Librería para la fusión de imágenes satelitales sobre arquitecturas homogéneas (CPU) y heterogéneas (CPU/GPU) (Sallfus)

Manual de Usuario

Tabla de contenido

1 Descripción

La fusión de imágenes satelitales es un proceso digital que permite reunir en una imagen procesada la riqueza espectral de una imagen multiespectral y la resolución espacial de una imagen pancromática. Para llevar esto a cabo se cuenta con algunas transformaciones ya establecidas previamente, como lo son la Transformada de Brovey, Transformada Multiplicative, Transformada por análisis de componentes principales y la Transformada Á Trous, entre otras.

Cada una de las distintas técnicas empleadas en la fusión de imágenes satelitales, presentan un conjunto distinto y variado de paso a seguir. No obstante, presentan dos similitudes. La primera de ellas, hace referencia a su propósito, es decir, estas técnicas buscan enriquecer espacialmente una imagen que contenga información espectral. El segundo aspecto, está relacionado con el tipo de operaciones que se realizan, puesto que, en la mayoría de casos son operaciones matriciales, donde a medida que aumenta el tamaño de dicha matriz se eleva es costo computacional.

Al implementar los algoritmos de fusión de imágenes satelitales en forma serial, es decir, realizando su ejecución exclusivamente en CPU, se presentan tiempos elevados al utilizar imágenes de dimensiones superiores, es por esto que este proyecto busca realizar la implementación de las transformadas mencionadas anteriormente mediante procesamiento heterogéneo CPU/GPU con el fin de optimizar los tiempos de ejecución para este algoritmo. Así mismo, se tiene como objetivo proporcionar herramientas para la comparación en términos de tiempos de ejecución y evaluación de la calidad de la imagen obtenida.

2 Requisitos

Para el funcionamiento de este software, es necesario el siguiente hardware:

1. Tarjeta gráfica NVIDIA, solo si desea realizar la fusión en GPU

Librerías necesarias para ejecutar en CPU:

1. Python 3.6: https://www.python.org/download/releases/3.6/
2. Numpy 1.14.5: https://pypi.org/project/numpy/
3. Scikit-image 0.14.2: https://pypi.org/project/scikit-image/
4. Scipy 1.4.1: https://scikit-cuda.readthedocs.io/en/latest/install.html

Librerías necesarias para ejecutar en GPU:

1. Python 3.6: https://www.python.org/download/releases/3.6/
2. Numpy 1.14.5: https://pypi.org/project/numpy/
3. Pycuda 2017.1.1: https://pypi.org/project/pycuda/
4. Skcuda 0.5.2: https://scikit-cuda.readthedocs.io/en/latest/install.html
5. Cupy 7.2.0: https://scikit-cuda.readthedocs.io/en/latest/install.html
6. Scikit-image 0.14.2: https://pypi.org/project/scikit-image/

3 Descripción de funcionalidades

Esta sección, tiene como propósito definir cada una de los módulos y sus respectivas funcionalidades con las cuales el usuario puede interactuar.

Nota: Para realizar la carga de las imágenes, se pueden utilizar cualquier librería con este propósito. Sin embargo, se debe tener en cuenta que, se debe tener como dimensión de la reprentación matricial (tamaño, tamaño, bandas), es decir, si se tiene una imagen multiespectral de 1024x1024 pixeles en formato RGB, su dimensión debería ser (1024,1024,3). A continuación de presentan ejemplos de como utilizar las librerías ***georasters*** y ***scikit-image*** para esta tarea.

Georasters

```
>>> import georasters as gr
>>> data = gr.from_file('multispectral.tif')
>>> data_m = data.raster.data
>>> multispectral = np.stack((data_m[0],data_m[1],data_m[2]),axis=2)
```

De esta manera, la variable multispectral, almacena la representación matricial de la imagen y se encuentra en el formato requerido para realizar el proceso de fusión.

Scikit-image

```
>>> import skimage.io
>>> multispectral = skimage.io.imread('multispectral.tif', plugin=
      'tifffile')
```

De esta manera, la variable multispectral, almacena la representación matricial de la imagen y se encuentra en el formato requerido para realizar el proceso de fusión.

