Local image co-registration

This local co-registration module of AROSICS has been designed to detect and correct geometric shifts present locally in your input image. The class `arosics.COREG_LOCAL` calculates a grid of spatial shifts with points spread over the whole overlap area of the input images. Based on this grid a correction of local shifts can be performed.

```
>>> from arosics import COREG_LOCAL

>>> im_reference = '/path/to/your/ref_image.bsq'
>>> im_target    = '/path/to/your/tgt_image.bsq'
>>> kwargs = {
>>>     'grid_res'     : 200,
>>>     'window_size'  : (64,64),
>>>     'path_out'     : 'auto',
>>>     'projectDir'   : 'my_project',
>>>     'q'            : False,
>>> }

>>> CRL = COREG_LOCAL(im_reference,im_target,**kwargs)
>>> CRL.correct_shifts()

Calculating actual data corner coordinates for reference image...
Corner coordinates of reference image:
    [[319090.0, 5790510.0], [351800.0, 5899940.0], [409790.0, 5900040.0], [409790.0, 5790250.0],
[319090.0, 5790250.0]]
Calculating actual data corner coordinates for image to be shifted...
Corner coordinates of image to be shifted:
    [[319460.0, 5790510.0], [352270.0, 5900040.0], [409790.0, 5900040.0], [409790.0, 5790250.0],
[319460.0, 5790250.0]]
Matching window position (X,Y): 372220.10753674706/5841066.947109019
Calculating tie point grid (1977 points) in mode 'multiprocessing'...
    progress: |================================================| 100.0% [1977/1977] Complete
9.75 sek
Found 1144 valid GCPs.
Correcting geometric shifts...
Translating progress |================================================| 100.0% Complete
Warping progress     |================================================| 100.0% Complete
Writing GeoArray of size (10979, 10979) to /home/gfz-
fe/scheffler/jupyter/arosics_jupyter/my_project/S2A_OPER_MSI_L1C_TL_SGS__20160608T153121_A005024_T33


OrderedDict([('band', None),
             ('is shifted', True),
             ('is resampled', True),
             ('updated map info',
              ['UTM',
               1,
               1,
               300000.0,
               5900030.0,
               10.0,
               10.0,
               33,
               'North',
               'WGS-84']),
             ('updated geotransform',
              [300000.0, 10.0, 0.0, 5900030.0, 0.0, -10.0]),
             ('updated projection',
              'PROJCS["WGS 84 / UTM zone 33N",GEOGCS["WGS 84",DATUM["WGS_1984",SPHEROID["WGS
```

```
                [   0,    0,    0, ...,  760,  769,  805],
                [   0,    0,    0, ...,  762,  755,  765],
                [   0,    0,    0, ...,    0,    0,    0]], dtype=uint16)),
             ('GeoArray_shifted',
              <geoarray.GeoArray at 0x7f451ac14a90>)])
```

detect and correct local shifts - without any disk access

All you have to do is to instanciate `arosics.COREG_LOCAL` with two instances of the `geoarray.GeoArray` class as described above.

```
>>> from geoarray import GeoArray

>>> CRL = COREG_LOCAL(GeoArray(ref_ndarray, ref_gt, ref_prj),
>>>                   GeoArray(tgt_ndarray, tgt_gt, tgt_prj),
>>>                   **kwargs)
>>> CRL.correct_shifts()
```

