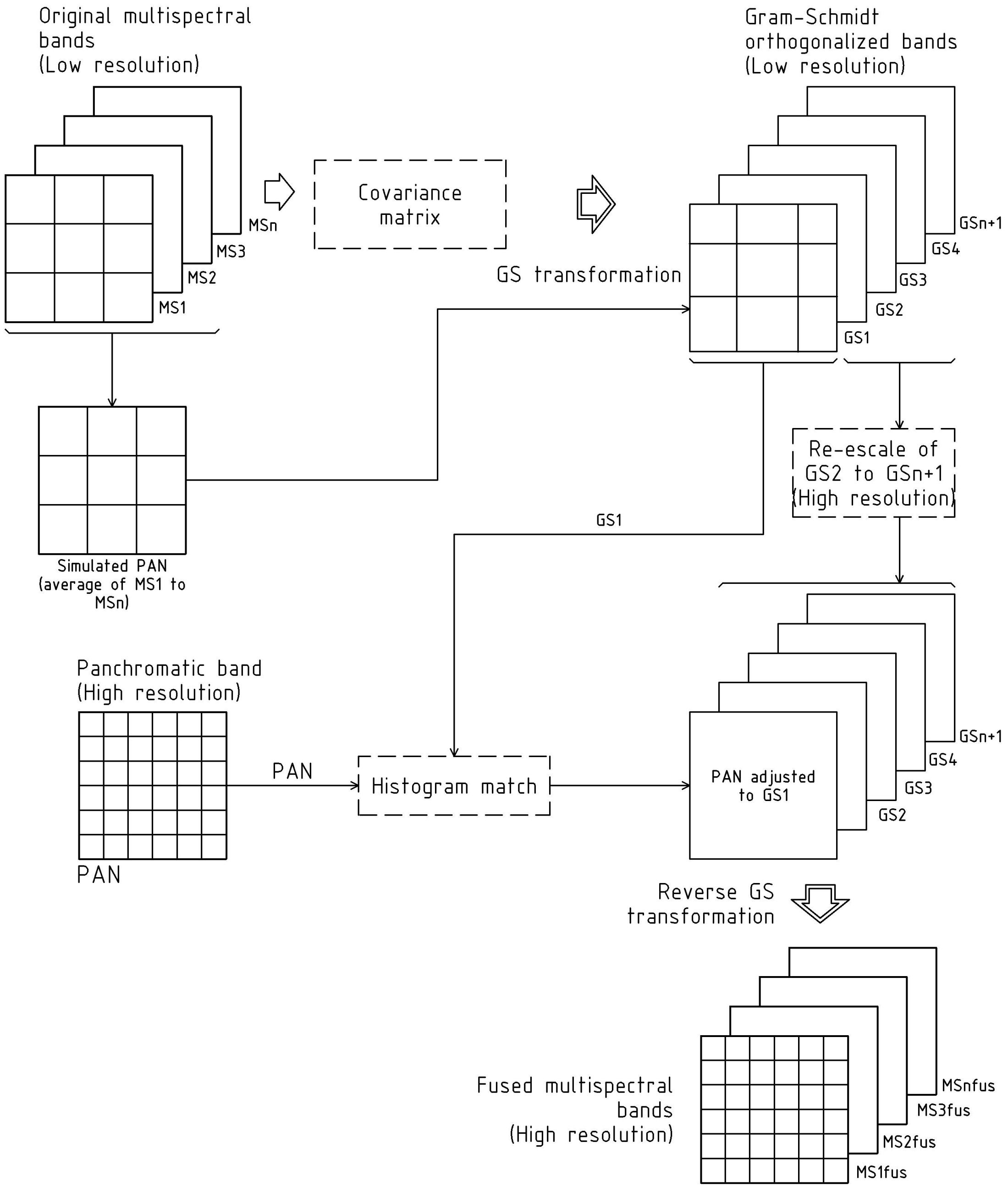

Original multispectral bands (Low resolution)
MSn
MS3
MS2
MS1
Covariance matrix
GS transformation
Gram-Schmidt orthogonalized bands (Low resolution)
GSn+1
GS4
GS3
GS2
GS1
Simulated PAN (average of MS1 to MSn)
GS1
Re-escale of GS2 to GSn+1 (High resolution)
Panchromatic band (High resolution)
PAN
PAN
Histogram match
PAN adjusted to GS1
GSn+1
GS4
GS3
GS2
Reverse GS transformation
Fused multispectral bands (High resolution)
MSnfus
MS3fus
MS2fus
MS1fus

Figure 3: Conceptual model of Gram-Schmidt fusion algorithm.