3.1.1 sallfus.fusion.broveyCPU

3.1.1.1 fusion_images

Realiza la fusión de imágenes satelitales mediante la transformada de Brovey, realizando su procesamiento en una arquitectura homogénea de forma serial haciendo uso de la CPU.

sallfus.fusion.broveyCPU.fusion_images*(multispectral, panchromatic, save_image= False , savepath=None, timeCondition=True)*

Parámetros:

multispectral: ***array-like***
Arreglo 2-D que contiene la representación matricial de la imagen multiespectral que se quiere utilizar en el proceso de fusión

panchromatic: ***array-like***
Arreglo 2-D que contiene la representación matricial de la imagen pancromática que se quiere utilizar en el proceso de fusión

save_image: ***bool, opcional***

		Si *save_image* es Falso (Por defecto), entonces no se guarda la imagen resultante del proceso de fusión. En otro caso, se realizar el almacenamiento de la imagen.
	savepath:	***string, opcional*** Dirección local donde se desea almacenar la imagen. Por defecto, la imagen se guardará en la dirección donde se esté ejecutando la fusión.
	timeCondition:	***bool, opcional*** Si *timeCondition* es Verdadero, durante la fusión de imágenes se calculará el tiempo de ejecución de la transformada. De otra manera, no se calculará este tiempo.
Retornos:	**final_results:**	*dictionary* Si *timeCondition* es Verdadero, se retornará un diccionario almacenando la imagen resultado y el tiempo de ejecución. La obtener la imagen, realiza una indexación mediante la clave “image” y para el tiempo mediante la clave “time”.
	final_image:	*ndarray* Si *timeCondition* es Falso, se retornará un arreglo con la imagen resultado.

Ejemplos:

```
>>> from sallfus.fusion import broveyCPU
>>> import skimage.io
>>> m = skimage.io.imread('multispectral.tif', plugin='tifffile')
>>> p = skimage.io.imread('panchromatic.tif', plugin='tifffile')
>>> results = broveyCPU.fusion_images(m,p)
>>> results[“image”]
>>>                              [[[18 29 19]
                                  [19 30 20]
                                  [42 68 46]
                                      ...
                                  [10 16 12]
                                  [22 35 27]
                                 [25 39 30]]
                                      ...
                                 [[28 55 34]
                                  [30 60 36]
                                  [28 55 34]
                                      ...
                                  [17 26 19]
                                  [17 26 19]
                                 [18 27 20]]]
>>> results[“time”]
>>> 81.40903687477112
```

3.1.2 sallfus.fusion.broveyGPU

3.1.2.1 fusion_images

Realiza la fusión de imágenes satelitales mediante la transformada de Brovey, realizando su procesamiento en una arquitectura heterogénea de forma paralela haciendo uso de la CPU y la GPU.

sallfus.fusion.broveyGPU.fusion_images*(multispectral, panchromatic, save_image= False , savepath=None, timeCondition=True)*

Parámetros:

multispectral: ***array-like***
Arreglo 2-D que contiene la representación matricial de la imagen multiespectral que se quiere utilizar en el proceso de fusión

panchromatic: ***array-like***
Arreglo 2-D que contiene la representación matricial de la imagen pancromática que se quiere utilizar en el proceso de fusión

save_image: ***bool, opcional***
Si *save_image* es Falso (Por defecto), entonces no se guarda la imagen resultante del proceso de fusión. En otro caso, se realizar el almacenamiento de la imagen.

savepath: ***string, opcional***
Dirección local donde se desea almacenar la imagen. Por defecto, la imagen se guardará en la dirección donde se esté ejecutando la fusión.

timeCondition: ***bool, opcional***
Si *timeCondition* es Verdadero, durante la fusión de imágenes se calculará el tiempo de ejecución de la transformada. De otra manera, no se calculará este tiempo.

Retornos:

final_results: *dictionary*
Si *timeCondition* es Verdadero, se retornará un diccionario almacenando la imagen resultado y el tiempo de ejecución. La obtener la imagen, realiza una indexación mediante la clave “image” y para el tiempo mediante la clave “time”.

final_image: *ndarray*
Si *timeCondition* es Falso, se retornará un arreglo con la imagen resultado.