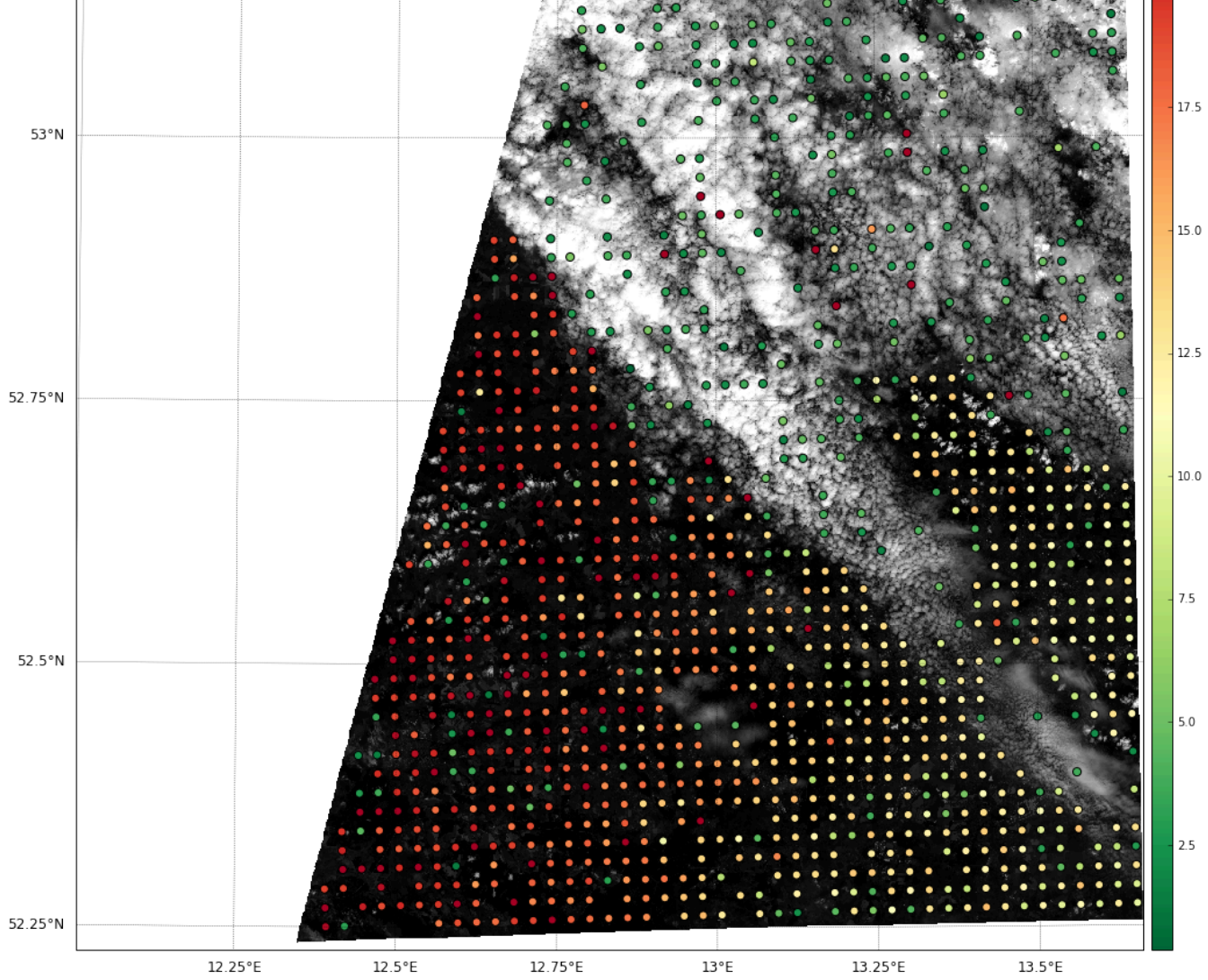
visualize tie point grid with INITIAL shifts present in your input target image

Use the method `CRL.view_CoRegPoints()` to visualize the tie point grid with the calculated absolute lenghts of the shift vectors (the unit corresponds to the input projection - UTM in the shown example, thus the unit is 'meters'.).

> **Note**
>
> A calculation of reliable shifts above cloud covered areas is not possible. In the current version of AROSICS these areas are not masked. A proper masking is planned.

```
>>> CRL.view_CoRegPoints(figsize=(15,15), backgroundIm='ref')

Note: array has been downsampled to 1000 x 1000 for faster visualization.
```