2. The Gram-Schmidt transformation is performed. Laben and Brower (2000) [7], introduce a modification: to each pixel of the image that is going to be transformed, the average of its band is subtracted before carrying out the rotation, which allows to optimize the computational calculations. In the Gram-Schmidt transformation, each GS_t band is calculated from the previous GS_{t-1}, and can be expressed as:

$$GS_t(i,j) = [B_t(i,j) - \mu_t] - \sum_{l=1}^{t-1} \phi(B_t, GS_l) * GS_l(i,j) \tag{10}$$

   where t is the band being transformed, B is the original band and μ_t is the average of band T. $\phi(B_t, GS_l)$ is calculated as:

$$\phi(B_t, GS_l) = \frac{\sigma(B_t, GS_l)}{\sigma_{GS_l}} \tag{11}$$

   that is, the covariance between the calculated GS band and the original B band, divided by the GS band variance.

3. The high-resolution panchromatic band is adjusted so that its mean and standard deviation match those of GS_1. This setting helps to preserve the spectral characteristics of the original multispectral bands. The procedure is performed following the equations 4 to 6.

4. The adjusted panchromatic band replaces the first Gram-Schmidt band (GS_1). The remaining transformed bands are resampled to the spatial resolution of the panchromatic band, and the inverse transformation of the data is performed. The equation that describes the process is:

$$B_t(i,j) = [GS_t(i,j) + \mu_t] + \sum_{l=1}^{t-1} \phi(B_t, GS_l) * GS_l(i,j) \tag{12}$$

2.4 *Erreur Relative Globale Adimensionnelle de Synthèse* (ERGAS)

The ERGAS index is used to compare the spectral quality of the pan-sharpened images. It was proposed by Wald (2000) [14] and tries to satisfy three main requirements:

1. To be independent of units, i.e. radiance values or quantities without units.
2. To be independent of the number of bands considered in the fusion.
3. To be independent of the resolution ratio between high and low spatial resolution images.

It uses the root mean square error:

$$RMSE = \sqrt{\frac{1}{P}\sum_{p=1}^{P}(Ds.MS_p - FMS_p)^2} \tag{13}$$

to measure how different are two bands. To calculate this statistic the original multispectral bands are downsampled to the spatial resolution of the panchromatic image. p is each of the pixels in the band, P is the total number of pixels, FMS_p is the value of the pixel in the pan-sharpened band and $DsMS_p$ is the value of the pixel in the downsampled multispectral band. Once the $RMSE$ for each band has been obtained, the $ERGAS$ index can be calculated:

$$ERGAS = 100\frac{r_{pan}}{r_{ms}}\sqrt{\frac{1}{N}\sum_{n=1}^{N}\frac{RMSE_n^2}{\overline{Ds.MS}_n^2}} \tag{14}$$

where r_{pan} is the spatial resolution of the panchromatic image, r_{ms} is the spatial resolution of the multi-spectral image, n refers to each of the multispectral bands involved in the pan-sharpening, N is the number of bands and $\overline{Ds.MS}_n$ is the arithmetic mean of the downsampled multispectral band n.

The value of ERGAS shows a strong tendency to decrease when the quality of the fused product increases. Values lower than 3 indicate good fusion quality [14, 10] which improves as it approaches zero.

2.5 Spatial ERGAS

As the ERGAS index only considers the spectral characteristics of the image, Lillo-Saavedra et al. (2005) [8] propose a new index called spatial ERGAS, introducing a spatial RMSE that can be calculated for each pan-sharpened band, defined by the equation:

$$spRMSE = \sqrt{\frac{1}{P}\sum_{p=1}^{P}(AdPAN_p - FMS_p)^2} \tag{15}$$

where p are the pixels in the image, P is the number of pixels in the image, $AdPAN_p$ is each of the pixels of the image obtained by adjusting the histogram of the original panchromatic image to the histogram of the downsampled multispectral band in question $(Ds.MS_n)$.

$$spERGAS = 100\frac{r_{pan}}{r_{ms}}\sqrt{\frac{1}{N}\sum_{n=1}^{N}\frac{spRMSE_n^2}{\overline{Ds.MS}_n^2}} \tag{16}$$

where r_{pan} is the spatial resolution of the panchromatic image, r_{ms} is the spatial resolution of the multi-spectral image, the subscript n refers to each of the multispectral bands involved in the fusion, N is the number of bands and $\overline{DsMS}_n$ is the arithmetic mean of $Ds.MS_n$.

3 Using the package

3.1 Installing and loading the package

This package depends on `sp` and `raster` packages.

GNU/Linux setup. Assuming that the file `fusionImage_0.0.1.tar.gz` has been downloaded in `/home/user`

```
install.packages("/home/user/fusionImage_0.0.1.tar.gz", repos = NULL, type
   = "source")
Installing package into '/home/user/R/x86_64-pc-linux-gnu-library/3.6'
(as 'lib' is unspecified)
* installing *source* package 'fusionImage' ...
** using staged installation
** R
** data
*** moving datasets to lazyload DB
** inst
** byte-compile and prepare package for lazy loading
** help
*** installing help indices
** building package indices
** installing vignettes
** testing if installed package can be loaded from temporary location
** testing if installed package can be loaded from final location
** testing if installed package keeps a record of temporary installation
   path
* DONE (fusionImage)
>
```

If it does not work, it is possible to run R as `superuser` and then enter the previous command.

GitHub setup on GNU/Linux. The devtools package is needed.

```
> library(devtools)
Loading required package: usethis
```

Once installed, the `fusionImage` package can be installed from GitHub.

```
> install_github("pacoalonso/fusionImage")
Installing package into "/home/user/R/x86_64-pc-linux-gnu-library/3.6"
(as "lib" is unspecified)
* installing *source* package "fusionImage" ...
** using staged installation
** R
** data
*** moving datasets to lazyload DB
** byte-compile and prepare package for lazy loading
** help
*** installing help indices
** building package indices
** installing vignettes
** testing if installed package can be loaded from temporary location
** testing if installed package can be loaded from final location
** testing if installed package keeps a record of temporary installation
   path
* DONE (fusionImage)
>
```

Windows 7 setup. Assuming that the file `fusionImage_0.0.1.tar.gz` has been downloaded in `C:\Users\fulgen\Downloads`. Replace "fulgen" with the actual username.

```
> install.packages("C:/Users/fulgen/Downloads/fusionImage_0.0.1.zip",
   type="source" )
Installing package into C:/Users/fulgen/Documents/R/win-library/3.4
(as lib is unspecified)
Warning message:
package C:/Users/fulgen/Downloads/fusionImage_0.0.1.zip is not available
   (for R version 3.4.1)
>
```

If as in the above case, the installation is not successful because of an outdated R version, it can be tried with:

```
utils:::menuInstallLocal()
```

A file selection dialogue will appear. Go to `Downloads` and select the file `fusionImage_0.0.1.zip`. If all went well, we'll get the following message :

```
package fusionImage successfully unpacked and MD5 sums checked
>
```

Both in Windows and GNU/Linux the package is loaded using the function `library`:

```
> library(fusionImage)
Loading required package: raster
Loading required package: sp
>
```

3.2 The hpf_fusion function

First, we import the image:

```
> l8.pan = raster("images/l8_pan_cuenca.tif") # The panchromatic image
> l8.mis = brick("images/l8_3ms_cuenca.tif") # The multispectral image
>
```

In the following function, mis indicates the object containing the lower resolution multispectral bands to be pan-sharpened, pan indicates the panchromatic image, method the resampling method, which can be ngb (nearest neighbour) or bilinear (bilinear interpolation) and bits indicates the number of bits with which the image is encoded.

```
> l8.hpf.nn = hpf_fusion(mis=l8.mis, pan=l8.pan, method="ngb", bits=16)
>
```

Here we have an example of the function using bilinear interpolation as a resampling method:

```
> l8.hpf.bil = hpf_fusion(mis=l8.mis, pan=l8.pan, method="bilinear",
  bits=16)
>
```

Now, we plot the two pan-sharpened images:

```
> par(mfrow=c(1,2)) # We split the layout in two
> plotRGB(l8.hpf.nn, r=3, g=2, b=1, stretch="lin" )  # The one of the
  nearest neighbor is printed
> plotRGB(l8.hpf.bil, r=3, g=2, b=1, stretch="lin" )  # The one of the
  bilinear interpolation is printed
>
```

Next, we zoom into a detail of the image (wastewater treatment plant of the city of Cuenca -Ecuador-).

```
> subset = extent(c(728054.6, 730733.3, -318397.8,-316375.2 ))
# pan-sharpened image
> plotRGB(l8.hpf.bil, r=3, g=2, b=1, stretch="lin", ext=subset )
# Original multiespectral image
> plotRGB(l8.mis, r=3, g=2, b=1, stretch="lin", ext=subset )
>
```

3.3 The pca_fusion function

Implementation in R of the Principal Components Analysis image fusion algorithm.

First, we import the input image:

```
> n08.pan = raster("images/n08_pan_balsa.tif")
> n08.mis = brick("images/n08_mis_balsa.tif")
>
```

In the following function, the parameters are the same than in the hpf_fusion function.

```
> n08.pca.nn = pca_fusion(mis=n08.mis, pan=n08.pan, method="ngb", bits=16)
>
```

The graphic monitor is split in two, the original multispectral image is placed on the left and the pan-sharpened image on the right.

```
> par(mfrow=c(1,2))
> plotRGB(n08.mis, r=1, g=2, b=3, stretch="lin" )
> plotRGB(n08.pca.nn, r=1, g=2, b=3, stretch="lin" )
>
```

An example of the function using bilinear interpolation as a resampling method:

```
> n08.pca.bil = pca_fusion(mis=n08.mis, pan=n08.pan, method="bilinear",
   bits=16)
>
```

We display the two pan-sharpened images

```
> par(mfrow=c(1,2)) # We split the layout in two
> plotRGB(n08.mis, r=1, g=2, b=3, stretch="lin" )
> plotRGB(n08.pca.bil, r=1, g=2, b=3, stretch="lin" )  # The one of the
   bilinear interpolation is printed
>
```

And the visualization zooming in on a detail of the image:

```
> subset = extent(c( 588116.5,  588160, 4214560, 4214586 )) # the subset
   definition
> par(mfrow=c(1,2)) # We split the layout in two
> plotRGB(n08.pca.nn, r=1, g=2, b=3, stretch="lin", ext=subset )  # nearest
   neighbour
> plotRGB(n08.pca.bil, r=1, g=2, b=3, stretch="lin", ext=subset )  #
   bilinear interpolation
>
```

In some cases, it has been observed that this function produces anomalous results. Some dark pixels appear with colours close to white. The following lines provide an example of this case, we also provide comments on how to identify the error and use a parameter of the function to correct it.

New data is imported and displayed:

```
> n08.pan = raster("images/n08_pan_archivel.tif")
> n08.mis = brick("images/n08_ms_archivel.tif")
> par(mfrow=c(1,1)) # In case the monitor was split into one part
> plotRGB(n08.mis, r=1, g=2, b=3, stretch="lin" )
>
```

The image does not have any problem, except for the particularity that its contour is irregular. We are not sure what influence this may have on the anomalous results, but we have observed that they usually appear when working with irregular contour images.

The fusion is done:

```
> n08.pca = pca_fusion(mis=n08.mis, pan=n08.pan, method="bilinear", bits=16)
WARNING: First component negatively correlated with bands. Use mode=-1
>
```

One of the results of the fusion is a warning. It tells us that the first component has negative coefficients and suggests that we use the parameter `mode` with the value -1. Negative signs in the first component are associated with anomalous results.

Now, we display the results:

```
# Display 1
> par(mfrow=c(1,2)) # We split the layout in two
> plotRGB(n08.mis, r=1, g=2, b=3, stretch="lin" ) # Original image
> plotRGB(n08.pca, r=1, g=2, b=3, stretch="lin" )  # Pan-sharpened image

# Display 2 (in detaill)
> subset = extent(c( 587417.2,  587544, 4214319, 4214405 ))
> par(mfrow=c(1,2)) # We split the layout in two
> plotRGB(n08.mis, r=1, g=2, b=3, stretch="lin", ext=subset )
> plotRGB(n08.pca, r=1, g=2, b=3, stretch="lin", ext=subset )
>
```

The result are clearly anomalous. Now let's solve the problem with the parameter mode. This is a parameter with two possible values: 1 (default option) and -1. If -1 is passed, the coefficients of the first component are multiplied by -1. In most cases, this option solves the problem. Let's look at an example:

```
> n08.pca2 = pca_fusion(mis=n08.mis, pan=n08.pan, method="bilinear",
   bits=16, mode=-1)
>
```

There is no warning.

We display the result:

```
> par(mfrow=c(1,2)) # We split the layout in two
> plotRGB(n08.mis, r=1, g=2, b=3, stretch="lin" ) # Original image
> plotRGB(n08.pca2, r=1, g=2, b=3, stretch="lin" )  # Pan-sharpened image
>
```

It can be seen that the results are good as expected.

Finally, the function allows us to enter a matrix with the component coefficients by ourselves. One possible application would be when mode=-1 does not work. In this case you can try to invert all the coefficients. In our case, just with the intention of doing a test, we will change the sign to all negative coefficients.

The matrix is created:

```
> mtx.pca = matrix( data=c(0.5614608, 0.5894995, 0.4356138, 0.3840478,
   0.03621311, 0.233387, 0.4129613, 0.879592, 0.7914442, 0.2037065,
   0.5131109, 0.2623679, 0.2388858, 0.7460057, 0.6135314, 0.09994122),
   nrow=4, byrow=TRUE )
> mtx.pca # The matrix is printed on screen
[,1]       [,2]      [,3]       [,4]
[1,] 0.56146080 0.5894995 0.4356138 0.38404780
[2,] 0.03621311 0.2333870 0.4129613 0.87959200
[3,] 0.79144420 0.2037065 0.5131109 0.26236790
[4,] 0.23888580 0.7460057 0.6135314 0.09994122
>
```

The fusion using the matrix parameter:

```
> n08.pca3 = pca_fusion(mis=n08.mis, pan=n08.pan, method="bilinear",
   bits=16, matrix=mtx.pca)
>
```

We print the results:

```
# Display 1
> par(mfrow=c(1,3)) # We split the layout in three
> plotRGB(n08.mis, r=1, g=2, b=3, stretch="lin" ) # mode=1
> plotRGB(n08.pca2, r=1, g=2, b=3, stretch="lin" ) # mode=-1
> plotRGB(n08.pca3, r=1, g=2, b=3, stretch="lin" ) # matrix
# Display 2 (in detail)
> par(mfrow=c(1,3))
> plotRGB(n08.pca, r=1, g=2, b=3, stretch="lin", ext=subset )
> plotRGB(n08.pca2, r=1, g=2, b=3, stretch="lin", ext=subset )
> plotRGB(n08.pca3, r=1, g=2, b=3, stretch="lin", ext=subset )
>
```

The result using a different matrix than the one provided by mode=1 or mode=-1 is not good, as expected. In this case it was only intended to show how to use the function, not to get a good result.

3.4 The gs_fusion function

Implementation in R of the Gram-Schmidt image fusion algorithm.

First, we import the input image:

```
> l8.pan = raster("images/l8_pan_cuenca.tif")
> l8.mis = brick("images/l8_3ms_cuenca.tif")
>
```

In the following function, the parameters are the same than in `hpf_fusion`.

```
> l8.gs.nn = gs_fusion(mis=l8.mis, pan=l8.pan, method="ngb", bits=16)
>
```

The function returns an object of class `brick` with the pan-sharpened multispectral bands.

The display is split in two, the original multispectral image is placed on the left and the pan-sharpened image on the right.

```
> par(mfrow=c(1,2))
> plotRGB(l8.mis, r=3, g=2, b=1, stretch="lin" )
> plotRGB(l8.gs.nn, r=3, g=2, b=1, stretch="lin" )
>
```

Now, an example using bilinear interpolation as a resampling method:

```
> l8.gs.bil = gs_fusion(mis=l8.mis, pan=l8.pan, method="bilinear",
bits=16)
>
```

We display the two pan-sharpened images:

```
> par(mfrow=c(1,2)) # We split the layout in two
> plotRGB(l8.gs.nn, r=3, g=2, b=1, stretch="lin" )  #The one of the nearest
   neighbour is printed
> plotRGB(l8.gs.bil, r=3, g=2, b=1, stretch="lin" )  # The one of the
   bilinear interpolation is printed
>
```

And zoom into a detailed area:

```
> subset = extent(c(728054.6, 730733.3, -318397.8,-316375.2 )) # The object
   indicating the extension is created
> plotRGB(l8.gs.bil, r=3, g=2, b=1, stretch="lin", ext=subset )  # Bilinear
   interpolation
> plotRGB(l8.mis, r=3, g=2, b=1, stretch="lin", ext=subset )  # Original
   image
>
```

3.5 The ergas_spec function

Implementation under R of the *Erreur Relative Globale Adimensionnelle de Synthèse* algorithm.

First, we import the input image:

```
> l8.pan = raster("images/l8_pan_cuenca.tif") # Panchromatic
> l8.mis = brick("images/l8_3ms_cuenca.tif") # Multiespectral
>
```

GS fusion is applied (see section **??**):

```
> l8.gs = gs_fusion(mis=l8.mis, pan=l8.pan, method="bilinear", bits=16) #
   GS fusion
>
```

Now, we compute ERGAS for the GS pan-sharpened image. The spatial resolution of the panchromatic image is specified in `rp`, the spatial resolution of the original multispectral image can be specified in `rms`, although if this value is not supplied it is computed from the multispectral image. The original multispectral image with the lowest spatial resolution is indicated in `original`, the pan-sharpened image is indicated in modified. The function resamples the images to the corresponding resolution. The sampling method can be nearest neighbour, ngb, or bilinear interpolation, bilinear. The function returns the ERGAS index, i.e. a vector of length one. The closer the value is to zero, the better the result of the fusion will be.

```
> ergas_spec(rp=15, original=l8.mis, modified=l8.gs, method="bilinear")
[1] 2.184561
>
# We get the same result if we indicate the spatial resolution of the
    multispectral image with rms=30
> ergas_spec(rp=15, rms=30, original=l8.mis, modified=l8.gs,
    method="bilinear")
[1] 2.184561
>
```

3.6 The ergas_spat function

Implementation under R of the Spatial *Erreur Relative Globale Adimensionnelle de Synthèse* algorithm.

First, we import the input image:

```
> l8.pan = raster("images/l8_pan_cuenca.tif") # Panchromatic
> l8.mis = brick("images/l8_3ms_cuenca.tif") # Multiespectral
>
```

GS fusion is applied (see section **??**):

```
> l8.gs = gs_fusion(mis=l8.mis, pan=l8.pan, method="bilinear", bits=16) #
    GS fusion
>
```

The original multispectral image with the lowest spatial resolution is indicated in original, the pan-sharpened image with the highest spatial resolution is indicated in modified and the original panchromatic image is specified in pan. The spatial resolution of the panchromatic image can be specified in rp and the spatial resolution of the original multispectral image can be specified in rms , although if these values are not supplied they are read from the corresponding images. The sampling method can be nearest neighbour, ngb, or bilinear interpolation, bilinear. The function returns the Spatial ERGAS index, i.e. a vector of length one. The closer the value is to zero, the better the result of the fusion.

```
> ergas_spat(original=l8.mis, modified=l8.gs, pan=l8.pan, method="bilinear")
[1] 0.79499
>
```

In this case the function provides a value of 0.79499.

We get the same result if we indicate the spatial resolution of the panchromatic, with rp=15, and the multispectral, with rms=30:

```
> ergas_spat(rp=15, rms=30, original=l8.mis, modified=l8.gs, pan=l8.pan,
    method="bilinear")
[1] 0.79499
>
```

Acknowledgements

This manual is partially the result of a postdoctoral contract (20023/SF/16) under the Saavedra Fajardo Program funded by the CARM (Region of Murcia Authority), through the Séneca Foundation-Agency for Science and Technology. We would like to thank the CARM Environmental Integration and Management Service (SIGA) and the United States Geological Survey for providing us with the Natmur-08 and Landsat imagery, respectively.

References

[1] B. Aiazzi, S. Baronti, M. Selva, and L. Alparone. Enhanced Gram-Schmidt Spectral Sharpening Based on Multivariate Regression of MS and Pan Data. pages 3806–3809, 2006.

[2] F. Cánovas-García and F. Alonso-Sarría. Comparación de técnicas de fusión en imágenes de alta resolución espacial. *GeoFocus*, 14:144–162, 2014.

[3] P.S. Chavez, Stuart C, and J.A. Anderson. Comparision of Three Different Methods to Merge Multiresolution and Multispectral Data: Landsat TM and SPOT Panchromatic. *Photogrammetric Engineering and Remote Sensing*, 57:295–303, 1991.

[4] A. Darvishi Boloorani. *Remotely Sensed Data Fusion as a Basis for Environmental Studies: Concepts, Techniques and Applications.* PhD thesis, Universität zu Göttingen, 2008.

[5] Ute G. Gangkofner, Pushkar S. Pradhan, and Derrold W. Holcomb. Optimizing the High-Pass Filter Addition Technique for Image Fusion. *Photogrammetric Engineering & Remote Sensing*, 74(9):1107–1118, 2008.

[6] S. Klonus and M. Ehlers. Performance of evaluation methods in image fusion. In *12th International Conference on Information Fusion*, pages 1409–1416, Seattle, USA, July 2009.

[7] C.A. Laben and B.V. Brower. Process for Enhancing the Spatial Resolution of Multispectral Imagery Using Pan-Sharpening. Technical report, United States Patent 6.011.875, 2000.

[8] M. Lillo-Saavedra, C. Gonzalo, A. Arquero, and E. Martinez. Fusion of multispectral and panchromatic satellite sensor imagery based on tailored filtering in the Fourier domain. *International Journal of Remote Sensing*, 26(6):1263–1268, 2005.

[9] S. Nussbaum and G. Menz. *Object-Based Image Analysis and Treaty Verification.* Springer, 2008.

[10] A. Ozdarici Ok and Z. Akyurek. Evaluation of Image Fusion Methods on Agricultural Lands. *Journal of Earth Science and Engineering*, 1:107–113, 2011.

[11] C. Pohl and J.L. Van Gendreen. Multisensor image fusion in remote sensing: concepts, methods and applications. *International Journal of Remote Sensing*, 19(5):823–854, 1998.

[12] Gulcan Sarp. Spectral and spatial quality analysis of pan-sharpening algorithms: A case study in Istanbul. *European Journal of Remote Sensing*, 47:19–28, 2014.

[13] V.K. Shettigara. A Generalized Component Substitution Technique for Spatial Enhancement of Multispectral lmages Using a Higher Resolution Data Set. *Photogrammetric Engineering & Remote Sensing*, 58(5):561–567, 1992.

[14] L. Wald. Quality of high resolution synthesised images: Is there a simple criterion ? In *Fusion of Earth data: merging point measurements, raster maps and remotely sensed images*, pages 99–103, Sophia Antipolis, France, January 2000. SEE/URISCA.

[15] H. Yésou, Y. Besnus, and J. Rolet. Extraction of spectral information from Landsat TM data and merger with SPOT panchromatic imagery - a contribution to the study of geological structures. *ISPRS Journal of Photogrammetry and Remote Sensing*, 48(5):23–36, 1993.

[16] Y. Zhang and R. K. Mishra. A review and comparison of commercially available pan-sharpening techniques for high resolution satellite image fusion. In *Geoscience and Remote Sensing Symposium (IGARSS)*, pages 182–185, 2012.