Ejemplos:

```
>>> from sallfus.fusion import broveyGPU
>>> import skimage.io
>>> m = skimage.io.imread('multispectral.tif', plugin='tifffile')
>>> p = skimage.io.imread('panchromatic.tif', plugin='tifffile')
>>> results = broveyGPU.fusion_images(m,p)
>>> results[“image”]
```

```
>>>                                [[[18 29 19]
                                    [19 30 20]
                                    [42 68 46]
                                       ...
                                    [10 16 12]
                                    [22 35 27]
                                   [25 39 30]]
                                       ...
                                   [[28 55 34]
                                    [30 60 36]
                                    [28 55 34]
                                       ...
                                    [17 26 19]
                                    [17 26 19]
                                   [18 27 20]]]
>>> results[“time”]
>>> 1.5630266666412354
```

3.1.3 sallfus.fusion.multiplicativeCPU

3.1.3.1 fusion_images

Realiza la fusión de imágenes satelitales mediante la transformada de Multiplicative, realizando su procesamiento en una arquitectura serial haciendo uso de la CPU.

sallfus.fusion.multiplicativeCPU.fusion_images*(multispectral, panchromatic, save_image= False , savepath=None, timeCondition=True)*

Parámetros:

Multispectral: ***array-like***
Arreglo 2-D que contiene la representación matricial de la imagen multiespectral que se quiere utilizar en el proceso de fusión

panchromatic: ***array-like***
Arreglo 2-D que contiene la representación matricial de la imagen pancromática que se quiere utilizar en el proceso de fusión

save_image: ***bool, opcional***
Si *save_image* es Falso (Por defecto), entonces no se guarda la imagen resultante del proceso de fusión. En otro caso, se realizar el almacenamiento de la imagen.

savepath: ***string, opcional***
Dirección local donde se desea almacenar la imagen. Por defecto, la imagen se guardará en la dirección donde se esté ejecutando la fusión.

timeCondition: ***bool, opcional***
Si *timeCondition* es Verdadero, durante la fusión de imágenes se calculará el tiempo de ejecución de la transformada. De otra manera, no se calculará este tiempo.

Retornos: **final_results:** *dictionary*
Si *timeCondition* es Verdadero, se retornará un diccionario almacenando la imagen resultado y el tiempo de ejecución. La obtener la imagen, realiza una indexación mediante la clave “image” y para el tiempo mediante la clave “time”.

final_image: *ndarray*
Si *timeCondition* es Falso, se retornará un arreglo con la imagen resultado.

Ejemplos:

```
>>> from sallfus.fusion import multiplicativeCPU
>>> import skimage.io
>>> m = skimage.io.imread('multispectral.tif', plugin='tifffile')
>>> p = skimage.io.imread('panchromatic.tif', plugin='tifffile')
>>> results = multiplicativeCPU.fusion_images(m,p)
>>> results[“image”]
>>>                          [[[ 5  8  7]
                              [ 6  8  7]
                              [13 18 16]
                                  ...
                              [ 2  3  3]
                              [ 6  8  8]
                              [ 6  9  9]]
                                  ...
                             [[ 7 13 10]
                              [ 8 14 11]
                              [ 7 13 10]
                                  ...
                              [ 4  6  6]
                              [ 4  6  6]
                             [ 4  6  6]]]
>>> results[“time”]
>>> 9.04886245727539
```

3.1.4 sallfus.fusion.multiplicativeGPU

3.1.4.1 fusion_images

Realiza la fusión de imágenes satelitales mediante la transformada de Multiplicative, realizando su procesamiento en una arquitectura heterogénea de forma paralela haciendo uso de la CPU y la GPU.

sallfus.fusion.multiplicativeGPU.fusion_images*(multispectral, panchromatic, save_image= False , savepath=None, timeCondition=True)*

Parámetros:

Multispectral: ***array-like***

Arreglo 2-D que contiene la representación matricial de la imagen multiespectral que se quiere utilizar en el proceso de fusión

panchromatic: ***array-like***
Arreglo 2-D que contiene la representación matricial de la imagen pancromática que se quiere utilizar en el proceso de fusión

save_image: ***bool, opcional***
Si *save_image* es Falso (Por defecto), entonces no se guarda la imagen resultante del proceso de fusión. En otro caso, se realizar el almacenamiento de la imagen.

savepath: ***string, opcional***
Dirección local donde se desea almacenar la imagen. Por defecto, la imagen se guardará en la dirección donde se esté ejecutando la fusión.

timeCondition: ***bool, opcional***
Si *timeCondition* es Verdadero, durante la fusión de imágenes se calculará el tiempo de ejecución de la transformada. De otra manera, no se calculará este tiempo.