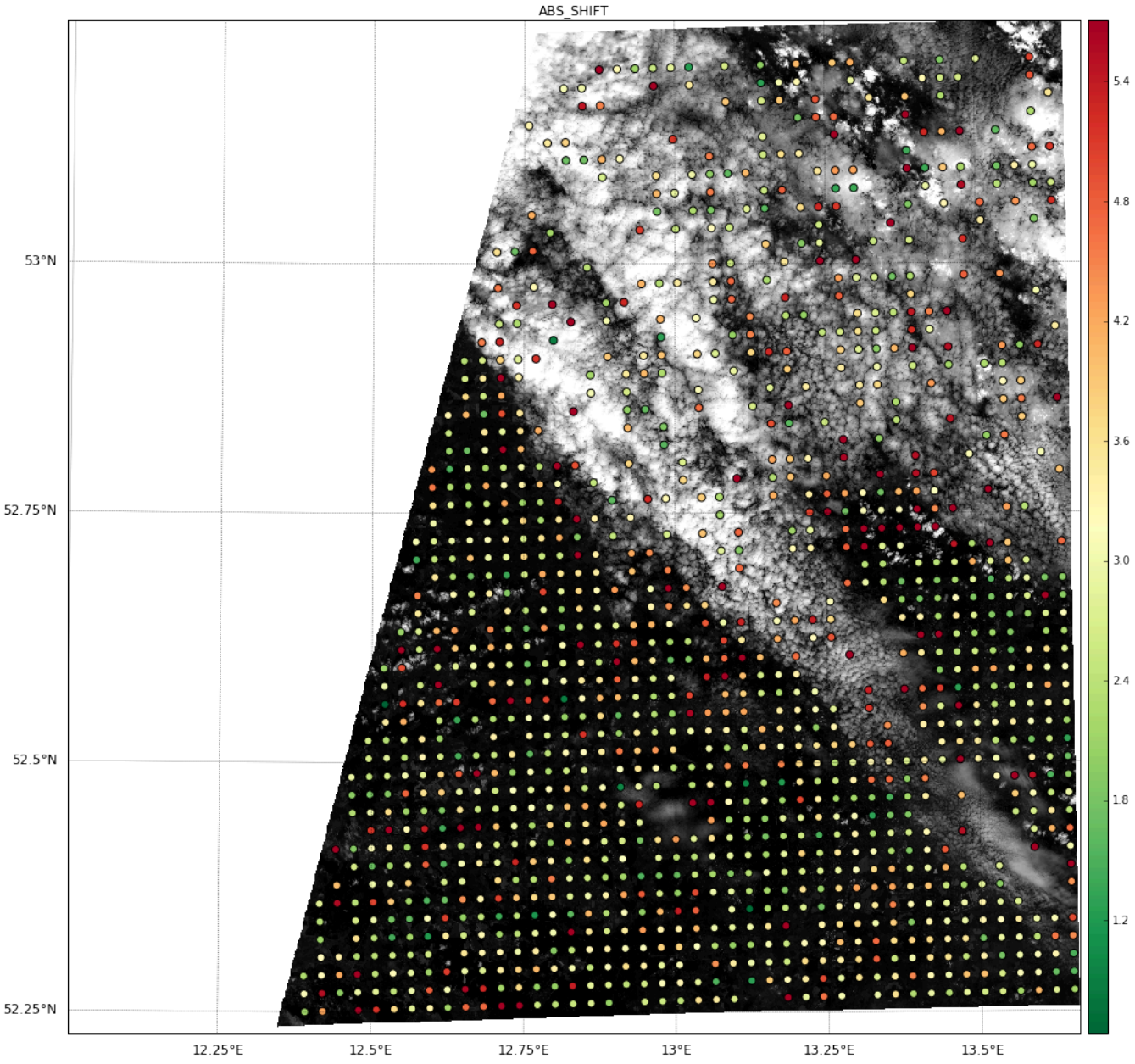


The output figure shows the calculated absolute lenghts of the shift vectors - in this case with shifts up to ~25 meters.

visualize tie point grid with shifts present AFTER shift correction

The remaining shifts after local correction can be calculated and visualized by instanciating the `arosics.COREG_LOCAL` with the output path of the above instance of `COREG_LOCAL`.

```
Corner coordinates of reference image:
    [[319090.0, 5790510.0], [351800.0, 5899940.0], [409790.0, 5900040.0], [409790.0, 5790250.0],
[319090.0, 5790250.0]]
Calculating actual data corner coordinates for image to be shifted...
Corner coordinates of image to be shifted:
    [[319460.0, 5790540.0], [352270.0, 5900030.0], [409780.0, 5900030.0], [409780.0, 5790260.0],
[322970.0, 5790250.0], [319460.0, 5790280.0]]
Matching window position (X,Y): 372216.38593955856/5841068.390957352
Note: array has been downsampled to 1000 x 1000 for faster visualization.
Calculating tie point grid (1977 points) in mode 'multiprocessing'...
    progress: |==================================================| 100.0% [1977/1977] Complete
10.78 sek
```



The output figure shows a significant reduction of geometric shifts.