Retornos:

final_results: *dictionary*
Si *timeCondition* es Verdadero, se retornará un diccionario almacenando la imagen resultado y el tiempo de ejecución. La obtener la imagen, realiza una indexación mediante la clave “image” y para el tiempo mediante la clave “time”.

final_image: *ndarray*
Si *timeCondition* es Falso, se retornará un arreglo con la imagen resultado.

Ejemplos:

```
>>> from sallfus.fusion import multiplicativeGPU
>>> import skimage.io
>>> m = skimage.io.imread('multispectral.tif', plugin='tifffile')
>>> p = skimage.io.imread('panchromatic.tif', plugin='tifffile')
>>> results = multiplicativeGPU.fusion_images(m,p)
>>> results[“image”]
>>>                          [[[ 5  8  7]
                              [ 6  8  7]
                              [13 18 16]
                                  ...
                              [ 2  3  3]
                              [ 6  8  8]
                              [ 6  9  9]]
                                  ...
                             [[ 7 13 10]
                              [ 8 14 11]
                              [ 7 13 10]
```

```
                ...
            [ 4  6  6]
            [ 4  6  6]
           [ 4  6  6]]]
>>> results["time"]
>>> 1.169908046722412
```

3.1.5 sallfus.fusion.pcaCPU

3.1.5.1 fusion_images

Realiza la fusión de imágenes satelitales mediante la transformada por Análisis de Componentes Principales (PCA), realizando su procesamiento en una arquitectura serial haciendo uso de la CPU.

sallfus.fusion.pcaCPU.fusion_images*(multispectral, panchromatic, save_image= False , savepath=None, timeCondition=True)*

Parámetros:

Multispectral: ***array-like***
Arreglo 2-D que contiene la representación matricial de la imagen multiespectral que se quiere utilizar en el proceso de fusión

panchromatic: ***array-like***
Arreglo 2-D que contiene la representación matricial de la imagen pancromática que se quiere utilizar en el proceso de fusión

save_image: ***bool, opcional***
Si *save_image* es Falso (Por defecto), entonces no se guarda la imagen resultante del proceso de fusión. En otro caso, se realizar el almacenamiento de la imagen.

savepath: ***string, opcional***
Dirección local donde se desea almacenar la imagen. Por defecto, la imagen se guardará en la dirección donde se esté ejecutando la fusión.

timeCondition: ***bool, opcional***
Si *timeCondition* es Verdadero, durante la fusión de imágenes se calculará el tiempo de ejecución de la transformada. De otra manera, no se calculará este tiempo.

Retornos:

final_results: *dictionary*
Si *timeCondition* es Verdadero, se retornará un diccionario almacenando la imagen resultado y el tiempo de ejecución. La obtener la imagen, realiza una indexación mediante la clave “image” y para el tiempo mediante la clave “time”.

final_image: *ndarray*

Si *timeCondition* es Falso, se retornará un arreglo con la imagen resultado.

Ejemplos:

```
>>> from sallfus.fusion import pcaCPU
>>> import skimage.io
>>> m = skimage.io.imread('multispectral.tif', plugin='tifffile')
>>> p = skimage.io.imread('panchromatic.tif', plugin='tifffile')
>>> results = pcaCPU.fusion_images(m,p)
>>> results[“image”]
>>>                              [[[10 26 27]
                                  [11 27 28]
                                  [40 56 50]
                                      ...
                                 [ 2 14  22]
                                 [17 29  34]
                                 [21 33  37]]
                                      ...
                                 [[24 47 38]
                                  [28 50 40]
                                  [24 47 38]
                                      ...
                                  [11 22 27]
                                  [11 22 27]
                                [12  23  28]]]
>>> results[“time”]
>>> 21.54115915298462
```

3.1.6 sallfus.fusion.pcaGPU

3.1.6.1 fusion_images

Realiza la fusión de imágenes satelitales mediante la transformada por Análisis por Componentes Principales, realizando su procesamiento en una arquitectura heterogénea de forma paralela haciendo uso de la CPU y la GPU.

sallfus.fusion.pcaGPU.fusion_images*(multispectral, panchromatic, save_image= False , savepath=None, timeCondition=True)*

Parámetros:

Multispectral: ***array-like***
Arreglo 2-D que contiene la representación matricial de la imagen multiespectral que se quiere utilizar en el proceso de fusión

panchromatic: ***array-like***
Arreglo 2-D que contiene la representación matricial de la imagen pancromática que se quiere utilizar en el proceso de fusión

save_image: ***bool, opcional***

Si *save_image* es Falso (Por defecto), entonces no se guarda la imagen resultante del proceso de fusión. En otro caso, se realizar el almacenamiento de la imagen.

savepath: ***string, opcional***
Dirección local donde se desea almacenar la imagen. Por defecto, la imagen se guardará en la dirección donde se esté ejecutando la fusión.

timeCondition: ***bool, opcional***
Si *timeCondition* es Verdadero, durante la fusión de imágenes se calculará el tiempo de ejecución de la transformada. De otra manera, no se calculará este tiempo.

Retornos:

final_results: *dictionary*
Si *timeCondition* es Verdadero, se retornará un diccionario almacenando la imagen resultado y el tiempo de ejecución. La obtener la imagen, realiza una indexación mediante la clave “image” y para el tiempo mediante la clave “time”.

final_image: *ndarray*
Si *timeCondition* es Falso, se retornará un arreglo con la imagen resultado.

Ejemplos:

```
>>> from sallfus.fusion import pcaGPU
>>> import skimage.io
>>> m = skimage.io.imread('multispectral.tif', plugin='tifffile')
>>> p = skimage.io.imread('panchromatic.tif', plugin='tifffile')
>>> results = pcaGPU.fusion_images(m,p)
>>> results[“image”]
>>>                            [[[10 26 27]
                               [11 27 28]
                               [40 56 50]
                                   ...
                              [ 2 14  22]
                              [17 29  34]
                              [21 33  37]]
                                   ...
                             [[24 47 38]
                              [28 50 40]
                              [24 47 38]
                                   ...
                              [11 22 27]
                              [11 22 27]
                             [12  23  28]]]
>>> results[“time”]
>>> 3.9373838901519775
```

3.1.7 sallfus.fusion.atrousCPU

3.1.7.1 fusion_images

Realiza la fusión de imágenes satelitales mediante la transformada À Trous, realizando su procesamiento en una arquitectura serial haciendo uso de la CPU.

sallfus.fusion.atrousCPU.fusion_images*(multispectral, panchromatic, save_image= False , savepath=None, timeCondition=True)*

Parámetros:

Multispectral: ***array-like***
Arreglo 2-D que contiene la representación matricial de la imagen multiespectral que se quiere utilizar en el proceso de fusión

panchromatic: ***array-like***
Arreglo 2-D que contiene la representación matricial de la imagen pancromática que se quiere utilizar en el proceso de fusión

save_image: ***bool, opcional***
Si *save_image* es Falso (Por defecto), entonces no se guarda la imagen resultante del proceso de fusión. En otro caso, se realizar el almacenamiento de la imagen.

savepath: ***string, opcional***
Dirección local donde se desea almacenar la imagen. Por defecto, la imagen se guardará en la dirección donde se esté ejecutando la fusión.

timeCondition: ***bool, opcional***
Si *timeCondition* es Verdadero, durante la fusión de imágenes se calculará el tiempo de ejecución de la transformada. De otra manera, no se calculará este tiempo.

Retornos:

final_results: *dictionary*
Si *timeCondition* es Verdadero, se retornará un diccionario almacenando la imagen resultado y el tiempo de ejecución. La obtener la imagen, realiza una indexación mediante la clave “image” y para el tiempo mediante la clave “time”.

final_image: *ndarray*
Si *timeCondition* es Falso, se retornará un arreglo con la imagen resultado.

Ejemplos:

```
>>> from sallfus.fusion import atrousCPU
>>> import skimage.io
>>> m = skimage.io.imread('multispectral.tif', plugin='tifffile')
>>> p = skimage.io.imread('panchromatic.tif', plugin='tifffile')
>>> results = atrpusCPU.fusion_images(m,p)
>>> results[“image”]
```

```
>>>                                [[[25 36 32]
                                    [27 38 34]
                                    [60 84 75]
                                       ...
                                    [15 20 21]
                                    [33 45 46]
                                    [37 50 52]]
                                       ...
                                   [[36 63 51]
                                    [39 68 55]
                                    [36 63 51]
                                       ...
                                    [25 34 33]
                                    [25 34 33]
                                   [27 36 35]]]
>>> results[“time”]
>>> 1.7968096733093262
```

3.1.8 sallfus.fusion.atrousGPU

3.1.8.1 fusion_images

Realiza la fusión de imágenes satelitales mediante la transformada À Trous , realizando su procesamiento en una arquitectura heterogénea de forma paralela haciendo uso de la CPU y la GPU.

sallfus.fusion.atrousGPU.fusion_images*(multispectral, panchromatic, save_image= False , savepath=None, timeCondition=True)*

Parámetros:

multispectral: ***array-like***
Arreglo 2-D que contiene la representación matricial de la imagen multiespectral que se quiere utilizar en el proceso de fusión

panchromatic: ***array-like***
Arreglo 2-D que contiene la representación matricial de la imagen pancromática que se quiere utilizar en el proceso de fusión

save_image: ***bool, opcional***
Si *save_image* es Falso (Por defecto), entonces no se guarda la imagen resultante del proceso de fusión. En otro caso, se realizar el almacenamiento de la imagen.

savepath: ***string, opcional***
Dirección local donde se desea almacenar la imagen. Por defecto, la imagen se guardará en la dirección donde se esté ejecutando la fusión.

timeCondition: ***bool, opcional***
Si *timeCondition* es Verdadero, durante la fusión de imágenes se calculará el tiempo de ejecución de la transformada. De otra manera, no se calculará este tiempo.

Retornos: **final_results:** *dictionary*
Si *timeCondition* es Verdadero, se retornará un diccionario almacenando la imagen resultado y el tiempo de ejecución. La obtener la imagen, realiza una indexación mediante la clave “image” y para el tiempo mediante la clave “time”.

final_image: *ndarray*
Si *timeCondition* es Falso, se retornará un arreglo con la imagen resultado.

Ejemplos:

```
>>> from sallfus.fusion import atrousGPU
>>> import skimage.io
>>> m = skimage.io.imread('multispectral.tif', plugin='tifffile')
>>> p = skimage.io.imread('panchromatic.tif', plugin='tifffile')
>>> results = atrpusGPU.fusion_images(m,p)
>>> results[“image”]
>>>                          [[[25 36 32]
                             [27 38 34]
                             [60 84 75]
                                ...
                             [15 20 21]
                             [33 45 46]
                             [37 50 52]]
                                ...
                            [[36 63 51]
                             [39 68 55]
                             [36 63 51]
                                ...
                             [25 34 33]
                             [25 34 33]
                            [27 36 35]]]
>>> results[“time”]
>>> 1.278876543045044
```

3.1.9 sallfus.measures

3.1.9.1 mse

Calcula el Error Cuadrado Promedio (MSE), entre dos imágenes.

sallfus.measures.mse*(fusioned, original)*

Parámetros:

fusioned: ***array-like***
Arreglo 2-D que contiene la representación matricial de la imagen fusionada a la cual se le quiere calcular su ***mse*** a cada una de sus bandas.

original: ***array-like***

Arreglo 2-D que contiene la representación matricial de la imagen original que va a servir como punto de referencia en el cálculo de este índice.

Retornos: **array_mse:** *ndarray*
Arreglo que contiene el valor de ***mse*** calculado a partir de una imagen fusionada y una imagen de referencia.

Ejemplos:

```
>>> from sallfus.measures import mse
>>> import skimage.io
>>> fus = skimage.io.imread('fusioned.tif', plugin='tifffile')
>>> ori = skimage.io.imread('original.tif', plugin='tifffile')
>>> results = mse(fus,ori)
>>> results
>>> [1041.97027779  411.49507332  686.37561035]
```

3.1.9.2 rmse

Calcula el Error Cuadrado Promedio (RMSE), entre dos imágenes.

sallfus.measures.rmse*(fusioned, original)*

Parámetros:

fusioned: ***array-like***
Arreglo 2-D que contiene la representación matricial de la imagen fusionada a la cual se le quiere calcular su ***rmse*** a cada una de sus bandas.

original: ***array-like***
Arreglo 2-D que contiene la representación matricial de la imagen original que va a servir como punto de referencia en el cálculo de este índice.

Retornos: **array_mse:** *ndarray*
Arreglo que contiene el valor de ***rmse*** calculado a partir de una imagen fusionada y una imagen de referencia.

Ejemplos:

```
>>> from sallfus.measures import rmse
>>> import skimage.io
>>> fus = skimage.io.imread('fusioned.tif', plugin='tifffile')
>>> ori = skimage.io.imread('original.tif', plugin='tifffile')
>>> results = rmse(fus,ori)
>>> results
>>> [32.2795644  20.28534134 26.19877116]
```

3.1.9.3 bias

Calcula el coeficiente de Bias, entre dos imágenes.

sallfus.measures.bias*(fusioned, original)*

Parámetros:

fusioned: ***array-like***
Arreglo 2-D que contiene la representación matricial de la imagen multiespectral que se quiere utilizar en el proceso de fusión

original: ***array-like***
Arreglo 2-D que contiene la representación matricial de la imagen original que va a servir como punto de referencia en el cálculo de este índice.

Retornos:

array_mse: *ndarray*
Arreglo que contiene el valor de ***bias*** calculado a partir de una imagen fusionada y una imagen de referencia.

Ejemplos:

```
>>> from sallfus.measures import bias
>>> import skimage.io
>>> fus = skimage.io.imread('fusioned.tif', plugin='tifffile')
>>> ori = skimage.io.imread('original.tif', plugin='tifffile')
>>> results = bias(fus,ori)
>>> results
>>> [0.5392358459738131, 0.32604520967818296, 0.4104577206944886]
```

3.1.9.4 correlation_coeff

Calcula el coeficiente de correlación, entre dos imágenes.

sallfus.measures.correlation_coeff*(fusioned, original)*

Parámetros:

fusioned: ***array-like***
Arreglo 2-D que contiene la representación matricial de la imagen fusionada a la cual se le quiere calcular su ***coeficiente de correlación*** a cada una de sus bandas.

original: ***array-like***
Arreglo 2-D que contiene la representación matricial de la imagen original que va a servir como punto de referencia en el cálculo de este índice.

Retornos:

array_mse: *ndarray*
Arreglo que contiene el valor de ***coeficiente de correlación*** calculado a partir de una imagen fusionada y una imagen de referencia.

Ejemplos:

```
>>> from sallfus.measures import bias
>>> import skimage.io
>>> fus = skimage.io.imread('fusioned.tif', plugin='tifffile')
>>> ori = skimage.io.imread('original.tif', plugin='tifffile')
>>> results = bias(fus,ori)
>>> results
>>> [0.944787313839959, 0.9706452337015585, 0.9161393083166899]
```

3.1.10 sallfus.comparison

3.1.10.1 time_comparison

Realiza la comparación de tiempos de ejecución empleados para llevar a cabo la fusión de imágenes satelitales mediante una técnica en específica tanto en CPU como en GPU.

sallfus.comparison.time_comparison*(multispectral, panchromatic, method)*

Parámetros:

Multispectral: ***array-like***
Arreglo 2-D que contiene la representación matricial de la imagen multiespectral que se quiere utilizar en el proceso de fusión.

panchromatic: ***array-like***
Arreglo 2-D que contiene la representación matricial de la imagen pancromática que se quiere utilizar en el proceso de fusión.

method: ***String***
Cadena que especifica cuál método de fusión de imágenes satelitales se desea comparar. Este parámetro puede tomar los siguientes valores: *'brovey', 'multiplicative', 'pca' y 'atrous'*.

Retornos:

results: *dictionary*
Diccionario que contiene como claves el tipo de técnica y la arquitectura utilizada para su procesamiento. Este diccionario, contiene los tiempos de ejecución del método seleccionado.

Ejemplos:

```
>>> from sallfus.comparison import time_comparison
>>> import skimage.io
>>> m = skimage.io.imread('multispectral.tif', plugin='tifffile')
>>> p = skimage.io.imread('panchromatic.tif', plugin='tifffile')
>>> results = time_comparison(m,p,'multiplicative')
>>> results
>>> {'multiplicative_cpu': 9.2219398021698, 'multiplicative_gpu':
      1.1091442108154297}
```

3.1.10.2 time_comparison_multiple

Realiza la comparación de tiempos de ejecución empleados para llevar a cabo la fusión de imágenes satelitales mediante un conjunto de técnicas tanto en CPU como en GPU.

sallfus.comparison.time_comparison_multiple*(multispectral, panchromatic, methods)*

Parámetros:

multispectral: ***array-like***
Arreglo 2-D que contiene la representación matricial de la imagen multiespectral que se quiere utilizar en el proceso de fusión

panchromatic: ***array-like***
Arreglo 2-D que contiene la representación matricial de la imagen pancromática que se quiere utilizar en el proceso de fusión

methods: ***list***
Lista de strings, donde se especifica cuáles métodos de fusión de imágenes se desea comparar. Estos métodos se implementarán tanto en CPU como en GPU. Este parámetro puede adjuntar a la lista los siguientes valores: *'brovey', 'multiplicative', 'pca' y 'atrous'*.

Retornos:

results: *dictionary*
Diccionario que contiene como claves el tipo de técnica y la arquitectura utilizada para su procesamiento. Este diccionario, contiene los tiempos de ejecución de los métodos seleccionado.

Ejemplos:

```
>>> from sallfus.comparison import time_comparison_multiple
>>> import skimage.io
>>> m = skimage.io.imread('multispectral.tif', plugin='tifffile')
>>> p = skimage.io.imread('panchromatic.tif', plugin='tifffile')
>>> results = time_comparison_multiple(m,p,['brovey','multiplicative'])
>>> results
>>> {'broveyCPU': 25.566669702529907, 'broveyGPU': 1.6562767028808594,
'multiplicativeCPU':      11.56865406036377,      'multiplicativeGPU':
0.6037395000457764}
```

3.1.10.3 measures_comparison

Realiza la comparación de tiempos de ejecución empleados para llevar a cabo la fusión de imágenes satelitales mediante un conjunto de técnicas tanto en CPU como en GPU.

sallfus.comparison.measures_comparison*(fusioned, original, measures)*

Parámetros:

fusioned: ***array-like***

Arreglo 2-D que contiene la representación matricial de la imagen fusionada a la cual se le quiere calcular los índices matemático-estadísticos a cada una de sus bandas.

original: ***array-like***
Arreglo 2-D que contiene la representación matricial de la imagen original que va a servir como punto de referencia en el cálculo de estos índices.

measures: ***list***
Lista de strings, donde se especifica cuáles índices matemático-estadísticos se desea calcular y comparar. Los posibles valores de la lista son: 'cc', 'mse', 'rmse' y/o 'bias'. Donde ***cc*** corresponde al coeficiente se correlación.

Retornos: **results:** *dictionary*
Diccionario que contiene como claves el índice implementado y como valores, los resultados obtenidos por cada uno de los índices.

Ejemplos:

```
>>> from sallfus.comparison import measures_comparison
>>> import skimage.io
>>> fus = skimage.io.imread('fusioned.tif', plugin='tifffile')
>>> ori = skimage.io.imread('original.tif', plugin='tifffile')
>>> results = measures_comparison(fus,ori,['cc','rmse'])
>>> results
>>> {'cc': {'R': 0.944787313839959, 'G': 0.9706452337015585, 'B': 0.9161393083166899}, 'rmse': {'R': 32.27956439895456, 'G': 20.28534134094079, 'B': 26.19877116109766}}
```