Point records where no valid match has been found are filled with -9999.

```
>>> CRL.CoRegPoints_table
```

	POINT_ID	X_IM	Y_IM	X_UTM	Y_UTM	X_WIN_SIZE	Y_WIN_SIZE	X_SHIFT_PX	Y_SHIFT_PX	X_SHIFT_M	Y_SHIFT_M	ABS_SHIFT	ANGLE
0	81	5200	200	352000.0	5898040.0	76.0	74.0	0.239249	0.146466	3.588731	-2.196988	4.207820	301.474581
2	83	5600	200	356000.0	5898040.0	512.0	388.0	-0.356977	0.373230	-5.354648	-5.598444	7.746923	43.724911
5	86	6200	200	362000.0	5898040.0	512.0	388.0	0.157178	0.404519	2.357663	-6.067784	6.509730	338.766151
7	88	6600	200	366000.0	5898040.0	512.0	388.0	-0.459250	0.331437	-6.888751	-4.971556	8.495367	54.182323
8	89	6800	200	368000.0	5898040.0	512.0	388.0	0.354148	0.451144	5.312223	-6.767153	8.603143	321.868048
10	91	7200	200	372000.0	5898040.0	512.0	388.0	0.349729	0.180863	5.245941	-2.712940	5.905925	297.345759
18	99	8800	200	388000.0	5898040.0	512.0	388.0	-0.372647	0.455296	-5.589699	-6.829444	8.825307	39.299320
24	105	10000	200	400000.0	5898040.0	512.0	388.0	-0.044553	-0.090755	-0.668299	1.361323	1.516517	153.852716
25	106	10200	200	402000.0	5898040.0	512.0	388.0	-4.168502	-0.378703	-62.527525	5.680549	62.785031	95.191001
26	107	10400	200	404000.0	5898040.0	512.0	388.0	-0.111583	0.122994	-1.673744	-1.844907	2.491004	42.215067
27	108	10600	200	406000.0	5898040.0	510.0	388.0	-0.127243	-0.141960	-1.908644	2.129396	2.859589	138.129140
28	109	10800	200	408000.0	5898040.0	376.0	388.0	-0.206990	0.239337	-3.104850	-3.590052	4.746427	40.854831
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1975	3013	8600	10800	386000.0	5792040.0	512.0	376.0	-0.744887	1.718145	-11.173299	-25.772176	28.089992	23.438716
1976	3014	8800	10800	388000.0	5792040.0	512.0	376.0	-0.722097	1.730853	-10.831454	-25.962800	28.131608	22.645471
1977	3015	9000	10800	390000.0	5792040.0	512.0	376.0	-0.774061	1.691232	-11.610910	-25.368481	27.899339	24.593115
1978	3016	9200	10800	392000.0	5792040.0	512.0	376.0	-0.709505	1.763357	-10.642570	-26.450359	28.511152	21.917889
1979	3017	9400	10800	394000.0	5792040.0	512.0	376.0	-0.714307	1.828628	-10.714611	-27.429422	29.447853	21.336846
1980	3018	9600	10800	396000.0	5792040.0	512.0	376.0	-0.681368	2.120825	-10.220519	-31.812373	33.413860	17.810912
1981	3019	9800	10800	398000.0	5792040.0	512.0	376.0	-0.454680	1.715511	-6.820207	-25.732666	26.621145	14.844413
1982	3020	10000	10800	400000.0	5792040.0	512.0	376.0	-0.611233	1.779538	-9.168499	-26.693068	28.223771	18.956503
1983	3021	10200	10800	402000.0	5792040.0	512.0	376.0	-0.655737	1.824597	-9.836051	-27.368948	29.082765	19.767790
1984	3022	10400	10800	404000.0	5792040.0	512.0	376.0	-0.608115	1.791172	-9.121724	-26.867574	28.373797	18.752699
1985	3023	10600	10800	406000.0	5792040.0	510.0	376.0	-0.577808	1.752265	-8.667122	-26.283981	27.676103	18.249931
1986	3024	10800	10800	408000.0	5792040.0	376.0	376.0	-0.584037	1.720898	-8.760555	-25.813463	27.259534	18.746150

1398 rows × 19 columns

export tie point grid to an ESRI point shapefile

```
>>> CRL.tiepoint_grid.to_PointShapefile(path_out='/path/to/your/output_shapefile.shp')
```

Using the Shell console

Follow these instructions to run AROSICS from a shell console. For example, the most simple call for a local co-registration would look like this: