Supplement of The Cryosphere, 17, 4063–4078, 2023
https://doi.org/10.5194/tc-17-4063-2023-supplement

The Cryosphere
Open Access

Supplement of

GLAcier Feature Tracking testkit (GLAFT): a statistically and physically based framework for evaluating glacier velocity products derived from optical satellite image feature tracking

Whyjay Zheng et al.

Correspondence to: Whyjay Zheng (whyjayzheng@gmail.com)

GLAFT manual and supporting material

Whyjay Zheng et al.

Jul 10, 2023

CONTENTS

Whyjay Zheng[1,2], Shashank Bhushan[3], Maximillian Van Wyk De Vries[4,5,6], William Kochtitzky[7,8], David Shean[3], Luke Copland[7], Christine Dow[9], Renette Jones-Ivey[10], Fernando Pérez[1]

[1]University of California Berkeley, Department of Statistics, Berkeley, CA 94720-3860, USA
[2]National Central University, Center for Space and Remote Sensing Research, Zhongli, Taoyuan 320317, Taiwan
[3]University of Washington, Department of Civil & Environmental Engineering, Seattle, WA 98195, USA
[4]St Anthonys Falls laboratory, University of Minnesota, Minneapolis, MN, USA
[5]School of Environmental Sciences, University of Liverpool, Liverpool, L3 5DA, UK
[6]School of Geography and the Environment, University of Oxford, Oxford, OX1 3QY, UK
[7]Department of Geography, Environment and Geomatics, University of Ottawa, Ottawa K1N 6N5, Canada
[8]School of Marine and Environmental Programs, University of New England, Biddeford, ME 04005, USA
[9]Department of Geography and Environmental Management, University of Waterloo, Waterloo N2L 3G1, Canada
[10]University at Buffalo, Institute for Artificial Intelligence and Data Science, Buffalo, NY 14260, USA
Corresponding author and email: Whyjay Zheng (whyjayzheng@gmail.com)

Article link

The article is currectly in review, and the preprint is available in The Cryosphere Discuss. The early work of this project was presented at the AGU 2021 meeting with a poster available on ESSOAR.

Table of Content

Part 1: GLAFT user manual.

Parts 2–5: Other supplemental material for the article “GLAcier Feature Tracking testkit (GLAFT): A statistically and physically based framework for evaluating glacier velocity products derived from optical satellite image feature tracking,” including all necessary components to reproduce the presented work:

- Part 2: Supplementary tables and figures
- Part 3: Figure scripts
- Part 4: Intermediate processing steps
- Part 5: ITS_LIVE data processing scripts

Part I

GLAFT user manual

CHAPTER

ONE

INTRODUCTION

GLAcier Feature Tracking testkit (GLAFT) is a Python package for assessing and benchmarking feature-tracked glacier velocity maps derived from satellite imagery. To be compatible with as many feature-tracking tools as possible, GLAFT analyzes velocity maps (and optional reliability files used as weight) and calculates two metrics based on statistics and ice flow dynamics. Along with GLAFT's visualization tools, users can intercompare the quality of velocity maps processed by different methods and parameter sets. In the GLAFT publication, we further provide a guideline for optimizing glacier velocity maps by comparing the calculated metrics to an ideal threshold value.

GLAFT is an open sourced project and is hosted on Github (https://github.com/whyjz/GLAFT). All documentation and cloud-executable demos are deployed as Jupyter Book pages (https://whyjz.github.io/GLAFT/).

1.1 Installation

Try GLAFT without installing: We recommend running our Quick Start notebook on MyBinder.org.

For cloud access: We recommend using the Ghub portal to launch GLAFT (registration required).

For local installation: GLAFT is available on PyPI and can be installed via `pip`.

```
pip install glaft
```

1.2 License

GLAFT uses the MIT License. More information is available here.

CHAPTER

TWO

GLAFT QUICK START

GLAcier Feature Tracking testkit (GLAFT) calculates two metrics for visualizing and benchmarking the quality of glacier velocity maps. Here we describe its basic usage.

2.1 Input files

To begin, the following files are necessary:

1. **Velocity map as two files**; one showing the V_x component and the other showing V_y. GLAFT accepts any raster format readable by the Python rasterio package, and we recommend GeoTiff as the preferred format.
2. **Polygon geometries indicating static terrain or ice flow locations**. GLAFT uses the Python geopandas module to parse the geometries. While there are many compatible formats, we recommend using ESRI shapefile for these geometries since this format has been tested. The other common formats should work too, such as geopackage or geojson, but please use them with discretion as if you choose to. Polygon geometries must use the **same coordinate reference system (CRS)** as the velocity maps.

2.2 Procedure

First, import the GLAFT module,

```
import glaft
```

and specify the input velocity maps and polygon geometries. We use one of the test pairs presented in the GLAFT publication as a demo: Kaskawulsh glacier velocity between March 4 and April 5, 2018, processed with the CARST software with the following key parameters. More details are available in the paper.

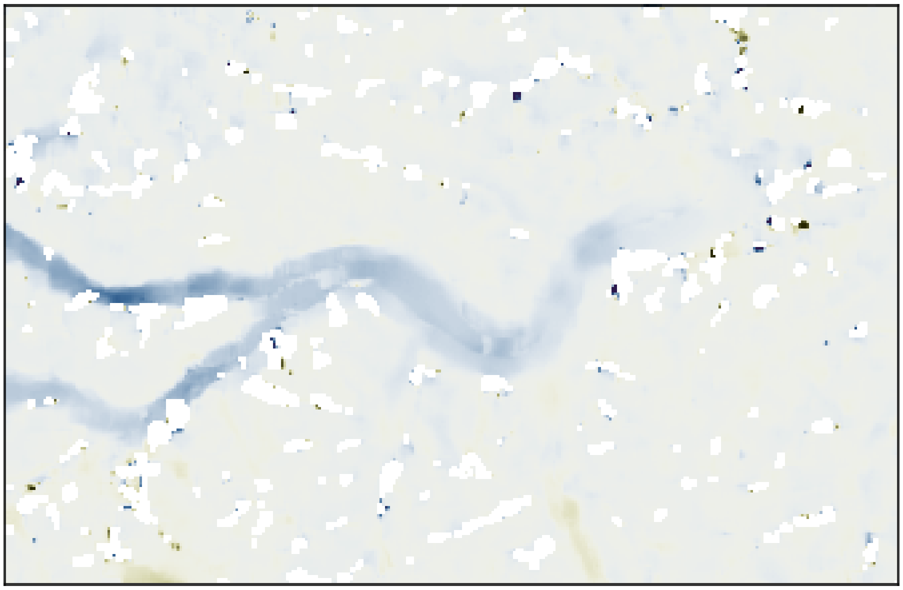
- Image source: Landsat 8
- Matching template size: 64 pixels (960 m)
- Output resolution (aka skip size): 4 pixels (60 m)
- Pre-processing filter: Gaussian filter

```
vx = 'demo-data/20180304-20180405_velo-raw_vx.tif'
vy = 'demo-data/20180304-20180405_velo-raw_vy.tif'
static_area  = 'demo-data/bedrock_V2.shp'
iceflow_area = 'demo-data/glacier_V1_Kaskawulsh_s_inwardBuffer600m.shp'
```

2.2.1 Checking input files

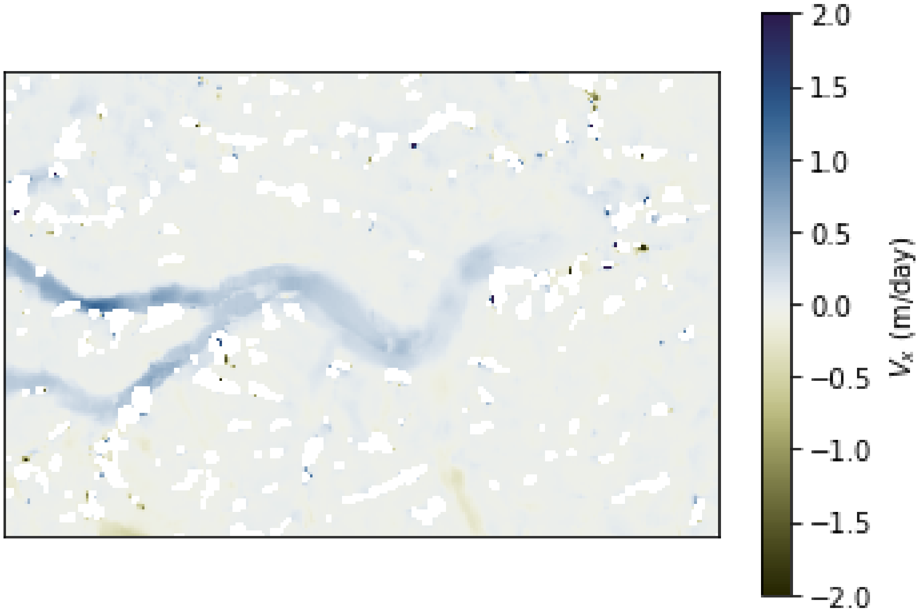
GLAFT has a function called `show_velocomp` to visualize the input raster.

```
glaft.show_velocomp(vx);
```



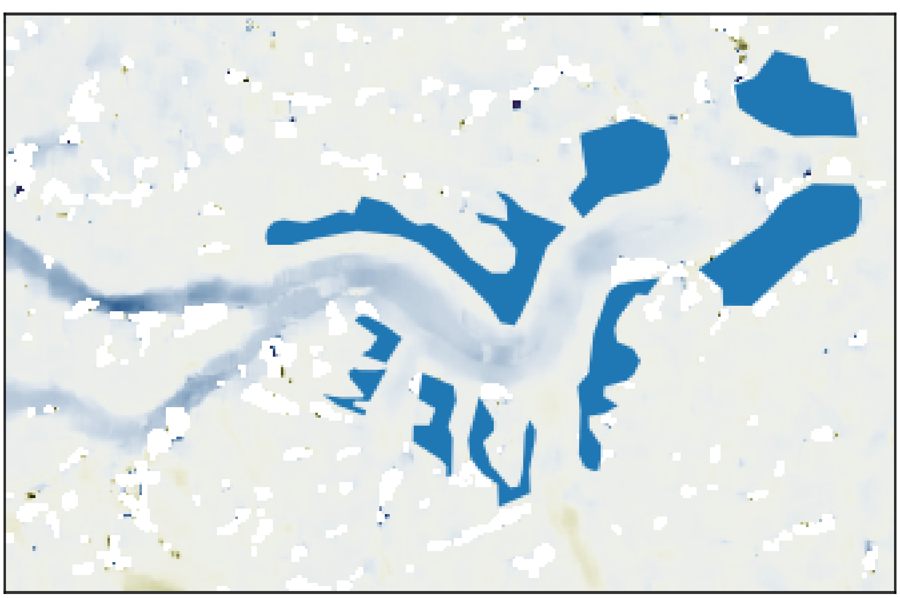
To prepare and show colorbar, we can use the `prep_colorbar_mappable` function:

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots(1, 1)
cm_settings = glaft.show_velocomp(vx, ax=ax)

mappable = glaft.prep_colorbar_mappable(**cm_settings)
fig.colorbar(mappable, label='$V_x$ (m/day)');
```



GLAFT does not provide functions to visualize and check the polygon geometries since we can simply use geopandas to achieve that.

```
import geopandas as gpd

fig, ax = plt.subplots(1, 1)
_ = glaft.show_velocomp(vx, ax=ax)

polygons = gpd.read_file(static_area)
polygons.plot(ax=ax);
```



2.2.2 Metric 1: Correct-match uncertainty on static terrains

To calculate Metric 1, we construct a `glaft.Velocity` object with all necessary files as arguments.

```
experiment = glaft.Velocity(vxfile=vx, vyfile=vy, static_area=static_area)
```

And then we can use the method `static_terrain_analysis` to perform the entire analysis. This method is essentially a wrapper script containing eleven steps for calculating kernel density estimate (KDE).

```
experiment.static_terrain_analysis()
```

```
Running clip_static_area
Running calculate_xystd
Running calculate_bandwidth
Running calculate_kde
Running construct_crude_mesh
Running eval_crude_mesh
Running construct_fine_mesh
Running eval_fine_mesh
Running thresholding_fine_mesh
Running thresholding_metric
Running cal_outlier_percent
```

Now the metric and the other derived values are accessible via the following object attributes.

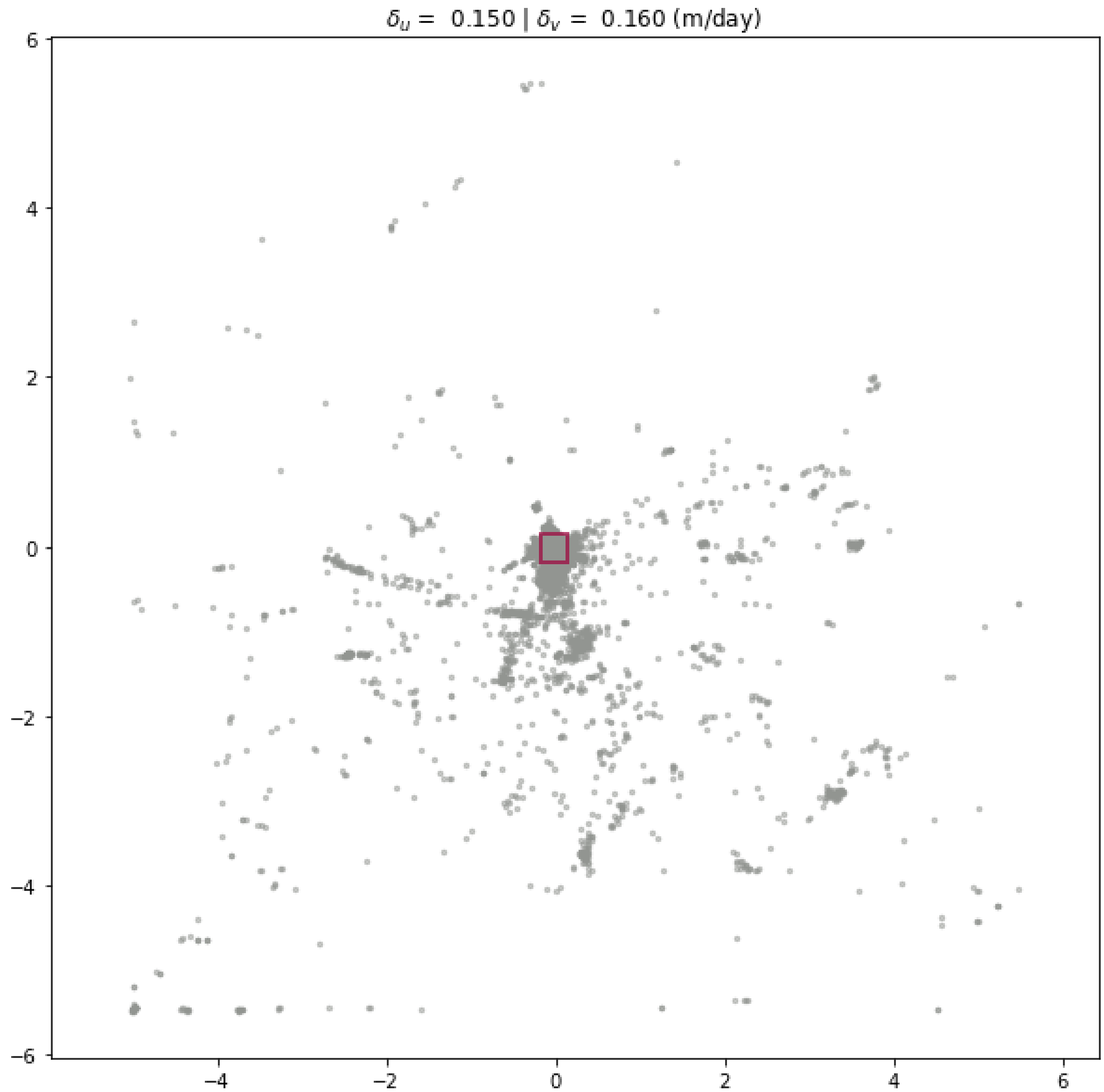
```
print('delta_x: {:.4} (m/day)'.format(experiment.metric_static_terrain_x))
print('delta_y: {:.4} (m/day)'.format(experiment.metric_static_terrain_y))
print('KDE peak location x: {:.4} (m/day)'.format(experiment.kdepeak_x))
print('KDE peak location y: {:.4} (m/day)'.format(experiment.kdepeak_y))
print('Incorrect match percentage: {:.2}%'.format(100 * experiment.outlier_percent))
```

```
delta_x: 0.1501 (m/day)
delta_y: 0.1598 (m/day)
KDE peak location x: -0.01949 (m/day)
KDE peak location y: -0.02445 (m/day)
Incorrect match percentage: 7.1%
```

There are two ways to visualize the analysis results. First, you can set the `plot` argument as either `full` or `zoomed` for the `static_terrain_analysis` method:

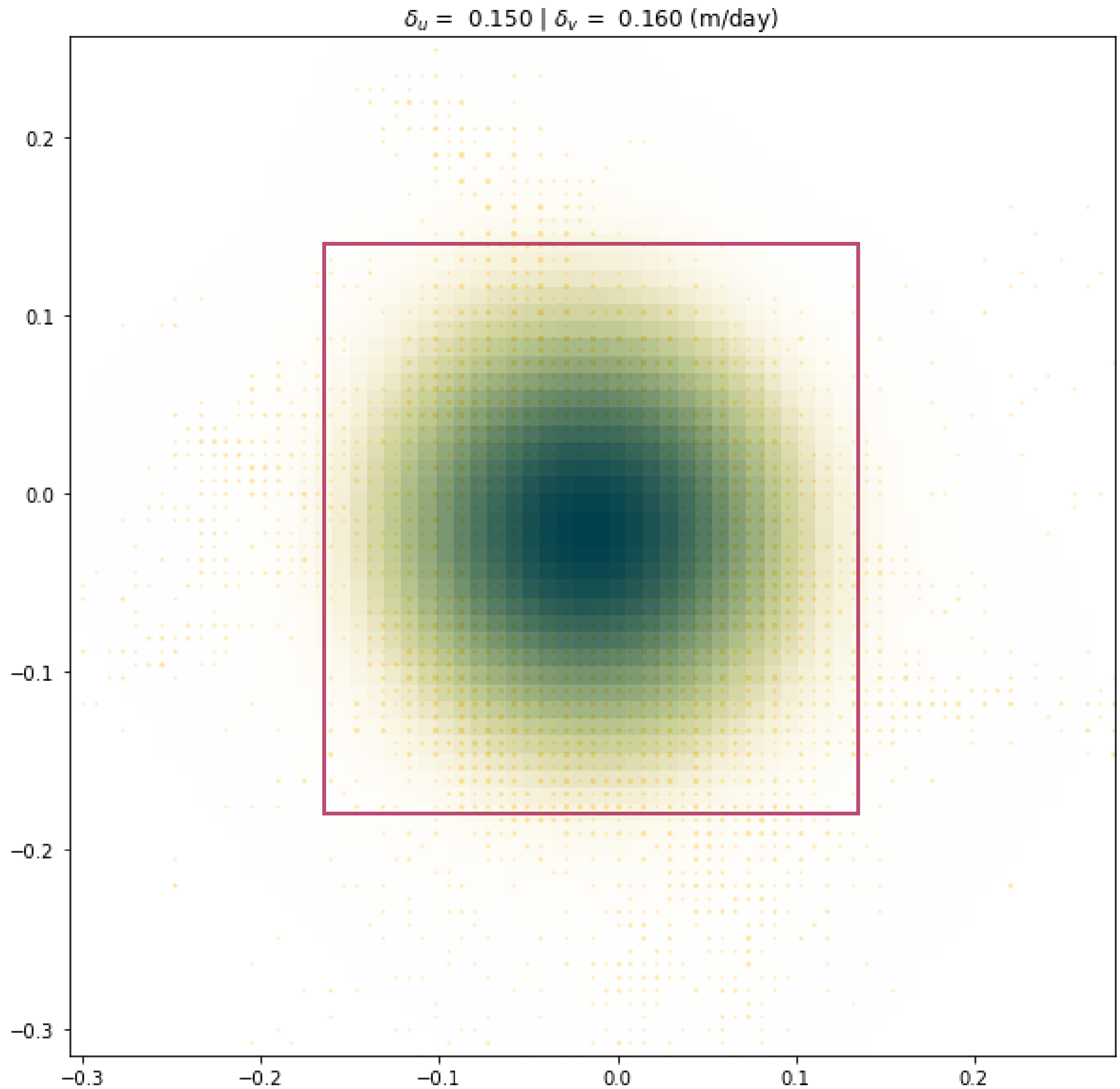
```
experiment.static_terrain_analysis(plot='full')
```

```
Running clip_static_area
Running calculate_xystd
Running calculate_bandwidth
Running calculate_kde
Running construct_crude_mesh
Running eval_crude_mesh
Running construct_fine_mesh
Running eval_fine_mesh
Running thresholding_fine_mesh
Running thresholding_metric
Running cal_outlier_percent
```



If you have performed the analysis and do not want to repeat, you can alternatively use the `plot_full_extent` or `plot_zoomed_extent` method for the same plotting functionality (with the `metric` flag set to 1).

```
experiment.plot_zoomed_extent(metric=1)
```



2.2.3 Metric 2: Along-flow strain rate variability

Again, we start by constructing a `glaft.Velocity` object. Instead of using the `static_area` argument, we pass the polygon file path to the `on_ice_area` argument.

```
experiment = glaft.Velocity(vxfile=vx, vyfile=vy, on_ice_area=iceflow_area)
```

And then we can execute the wrapper method `longitudinal_shear_analysis` to calculate Metric 2. This method contains fifteen sub-steps.

```
experiment.longitudinal_shear_analysis()
```

```
Running clip_on_ice_area
Running get_grid_spacing
```

(continues on next page)

(continued from previous page)

```
Running calculate_flow_theta
Running calculate_strain_rate
Running prep_strain_rate_kde
Running calculate_xystd
Running calculate_bandwidth
Running calculate_kde
Running construct_crude_mesh
Running eval_crude_mesh
Running construct_fine_mesh
Running eval_fine_mesh
Running thresholding_fine_mesh
Running thresholding_metric
Running cal_outlier_percent
```

Now the metric are accessible via the following object attributes. Other derived values are also available; see *reference* for detail.

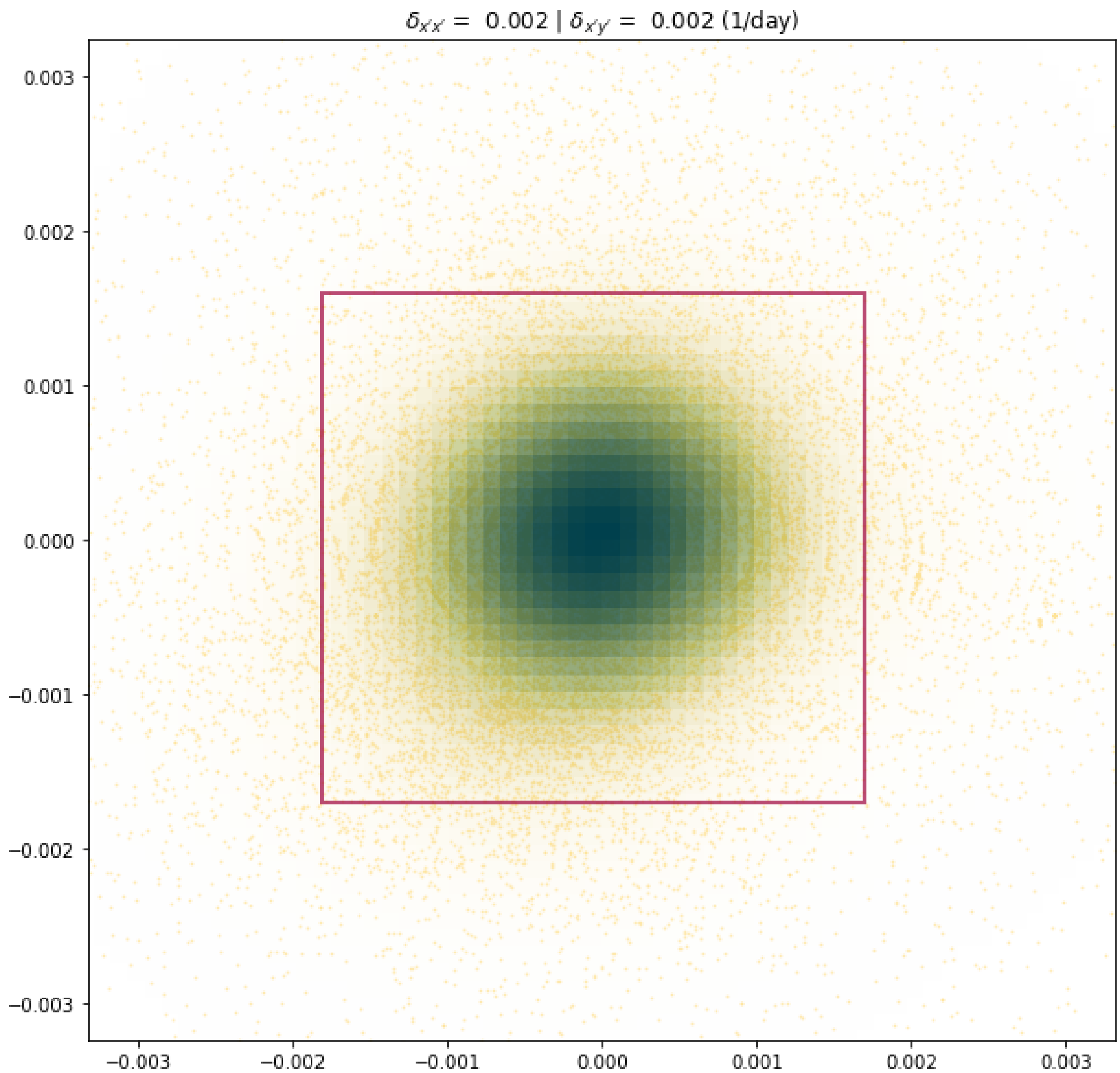
```
print("delta_x'x': {:.4} (1/day)".format(experiment.metric_alongflow_normal))
print("delta_x'y': {:.4} (1/day)".format(experiment.metric_alongflow_shear))
```

```
delta_x'x': 0.001755 (1/day)
delta_x'y': 0.001646 (1/day)
```

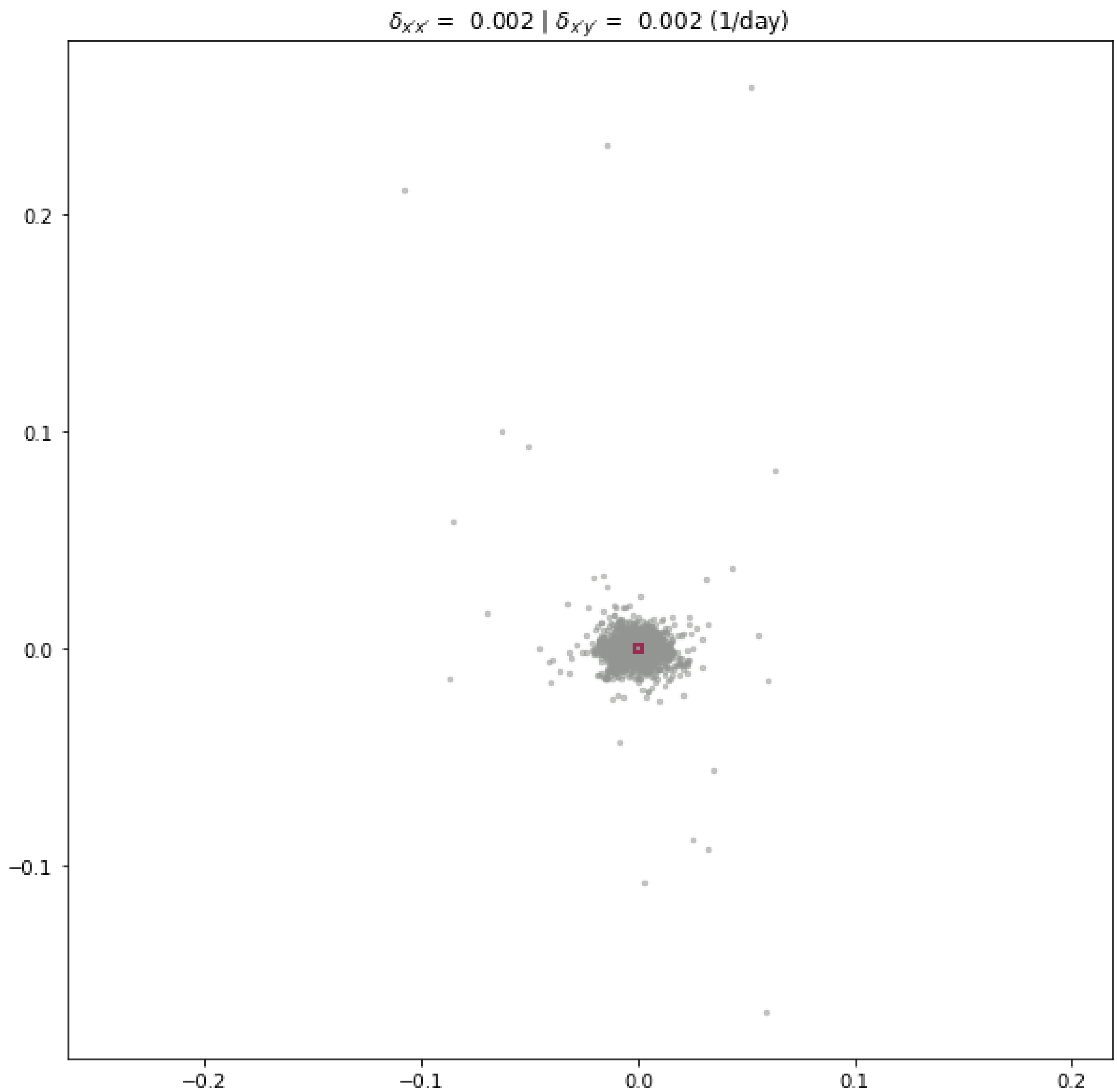
We have the same two ways to visualize the results: setting the `plot` argument for `longitudinal_shear_analysis`, or calling `plot_full_extent` or `plot_zoomed_extent` (with a `metric` argument set to `2`) after `longitudinal_shear_analysis` is executed.

```
experiment.longitudinal_shear_analysis(plot='zoomed')
```

```
Running clip_on_ice_area
Running get_grid_spacing
Running calculate_flow_theta
Running calculate_strain_rate
Running prep_strain_rate_kde
Running calculate_xystd
Running calculate_bandwidth
Running calculate_kde
Running construct_crude_mesh
Running eval_crude_mesh
Running construct_fine_mesh
Running eval_fine_mesh
Running thresholding_fine_mesh
Running thresholding_metric
Running cal_outlier_percent
```




```
experiment.plot_full_extent(metric=2)
```



For advanced settings and workflows, please see *reference page* for details.

CHAPTER

THREE

REFERENCE

3.1 glaft.Velocity class

3.1.1 glaft.Velocity(vxfile: str=None, vyfile: str=None, wfile: str=None, static_area: str=None, on_ice_area: str=None, nodata: float=-9999.0, velocity_unit: str='m/day', thres_sigma: float=2.0, kde_gridsize: int=60)

Construct an experiment to calculate velocity map benchmarking metrics.

```
vxfile:        str, Vx raster file path
                    (single band with meters as the geotransform unit)
vyfile:        str, Vy raster file path
                    (single band with meters as the geotransform unit)
wfile:         str, weight raster file path [Optional]
static_area:   str, static area (shapefile) file path
on_ice_area:   str, on-ice area (shapefile) file path
nodata:      float, nodata value in the provided geotiff. NOT FULLY IMPLEMENTED YET.
velocity_unit: str, velocity unit to be shown on the result plots.
thres_sigma: float, selected thresholding z values.
kde_gridsize:  int, grid size used for evaluating a crude KDE surface
                    (larger value means a faster but less precise process)
```

Velocity.static_terrain_analysis(plot=None, ax=None)

Perform the static terrain analysis and calculate the correct-match uncertainty.

```
plot: None for no plot;
      "full" for plotting the results in the full extent;
      "zoomed" for plotting the results in the zoomed extent.
ax:   matplotlib.axes object to place the plot.
```

Velocity.longitudinal_shear_analysis(plot=None, ax=None)

Perform the longitudinal strain rate analysis and calculate the associated metrics.

```
plot: None for no plot;
      "full" for plotting the results in the full extent;
      "zoomed" for plotting the results in the zoomed extent.
ax:   matplotlib.axes object to place the plot.
```

Velocity.plot_full_extent(ax=None, rect=None, metric: int=1, **pt_style)

Plot the analysis results in the full extent.

```
ax:       matplotlib.axes object to place the plot.
rect:     style of the rectanglar box indicating the correct match area.
          If None, the plot uses thick red line as default.
          See Velocity.create_rectangle_patch for details.
metric:   which metric to be plotted.
          1 for static terrain;
          2 for along-flow strain rate.
pt_style: point style passed to plt.scatter.
```

Velocity.plot_zoomed_extent(ax=None, rect=None, base_colormap=None, metric: int=1, **pt_style)

Plot the analysis results in the zoomed extent focusing on the correct-match area.

```
ax:       matplotlib.axes object to place the plot.
rect:     style of the rectanglar box indicating the correct match area.
          If None, the plot uses thick red line as default.
          See Velocity.create_rectangle_patch for details.
base_colormap: colormap for showing the KDE probability distribtuion.
          If None, the default (cramericm.bamako_r) is used.
metric:   which metric to be plotted.
          1 for static terrain;
          2 for along-flow strain rate.
pt_style: point style passed to plt.scatter.
```

Velocity.create_rectangle_patch(var_x, var_y, **rect_style)

Create a rectangle patch to be plotted on the visualization of the processing results.

```
var_x: half width of the rectangle
var_y: half height of the rectangle
rect_style: style of the rectangle.
            Default is {'linewidth': 2, 'edgecolor': 'xkcd:cranberry',
            'facecolor': 'none', 'alpha': 0.7}

Returns
--------------
matplotlib.patches.Rectangle object.
```

Velocity.plot_strain_map(ax=None, base_colormap=None, vmax=None)

Show the strain map. Only works after `longitudinal_shear_analysis` is performed).

```
ax:       matplotlib.axes object to place the plot.
base_colormap: colormap for showing the strain map.
          If None, the default (cramericm.bamako) is used.
vmax:     value at where color is saturated.
          If None, the saturation limit will be automatically determined.
```

Velocity.cal_invalid_pixel_percent()

Calculate the amount of invalid pixels in a velocity map.

Important attributes

- `Velocity.vxfile` Vx file path.
- `Velocity.vyfile` Vy file path.
- `Velocity.wfile` Weight file path.
- `Velocity.static_area` static area geometries file path.
- `Velocity.on_ice_area` on-ice area geometries file path.
- `Velocity.nodata` NoData value for the raster files.
- `Velocity.velocity_unit` velocity unit to be shown on the result plots.
- `Velocity.vx` 2-D, clipped vx data
- `Velocity.vy` 2-D, clipped vy data
- `Velocity.xy` = np.vstack([flattened_vx, flattened_vy]) –> (1-D)-by-2 array containing vx and vy.
- `Velocity.w` 2-D, clipped weight data
- `Velocity.w_flat` 1-D, clipped & flattened weight data
- `Velocity.dx` velocity map pixel spacing, x
- `Velocity.dy` velocity map pixel spacing, y
- `Velocity.thres_sigma` selected thresholding z values.
- `Velocity.kernel` KDE kernel (default `epanechnikov`)
- `Velocity.bandwidth` KDE bandwidth (default: calculated using the rule of thumb)
- `Velocity.kde_gridsize` grid size used for evaluating a crude KDE surface (larger value means a faster but less precise process)
- `Velocity.mesh_fine` Final KDE mesh
- `Velocity.mesh_fine_z` KDE values at the final KDE mesh vertices
- `Velocity.mesh_fine_thres_idx` boolean array showing whether a vertex falls within the correct match area
- `Velocity.metric_static_terrain_x` delta_x
- `Velocity.metric_static_terrain_y` delta_y

- `Velocity.kdepeak_x` KDE peak location x
- `Velocity.kdepeak_y` KDE peak location y
- `Velocity.outlier_percent` Incorrect match percentage (*100%)
- `Velocity.invalid_percent` Invalid pixels percentage (*100%)
- `Velocity.flow_theta` Flow direction
- `Velocity.exx` normal strain rate exx, image axis
- `Velocity.eyy` normal strain rate eyy, image axis
- `Velocity.exy` shear strain rate exy, image axis
- `Velocity.flow_exx` normal strain rate exx, flow axis (that is, ex'x')
- `Velocity.flow_eyy` normal strain rate eyy, flow axis (that is, ey'y')
- `Velocity.flow_exy` shear strain rate exy, flow axis (that is, ex'y')
- `Velocity.metric_alongflow_normal` delta_x'x'
- `Velocity.metric_alongflow_shear` delta_x'y'

3.2 Auxillary functions

3.2.1 `glaft.show_velocomp(gtiff: str, ax=None, **cm_settings)`

Preview a geotiff file (presuming a velocity map).

```
Parameters
-------------
gtiff: geotiff file path (much be single band. e.g., Velocity component, such as Vx)
ax:    matplotlib.axes object to place the plot
cm_settings: colormap settings to be used in the visualization.
       Default: {'vmin': -2, 'vmax': 2, 'cmap': cramericm.broc_r}

Returns
--------------
cm_settings: colormap settings using in the visualization.
```

3.2.2 `glaft.prep_colorbar_mappable(vmin=-2, vmax=2, cmap=cramericm.broc_r)`

Generate a mappbale object for creating a colorbar.

```
Parameters
-------------
vmin: colormap lower range
vmax: colormap upper range
cmap: colormap object as the master colormap

Returns
--------------
mappable: matplotlib.mappable object
```

3.2.3 `glaft.naof2(im)`

NAOF preprocessing filter. Translated to Python from the GIV source code. All implementation credit to Max VWDV.

MaxVWDV. (2021). MaxVWDV/glacier-image-velocimetry: Glacie Image Velocimetry (v1.0.1). Zenodo. https://doi.org/10.5281/zenodo.4904544

```
Parameters
-------------
im:       np.ma.array;  input image

Returns
--------------
naof_im:  np.ma.array;  image with NAOF filter applied
```

Part II

Supplementary tables and figures

CHAPTER

FOUR

TABLE S1: PARAMETERS OF ALL 172 TESTS

These tests consist of four satellite image pairs, each with 43 distinct parameter combinations. The machine-readable CSV file is available at `notebooks/manifest.csv`.

```
                     Date Duration (days)   Template size (px)  \
0    Sen2-20180304-20180314             10                   48
1    Sen2-20180304-20180314             10                   48
2    Sen2-20180304-20180314             10                   48
3    Sen2-20180304-20180314             10                   48
4    Sen2-20180304-20180314             10                   48
5    Sen2-20180304-20180314             10                   48
6    Sen2-20180304-20180314             10                   48
7    Sen2-20180304-20180314             10                   48
8    Sen2-20180304-20180314             10                   48
9    Sen2-20180304-20180314             10                   64
10   Sen2-20180304-20180314             10                   64
11   Sen2-20180304-20180314             10                   64
12   Sen2-20180304-20180314             10                   64
13   Sen2-20180304-20180314             10                   64
14   Sen2-20180304-20180314             10                   64
15   Sen2-20180304-20180314             10                   64
16   Sen2-20180304-20180314             10                   64
17   Sen2-20180304-20180314             10                   64
18    LS8-20180304-20180405             32                   32
19    LS8-20180304-20180405             32                   32
20    LS8-20180304-20180405             32                   32
21    LS8-20180304-20180405             32                   32
22    LS8-20180304-20180405             32                   32
23    LS8-20180304-20180405             32                   32
24    LS8-20180304-20180405             32                   32
25    LS8-20180304-20180405             32                   32
26    LS8-20180304-20180405             32                   32
27    LS8-20180304-20180405             32                   64
28    LS8-20180304-20180405             32                   64
29    LS8-20180304-20180405             32                   64
30    LS8-20180304-20180405             32                   64
31    LS8-20180304-20180405             32                   64
32    LS8-20180304-20180405             32                   64
33    LS8-20180304-20180405             32                   64
34    LS8-20180304-20180405             32                   64
35    LS8-20180304-20180405             32                   64
36   Sen2-20180508-20180627             50                   48
37   Sen2-20180508-20180627             50                   48
38   Sen2-20180508-20180627             50                   48
```

(continues on next page)

(continued from previous page)

```
39   Sen2-20180508-20180627            50                48
40   Sen2-20180508-20180627            50                48
41   Sen2-20180508-20180627            50                48
42   Sen2-20180508-20180627            50                48
43   Sen2-20180508-20180627            50                48
44   Sen2-20180508-20180627            50                48
45   Sen2-20180508-20180627            50                64
46   Sen2-20180508-20180627            50                64
47   Sen2-20180508-20180627            50                64
48   Sen2-20180508-20180627            50                64
49   Sen2-20180508-20180627            50                64
50   Sen2-20180508-20180627            50                64
51   Sen2-20180508-20180627            50                64
52   Sen2-20180508-20180627            50                64
53   Sen2-20180508-20180627            50                64
54    LS8-20180802-20180818            16                32
55    LS8-20180802-20180818            16                32
56    LS8-20180802-20180818            16                32
57    LS8-20180802-20180818            16                32
58    LS8-20180802-20180818            16                32
59    LS8-20180802-20180818            16                32
60    LS8-20180802-20180818            16                32
61    LS8-20180802-20180818            16                32
62    LS8-20180802-20180818            16                32
63    LS8-20180802-20180818            16                64
64    LS8-20180802-20180818            16                64
65    LS8-20180802-20180818            16                64
66    LS8-20180802-20180818            16                64
67    LS8-20180802-20180818            16                64
68    LS8-20180802-20180818            16                64
69    LS8-20180802-20180818            16                64
70    LS8-20180802-20180818            16                64
71    LS8-20180802-20180818            16                64
72    LS8-20180304-20180405            32  varying: multi-pass
73    LS8-20180304-20180405            32  varying: multi-pass
74    LS8-20180304-20180405            32  varying: multi-pass
75    LS8-20180304-20180405            32  varying: multi-pass
76    LS8-20180304-20180405            32  varying: multi-pass
77    LS8-20180304-20180405            32  varying: multi-pass
78    LS8-20180802-20180818            16  varying: multi-pass
79    LS8-20180802-20180818            16  varying: multi-pass
80    LS8-20180802-20180818            16  varying: multi-pass
81    LS8-20180802-20180818            16  varying: multi-pass
82    LS8-20180802-20180818            16  varying: multi-pass
83    LS8-20180802-20180818            16  varying: multi-pass
84   Sen2-20180304-20180314            10  varying: multi-pass
85   Sen2-20180304-20180314            10  varying: multi-pass
86   Sen2-20180304-20180314            10  varying: multi-pass
87   Sen2-20180304-20180314            10  varying: multi-pass
88   Sen2-20180304-20180314            10  varying: multi-pass
89   Sen2-20180304-20180314            10  varying: multi-pass
90   Sen2-20180508-20180627            50  varying: multi-pass
91   Sen2-20180508-20180627            50  varying: multi-pass
92   Sen2-20180508-20180627            50  varying: multi-pass
93   Sen2-20180508-20180627            50  varying: multi-pass
94   Sen2-20180508-20180627            50  varying: multi-pass
```

(continues on next page)

(continued from previous page)

```
95   Sen2-20180508-20180627             50  varying: multi-pass
96    LS8-20180304-20180405             32                   31
97    LS8-20180304-20180405             32                   65
98    LS8-20180802-20180818             16                   31
99    LS8-20180802-20180818             16                   65
100  Sen2-20180304-20180314             10                   31
101  Sen2-20180304-20180314             10                   65
102  Sen2-20180508-20180627             50                   31
103  Sen2-20180508-20180627             50                   65
104   LS8-20180304-20180405             32                   31
105   LS8-20180304-20180405             32                   65
106   LS8-20180802-20180818             16                   31
107   LS8-20180802-20180818             16                   65
108  Sen2-20180304-20180314             10                   31
109  Sen2-20180304-20180314             10                   65
110  Sen2-20180508-20180627             50                   31
111  Sen2-20180508-20180627             50                   65
112   LS8-20180304-20180405             32                   31
113   LS8-20180304-20180405             32                   31
114   LS8-20180304-20180405             32                   31
115   LS8-20180802-20180818             16                   31
116   LS8-20180802-20180818             16                   31
117   LS8-20180802-20180818             16                   31
118  Sen2-20180304-20180314             10                   31
119  Sen2-20180304-20180314             10                   31
120  Sen2-20180304-20180314             10                   31
121  Sen2-20180508-20180627             50                   31
122  Sen2-20180508-20180627             50                   31
123  Sen2-20180508-20180627             50                   31
124   LS8-20180304-20180405             32                   32
125   LS8-20180304-20180405             32                   32
126   LS8-20180304-20180405             32                   64
127   LS8-20180304-20180405             32                   64
128   LS8-20180304-20180405             32                   32
129   LS8-20180304-20180405             32                   32
130   LS8-20180304-20180405             32                   64
131   LS8-20180304-20180405             32                   64
132   LS8-20180304-20180405             32                   32
133   LS8-20180304-20180405             32                   32
134   LS8-20180304-20180405             32                   64
135   LS8-20180304-20180405             32                   64
136   LS8-20180802-20180818             16                   32
137   LS8-20180802-20180818             16                   32
138   LS8-20180802-20180818             16                   64
139   LS8-20180802-20180818             16                   64
140   LS8-20180802-20180818             16                   32
141   LS8-20180802-20180818             16                   32
142   LS8-20180802-20180818             16                   64
143   LS8-20180802-20180818             16                   64
144   LS8-20180802-20180818             16                   32
145   LS8-20180802-20180818             16                   32
146   LS8-20180802-20180818             16                   64
147   LS8-20180802-20180818             16                   64
148  Sen2-20180304-20180314             10                   32
149  Sen2-20180304-20180314             10                   32
150  Sen2-20180304-20180314             10                   64
```

(continues on next page)

(continued from previous page)

```
151  Sen2-20180304-20180314            10                 64
152  Sen2-20180304-20180314            10                 32
153  Sen2-20180304-20180314            10                 32
154  Sen2-20180304-20180314            10                 64
155  Sen2-20180304-20180314            10                 64
156  Sen2-20180304-20180314            10                 32
157  Sen2-20180304-20180314            10                 32
158  Sen2-20180304-20180314            10                 64
159  Sen2-20180304-20180314            10                 64
160  Sen2-20180508-20180627            50                 32
161  Sen2-20180508-20180627            50                 32
162  Sen2-20180508-20180627            50                 64
163  Sen2-20180508-20180627            50                 64
164  Sen2-20180508-20180627            50                 32
165  Sen2-20180508-20180627            50                 32
166  Sen2-20180508-20180627            50                 64
167  Sen2-20180508-20180627            50                 64
168  Sen2-20180508-20180627            50                 32
169  Sen2-20180508-20180627            50                 32
170  Sen2-20180508-20180627            50                 64
171  Sen2-20180508-20180627            50                 64

       Template size (m) Pixel spacing (px) Pixel spacing (m) Prefilter  \
0                    480                 12               120       Gau
1                    480                 12               120      NAOF
2                    480                 12               120      None
3                    480                  1                10       Gau
4                    480                  1                10      NAOF
5                    480                  1                10      None
6                    480                  4                40       Gau
7                    480                  4                40      NAOF
8                    480                  4                40      None
9                    640                 12               120       Gau
10                   640                 12               120      NAOF
11                   640                 12               120      None
12                   640                  1                10       Gau
13                   640                  1                10      NAOF
14                   640                  1                10      None
15                   640                  4                40       Gau
16                   640                  4                40      NAOF
17                   640                  4                40      None
18                   480                  1                15       Gau
19                   480                  1                15      NAOF
20                   480                  1                15      None
21                   480                  4                60       Gau
22                   480                  4                60      NAOF
23                   480                  4                60      None
24                   480                  8               120       Gau
25                   480                  8               120      NAOF
26                   480                  8               120      None
27                   960                  1                15       Gau
28                   960                  1                15      NAOF
29                   960                  1                15      None
30                   960                  4                60       Gau
31                   960                  4                60      NAOF
32                   960                  4                60      None
```

(continues on next page)

(continued from previous page)

```
33                  960                 8                 120       Gau
34                  960                 8                 120      NAOF
35                  960                 8                 120      None
36                  480                12                 120       Gau
37                  480                12                 120      NAOF
38                  480                12                 120      None
39                  480                 1                  10       Gau
40                  480                 1                  10      NAOF
41                  480                 1                  10      None
42                  480                 4                  40       Gau
43                  480                 4                  40      NAOF
44                  480                 4                  40      None
45                  640                12                 120       Gau
46                  640                12                 120      NAOF
47                  640                12                 120      None
48                  640                 1                  10       Gau
49                  640                 1                  10      NAOF
50                  640                 1                  10      None
51                  640                 4                  40       Gau
52                  640                 4                  40      NAOF
53                  640                 4                  40      None
54                  480                 1                  15       Gau
55                  480                 1                  15      NAOF
56                  480                 1                  15      None
57                  480                 4                  60       Gau
58                  480                 4                  60      NAOF
59                  480                 4                  60      None
60                  480                 8                 120       Gau
61                  480                 8                 120      NAOF
62                  480                 8                 120      None
63                  960                 1                  15       Gau
64                  960                 1                  15      NAOF
65                  960                 1                  15      None
66                  960                 4                  60       Gau
67                  960                 4                  60      NAOF
68                  960                 4                  60      None
69                  960                 8                 120       Gau
70                  960                 8                 120      NAOF
71                  960                 8                 120      None
72   varying: multi-pass            15.13               242.1      NAOF
73   varying: multi-pass            4.009               60.14      NAOF
74   varying: multi-pass            15.13               242.1       Gau
75   varying: multi-pass            4.009               60.14       Gau
76   varying: multi-pass            15.13               242.1      None
77   varying: multi-pass            4.009               60.14      None
78   varying: multi-pass            15.13               242.1      NAOF
79   varying: multi-pass            4.009               60.14      NAOF
80   varying: multi-pass            15.13               242.1       Gau
81   varying: multi-pass            4.009               60.14       Gau
82   varying: multi-pass            15.13               242.1      None
83   varying: multi-pass            4.009               60.14      None
84   varying: multi-pass            16.04               160.4      NAOF
85   varying: multi-pass            4.003               40.03      NAOF
86   varying: multi-pass            16.04               160.4       Gau
87   varying: multi-pass            4.003               40.03       Gau
88   varying: multi-pass            16.04               160.4      None
```

(continues on next page)

(continued from previous page)

89	varying: multi-pass	4.003	40.03	None
90	varying: multi-pass	16.04	160.4	NAOF
91	varying: multi-pass	4.003	40.03	NAOF
92	varying: multi-pass	16.04	160.4	Gau
93	varying: multi-pass	4.003	40.03	Gau
94	varying: multi-pass	16.04	160.4	None
95	varying: multi-pass	4.003	40.03	None
96	465	1	15	Gau
97	975	1	15	Gau
98	465	1	15	Gau
99	975	1	15	Gau
100	310	1	10	Gau
101	650	1	10	Gau
102	310	1	10	Gau
103	650	1	10	Gau
104	465	1	15	None
105	975	1	15	None
106	465	1	15	None
107	975	1	15	None
108	310	1	10	None
109	650	1	10	None
110	310	1	10	None
111	650	1	10	None
112	465	1	15	LoG
113	465	1	15	LoG
114	465	1	15	LoG
115	465	1	15	LoG
116	465	1	15	LoG
117	465	1	15	LoG
118	310	1	10	LoG
119	310	1	10	LoG
120	310	1	10	LoG
121	310	1	10	LoG
122	310	1	10	LoG
123	310	1	10	LoG
124	480	4	60	None
125	480	8	120	None
126	960	4	60	None
127	960	8	120	None
128	480	4	60	Gau
129	480	8	120	Gau
130	960	4	60	Gau
131	960	8	120	Gau
132	480	4	60	NAOF
133	480	8	120	NAOF
134	960	4	60	NAOF
135	960	8	120	NAOF
136	480	4	60	None
137	480	8	120	None
138	960	4	60	None
139	960	8	120	None
140	480	4	60	Gau
141	480	8	120	Gau
142	960	4	60	Gau
143	960	8	120	Gau
144	480	4	60	NAOF

(continues on next page)

(continued from previous page)

```
145                  480                  8               120       NAOF
146                  960                  4                60       NAOF
147                  960                  8               120       NAOF
148                  320                  4                40       None
149                  320                  8                80       None
150                  640                  4                40       None
151                  640                  8                80       None
152                  320                  4                40        Gau
153                  320                  8                80        Gau
154                  640                  4                40        Gau
155                  640                  8                80        Gau
156                  320                  4                40       NAOF
157                  320                  8                80       NAOF
158                  640                  4                40       NAOF
159                  640                  8                80       NAOF
160                  320                  4                40       None
161                  320                  8                80       None
162                  640                  4                40       None
163                  640                  8                80       None
164                  320                  4                40        Gau
165                  320                  8                80        Gau
166                  640                  4                40        Gau
167                  640                  8                80        Gau
168                  320                  4                40       NAOF
169                  320                  8                80       NAOF
170                  640                  4                40       NAOF
171                  640                  8                80       NAOF

                 Subpixel  Software
0     16-node oversampling     CARST
1     16-node oversampling     CARST
2     16-node oversampling     CARST
3     16-node oversampling     CARST
4     16-node oversampling     CARST
5     16-node oversampling     CARST
6     16-node oversampling     CARST
7     16-node oversampling     CARST
8     16-node oversampling     CARST
9     16-node oversampling     CARST
10    16-node oversampling     CARST
11    16-node oversampling     CARST
12    16-node oversampling     CARST
13    16-node oversampling     CARST
14    16-node oversampling     CARST
15    16-node oversampling     CARST
16    16-node oversampling     CARST
17    16-node oversampling     CARST
18    16-node oversampling     CARST
19    16-node oversampling     CARST
20    16-node oversampling     CARST
21    16-node oversampling     CARST
22    16-node oversampling     CARST
23    16-node oversampling     CARST
24    16-node oversampling     CARST
25    16-node oversampling     CARST
26    16-node oversampling     CARST
```

(continues on next page)

(continued from previous page)

```
27   16-node oversampling     CARST
28   16-node oversampling     CARST
29   16-node oversampling     CARST
30   16-node oversampling     CARST
31   16-node oversampling     CARST
32   16-node oversampling     CARST
33   16-node oversampling     CARST
34   16-node oversampling     CARST
35   16-node oversampling     CARST
36   16-node oversampling     CARST
37   16-node oversampling     CARST
38   16-node oversampling     CARST
39   16-node oversampling     CARST
40   16-node oversampling     CARST
41   16-node oversampling     CARST
42   16-node oversampling     CARST
43   16-node oversampling     CARST
44   16-node oversampling     CARST
45   16-node oversampling     CARST
46   16-node oversampling     CARST
47   16-node oversampling     CARST
48   16-node oversampling     CARST
49   16-node oversampling     CARST
50   16-node oversampling     CARST
51   16-node oversampling     CARST
52   16-node oversampling     CARST
53   16-node oversampling     CARST
54   16-node oversampling     CARST
55   16-node oversampling     CARST
56   16-node oversampling     CARST
57   16-node oversampling     CARST
58   16-node oversampling     CARST
59   16-node oversampling     CARST
60   16-node oversampling     CARST
61   16-node oversampling     CARST
62   16-node oversampling     CARST
63   16-node oversampling     CARST
64   16-node oversampling     CARST
65   16-node oversampling     CARST
66   16-node oversampling     CARST
67   16-node oversampling     CARST
68   16-node oversampling     CARST
69   16-node oversampling     CARST
70   16-node oversampling     CARST
71   16-node oversampling     CARST
72  interest point groups       GIV
73  interest point groups       GIV
74  interest point groups       GIV
75  interest point groups       GIV
76  interest point groups       GIV
77  interest point groups       GIV
78  interest point groups       GIV
79  interest point groups       GIV
80  interest point groups       GIV
81  interest point groups       GIV
82  interest point groups       GIV
```

(continues on next page)

(continued from previous page)

```
83   interest point groups       GIV
84   interest point groups       GIV
85   interest point groups       GIV
86   interest point groups       GIV
87   interest point groups       GIV
88   interest point groups       GIV
89   interest point groups       GIV
90   interest point groups       GIV
91   interest point groups       GIV
92   interest point groups       GIV
93   interest point groups       GIV
94   interest point groups       GIV
95   interest point groups       GIV
96               parabolic      Vmap
97               parabolic      Vmap
98               parabolic      Vmap
99               parabolic      Vmap
100              parabolic      Vmap
101              parabolic      Vmap
102              parabolic      Vmap
103              parabolic      Vmap
104              parabolic      Vmap
105              parabolic      Vmap
106              parabolic      Vmap
107              parabolic      Vmap
108              parabolic      Vmap
109              parabolic      Vmap
110              parabolic      Vmap
111              parabolic      Vmap
112              parabolic      Vmap
113        affine adaptive      Vmap
114                 affine      Vmap
115              parabolic      Vmap
116        affine adaptive      Vmap
117                 affine      Vmap
118              parabolic      Vmap
119        affine adaptive      Vmap
120                 affine      Vmap
121              parabolic      Vmap
122        affine adaptive      Vmap
123                 affine      Vmap
124                  pyrUP  autoRIFT
125                  pyrUP  autoRIFT
126                  pyrUP  autoRIFT
127                  pyrUP  autoRIFT
128                  pyrUP  autoRIFT
129                  pyrUP  autoRIFT
130                  pyrUP  autoRIFT
131                  pyrUP  autoRIFT
132                  pyrUP  autoRIFT
133                  pyrUP  autoRIFT
134                  pyrUP  autoRIFT
135                  pyrUP  autoRIFT
136                  pyrUP  autoRIFT
137                  pyrUP  autoRIFT
138                  pyrUP  autoRIFT
```

(continues on next page)

(continued from previous page)

```
139                    pyrUP  autoRIFT
140                    pyrUP  autoRIFT
141                    pyrUP  autoRIFT
142                    pyrUP  autoRIFT
143                    pyrUP  autoRIFT
144                    pyrUP  autoRIFT
145                    pyrUP  autoRIFT
146                    pyrUP  autoRIFT
147                    pyrUP  autoRIFT
148                    pyrUP  autoRIFT
149                    pyrUP  autoRIFT
150                    pyrUP  autoRIFT
151                    pyrUP  autoRIFT
152                    pyrUP  autoRIFT
153                    pyrUP  autoRIFT
154                    pyrUP  autoRIFT
155                    pyrUP  autoRIFT
156                    pyrUP  autoRIFT
157                    pyrUP  autoRIFT
158                    pyrUP  autoRIFT
159                    pyrUP  autoRIFT
160                    pyrUP  autoRIFT
161                    pyrUP  autoRIFT
162                    pyrUP  autoRIFT
163                    pyrUP  autoRIFT
164                    pyrUP  autoRIFT
165                    pyrUP  autoRIFT
166                    pyrUP  autoRIFT
167                    pyrUP  autoRIFT
168                    pyrUP  autoRIFT
169                    pyrUP  autoRIFT
170                    pyrUP  autoRIFT
171                    pyrUP  autoRIFT
```

4.1 Abbreviations in Table S1

- LS8: Landsat 8
- Sen2: Sentinel-2
- px: pixels
- Gau: Gaussian high-pass filter
- NAOF: Near anisotropic orientation filter
- LoG: Laplacian of Gaussian filter
- Subpixel: Sub-pixel matching method
- pyrUP: Laplacian pyramid method

CHAPTER

FIVE

TABLE S2: METRICS AND OTHER RESULTS FOR ALL 172 TESTS

The machine-readable CSV file is available at `notebooks/results_2022.csv`.

```
                       Date  SAV-uncertainty-x  SAV-uncertainty-y  SAV-peak-x  \
0    Sen2-20180304-20180314             0.3156             0.3156     -0.0808
1    Sen2-20180304-20180314             0.2472             0.2554     -0.0707
2    Sen2-20180304-20180314             0.4887             0.4887     -0.0470
3    Sen2-20180304-20180314             0.1947             0.2058     -0.0792
4    Sen2-20180304-20180314             0.1596             0.1729     -0.0669
5    Sen2-20180304-20180314             0.2770             0.2967     -0.0378
6    Sen2-20180304-20180314             0.2537             0.2690     -0.0702
7    Sen2-20180304-20180314             0.2291             0.2358     -0.0692
8    Sen2-20180304-20180314             0.3824             0.4030     -0.0416
9    Sen2-20180304-20180314             0.2567             0.2738     -0.0711
10   Sen2-20180304-20180314             0.1601             0.1704     -0.0677
11   Sen2-20180304-20180314             0.3620             0.3981     -0.0504
12   Sen2-20180304-20180314             0.1661             0.1775     -0.0682
13   Sen2-20180304-20180314             0.1117             0.1229     -0.0588
14   Sen2-20180304-20180314             0.2387             0.2611     -0.0700
15   Sen2-20180304-20180314             0.2154             0.2281     -0.0688
16   Sen2-20180304-20180314             0.1473             0.1559     -0.0668
17   Sen2-20180304-20180314             0.2970             0.3258     -0.0529
18    LS8-20180304-20180405             0.1656             0.1763     -0.0240
19    LS8-20180304-20180405             0.1582             0.1619     -0.0184
20    LS8-20180304-20180405             0.2371             0.2371     -0.0079
21    LS8-20180304-20180405             0.2133             0.2204     -0.0149
22    LS8-20180304-20180405             0.2146             0.2146     -0.0200
23    LS8-20180304-20180405             0.2812             0.2902     -0.0056
24    LS8-20180304-20180405             0.2526             0.2608     -0.0138
25    LS8-20180304-20180405             0.2453             0.2523     -0.0217
26    LS8-20180304-20180405             0.3126             0.3126     -0.0049
27    LS8-20180304-20180405             0.1136             0.1287     -0.0184
28    LS8-20180304-20180405             0.0568             0.0639     -0.0164
29    LS8-20180304-20180405             0.1722             0.2009     -0.0089
30    LS8-20180304-20180405             0.1501             0.1598     -0.0195
31    LS8-20180304-20180405             0.0722             0.0767     -0.0169
32    LS8-20180304-20180405             0.2066             0.2273     -0.0004
33    LS8-20180304-20180405             0.1690             0.1806     -0.0125
34    LS8-20180304-20180405             0.0720             0.0791     -0.0170
35    LS8-20180304-20180405             0.2291             0.2443      0.0003
36   Sen2-20180508-20180627             0.3024             0.3122      0.0449
37   Sen2-20180508-20180627             0.3236             0.3347      0.0487
38   Sen2-20180508-20180627             0.2146             0.2590      0.0527
39   Sen2-20180508-20180627             0.1590             0.1749      0.0523
40   Sen2-20180508-20180627             0.1650             0.1694      0.0420
```

(continues on next page)

(continued from previous page)

41	Sen2-20180508-20180627	0.1536	0.2119	0.0490
42	Sen2-20180508-20180627	0.2358	0.2420	0.0500
43	Sen2-20180508-20180627	0.2561	0.2634	0.0448
44	Sen2-20180508-20180627	0.1883	0.2354	0.0505
45	Sen2-20180508-20180627	0.2800	0.2800	0.0457
46	Sen2-20180508-20180627	0.2855	0.2855	0.0288
47	Sen2-20180508-20180627	0.1752	0.2206	0.0581
48	Sen2-20180508-20180627	0.1502	0.1652	0.0525
49	Sen2-20180508-20180627	0.1466	0.1505	0.0415
50	Sen2-20180508-20180627	0.1223	0.1741	0.0547
51	Sen2-20180508-20180627	0.2223	0.2281	0.0558
52	Sen2-20180508-20180627	0.2274	0.2274	0.0440
53	Sen2-20180508-20180627	0.1529	0.2058	0.0559
54	LS8-20180802-20180818	0.0899	0.1001	0.0509
55	LS8-20180802-20180818	0.0907	0.0907	0.0369
56	LS8-20180802-20180818	0.1235	0.1354	0.0546
57	LS8-20180802-20180818	0.1174	0.1241	0.0485
58	LS8-20180802-20180818	0.1020	0.1049	0.0378
59	LS8-20180802-20180818	0.1539	0.1635	0.0538
60	LS8-20180802-20180818	0.1371	0.1456	0.0457
61	LS8-20180802-20180818	0.1222	0.1261	0.0406
62	LS8-20180802-20180818	0.1685	0.1797	0.0530
63	LS8-20180802-20180818	0.0485	0.0616	0.0605
64	LS8-20180802-20180818	0.0373	0.0488	0.0279
65	LS8-20180802-20180818	0.0781	0.0893	0.0614
66	LS8-20180802-20180818	0.0496	0.0530	0.0569
67	LS8-20180802-20180818	0.0376	0.0496	0.0308
68	LS8-20180802-20180818	0.0818	0.0940	0.0495
69	LS8-20180802-20180818	0.0517	0.0636	0.0566
70	LS8-20180802-20180818	0.0413	0.0490	0.0584
71	LS8-20180802-20180818	0.0841	0.1035	0.0618
72	LS8-20180304-20180405	0.2434	0.2769	-0.0104
73	LS8-20180304-20180405	0.2881	0.3579	-0.0165
74	LS8-20180304-20180405	0.3702	0.3702	0.0038
75	LS8-20180304-20180405	2.1477	2.2028	0.0474
76	LS8-20180304-20180405	0.8249	0.8249	0.0248
77	LS8-20180304-20180405	1.0560	1.1164	0.0277
78	LS8-20180802-20180818	0.0927	0.1098	0.0672
79	LS8-20180802-20180818	0.1880	0.1945	0.0306
80	LS8-20180802-20180818	0.0737	0.0762	0.0298
81	LS8-20180802-20180818	0.1190	0.1156	0.0219
82	LS8-20180802-20180818	0.3505	0.3611	0.0201
83	LS8-20180802-20180818	0.3877	0.3877	0.0152
84	Sen2-20180304-20180314	0.3603	0.3370	-0.0893
85	Sen2-20180304-20180314	0.4548	0.3584	-0.0960
86	Sen2-20180304-20180314	0.5988	0.5988	-0.0765
87	Sen2-20180304-20180314	0.6035	0.5868	-0.0651
88	Sen2-20180304-20180314	1.1643	1.2044	0.0017
89	Sen2-20180304-20180314	0.7479	0.7479	-0.0534
90	Sen2-20180508-20180627	0.2157	0.2445	0.0497
91	Sen2-20180508-20180627	0.7432	0.7856	0.0237
92	Sen2-20180508-20180627	0.3419	0.3773	0.0461
93	Sen2-20180508-20180627	1.9676	1.9676	0.0835
94	Sen2-20180508-20180627	0.7659	0.8153	0.0572
95	Sen2-20180508-20180627	1.2740	1.3104	-0.0217
96	LS8-20180304-20180405	0.0881	0.0932	-0.0131

(continues on next page)

(continued from previous page)

```
97   LS8-20180304-20180405          0.0630          0.0735    -0.0100
98   LS8-20180802-20180818          0.0456          0.0471     0.0347
99   LS8-20180802-20180818          0.0271          0.0364     0.0332
100 Sen2-20180304-20180314          0.1413          0.1743    -0.0428
101 Sen2-20180304-20180314          0.1198          0.1597    -0.0420
102 Sen2-20180508-20180627          0.0647          0.0910     0.0376
103 Sen2-20180508-20180627          0.0806          0.1272     0.0379
104  LS8-20180304-20180405          0.1777          0.1568    -0.0093
105  LS8-20180304-20180405          0.1616          0.1665    -0.0044
106  LS8-20180802-20180818          0.2362          0.2227     0.0244
107  LS8-20180802-20180818          0.1576          0.1576     0.0299
108 Sen2-20180304-20180314          0.2655          0.2738    -0.0247
109 Sen2-20180304-20180314          0.2666          0.3182    -0.0253
110 Sen2-20180508-20180627          0.2856          0.3152     0.0674
111 Sen2-20180508-20180627          0.2631          0.3289     0.0625
112  LS8-20180304-20180405          0.1308          0.1439    -0.0142
113  LS8-20180304-20180405          0.2111          0.2256    -0.0253
114  LS8-20180304-20180405          0.1883          0.2054    -0.0143
115  LS8-20180802-20180818          0.0610          0.0650     0.0409
116  LS8-20180802-20180818          0.0814          0.0954     0.0630
117  LS8-20180802-20180818          0.0879          0.1055     0.0629
118 Sen2-20180304-20180314          0.2063          0.2188    -0.0578
119 Sen2-20180304-20180314          0.2915          0.2998    -0.0788
120 Sen2-20180304-20180314          0.2669          0.2905    -0.0781
121 Sen2-20180508-20180627          0.1077          0.1231     0.0425
122 Sen2-20180508-20180627          0.1830          0.2134     0.0570
123 Sen2-20180508-20180627          0.1622          0.2246     0.0586
124  LS8-20180304-20180405          0.6131          0.6131    -0.0198
125  LS8-20180304-20180405          0.4427          0.4579    -0.0153
126  LS8-20180304-20180405          0.6196          0.6364     0.0167
127  LS8-20180304-20180405          0.5880          0.6236     0.0178
128  LS8-20180304-20180405          0.3441          0.3678    -0.0119
129  LS8-20180304-20180405          0.2810          0.2997    -0.0094
130  LS8-20180304-20180405          0.2181          0.2393     0.0070
131  LS8-20180304-20180405          0.2132          0.2345     0.0071
132  LS8-20180304-20180405          0.1123          0.1189    -0.0099
133  LS8-20180304-20180405          0.0875          0.0987    -0.0141
134  LS8-20180304-20180405          0.0683          0.0854    -0.0122
135  LS8-20180304-20180405          0.0729          0.0885    -0.0130
136  LS8-20180802-20180818          0.2061          0.2121     0.0404
137  LS8-20180802-20180818          0.0945          0.1120     0.0551
138  LS8-20180802-20180818          0.0868          0.0987     0.0556
139  LS8-20180802-20180818          0.0833          0.0861     0.0558
140  LS8-20180802-20180818          0.0771          0.0820     0.0462
141  LS8-20180802-20180818          0.0588          0.0607     0.0567
142  LS8-20180802-20180818          0.0430          0.0459     0.0600
143  LS8-20180802-20180818          0.0480          0.0496     0.0602
144  LS8-20180802-20180818          0.0400          0.0387     0.0573
145  LS8-20180802-20180818          0.0422          0.0408     0.0600
146  LS8-20180802-20180818          0.0373          0.0373     0.0598
147  LS8-20180802-20180818          0.0397          0.0397     0.0573
148 Sen2-20180304-20180314          0.6263          0.6448    -0.0441
149 Sen2-20180304-20180314          0.4624          0.5057    -0.0481
150 Sen2-20180304-20180314          0.6021          0.6523    -0.0458
151 Sen2-20180304-20180314          0.5742          0.6280    -0.0446
152 Sen2-20180304-20180314          0.4216          0.4588    -0.0501
```

(continues on next page)

(continued from previous page)

```
153  Sen2-20180304-20180314            0.3489            0.3939    -0.0512
154  Sen2-20180304-20180314            0.3518            0.3958    -0.0515
155  Sen2-20180304-20180314            0.3446            0.3905    -0.0510
156  Sen2-20180304-20180314            0.1597            0.2150    -0.0441
157  Sen2-20180304-20180314            0.1139            0.1757    -0.0388
158  Sen2-20180304-20180314            0.1121            0.1536    -0.0417
159  Sen2-20180304-20180314            0.1046            0.1448    -0.0424
160  Sen2-20180508-20180627            0.4475            0.4908     0.0644
161  Sen2-20180508-20180627            0.3904            0.4294     0.0630
162  Sen2-20180508-20180627            0.4740            0.5147     0.0635
163  Sen2-20180508-20180627            0.5601            0.5940     0.0670
164  Sen2-20180508-20180627            0.2953            0.3600     0.0592
165  Sen2-20180508-20180627            0.2485            0.2999     0.0586
166  Sen2-20180508-20180627            0.2469            0.3065     0.0415
167  Sen2-20180508-20180627            0.2482            0.3218     0.0408
168  Sen2-20180508-20180627            0.1277            0.1629     0.0419
169  Sen2-20180508-20180627            0.0832            0.1165     0.0408
170  Sen2-20180508-20180627            0.0943            0.1292     0.0410
171  Sen2-20180508-20180627            0.0839            0.1193     0.0407

     SAV-peak-y  SAV-outlier-percent  LSR-uncertainty-nm  LSR-uncertainty-sh  \
0        0.2004               7.3336              0.0017              0.0014
1        0.1801               4.7821              0.0097              0.0091
2        0.1092              17.1979              0.0127              0.0127
3        0.2243              15.7562              0.0075              0.0075
4        0.1742               9.7302              0.0541              0.0541
5        0.1448              20.8806              0.0300              0.0285
6        0.2111               8.8609              0.0042              0.0037
7        0.1808               5.4252              0.0219              0.0213
8        0.1147              17.5695              0.0180              0.0180
9        0.1946               6.3275              0.0017              0.0014
10       0.1745               6.5314              0.0066              0.0064
11       0.1683              14.5105              0.0022              0.0017
12       0.2130              15.0450              0.0056              0.0058
13       0.1689              12.6928              0.0402              0.0402
14       0.2085              25.0045              0.0082              0.0079
15       0.2124               8.8189              0.0029              0.0025
16       0.1745               8.3283              0.0162              0.0162
17       0.1850              18.5936              0.0045              0.0040
18      -0.0200              14.1306              0.0083              0.0069
19       0.0038               9.2412              0.0312              0.0312
20      -0.0383              26.4707              0.0095              0.0079
21      -0.0218              11.7369              0.0034              0.0030
22       0.0127               9.6975              0.0114              0.0114
23      -0.0495              21.0126              0.0046              0.0040
24      -0.0102              11.5134              0.0022              0.0021
25       0.0107              10.6203              0.0068              0.0064
26      -0.0488              18.7794              0.0034              0.0031
27      -0.0179               9.3420              0.0034              0.0028
28       0.0058               5.6007              0.0186              0.0186
29      -0.0560              22.3524              0.0071              0.0061
30      -0.0245               7.0592              0.0018              0.0016
31       0.0096               2.4776              0.0069              0.0069
32      -0.0526              17.2777              0.0030              0.0030
33      -0.0235               6.5373              0.0012              0.0013
34       0.0074               2.4417              0.0040              0.0038
```

(continues on next page)

(continued from previous page)

35	-0.0656	15.1006	0.0021	0.0021
36	-0.0504	46.5259	0.0053	0.0056
37	-0.0201	58.3552	0.0208	0.0170
38	-0.0402	29.4852	0.0073	0.0071
39	-0.0398	53.8719	0.0223	0.0213
40	-0.0295	60.1249	0.0484	0.0484
41	-0.0260	39.7546	0.0181	0.0181
42	-0.0375	48.4499	0.0109	0.0109
43	-0.0323	56.7480	0.0257	0.0257
44	-0.0380	33.3910	0.0106	0.0106
45	-0.0512	39.9537	0.0057	0.0059
46	-0.0195	47.0995	0.0121	0.0117
47	-0.0294	23.8219	0.0045	0.0048
48	-0.0400	47.6595	0.0178	0.0174
49	-0.0210	49.9704	0.0404	0.0404
50	-0.0265	36.9220	0.0159	0.0134
51	-0.0496	42.2368	0.0091	0.0091
52	-0.0315	46.7270	0.0210	0.0210
53	-0.0254	28.7540	0.0082	0.0082
54	0.0216	13.2209	0.0361	0.0345
55	0.0126	5.8884	0.0853	0.0853
56	0.0333	23.2317	0.0255	0.0229
57	0.0259	7.7471	0.0120	0.0120
58	0.0085	5.0026	0.0300	0.0300
59	0.0341	15.1365	0.0106	0.0100
60	0.0336	4.7878	0.0063	0.0063
61	0.0118	4.9954	0.0180	0.0180
62	0.0422	12.9564	0.0059	0.0059
63	0.0274	15.5145	0.0214	0.0205
64	-0.0014	5.1392	0.0500	0.0500
65	0.0167	28.1699	0.0169	0.0161
66	0.0276	24.5823	0.0078	0.0075
67	0.0015	5.7859	0.0166	0.0172
68	0.0374	27.8417	0.0070	0.0066
69	0.0273	17.6893	0.0051	0.0051
70	0.0015	5.5337	0.0098	0.0098
71	0.0351	24.9979	0.0041	0.0039
72	-0.0135	13.4694	0.0026	0.0022
73	-0.0038	22.1503	0.0193	0.0165
74	-0.0314	11.8707	0.0020	0.0018
75	-0.0645	0.5065	0.0087	0.0076
76	-0.0657	15.8844	0.0037	0.0035
77	-0.0495	11.1949	0.0149	0.0140
78	0.0092	18.6395	0.0090	0.0090
79	-0.0044	24.0711	0.0250	0.0220
80	0.0116	10.4422	0.0102	0.0106
81	0.0047	11.3147	0.1170	0.1170
82	0.0356	1.7007	0.0204	0.0204
83	0.0248	3.8290	0.1075	0.1075
84	0.2314	7.5991	0.0062	0.0060
85	0.2173	20.5638	0.0255	0.0208
86	0.1313	5.7741	0.0053	0.0049
87	0.1456	10.5457	0.0366	0.0357
88	0.0643	6.4174	0.0089	0.0084
89	0.0742	8.5886	0.0191	0.0113
90	-0.0052	23.8444	0.0047	0.0050

(continues on next page)

(continued from previous page)

91	-0.0368	0.6403	0.0222	0.0158
92	-0.0009	30.6956	0.0059	0.0063
93	-0.0674	0.6589	0.0375	0.0306
94	0.0045	44.6672	0.0175	0.0169
95	-0.0471	2.2251	0.0655	0.0584
96	-0.0135	29.0491	0.0092	0.0073
97	-0.0184	23.9667	0.0052	0.0048
98	0.0083	27.7419	0.0139	0.0099
99	0.0111	24.7486	0.0082	0.0068
100	0.1522	27.9104	0.0152	0.0135
101	0.1443	20.9833	0.0107	0.0102
102	-0.0239	51.6452	0.0106	0.0078
103	-0.0260	36.6759	0.0066	0.0053
104	-0.0219	33.7973	0.0113	0.0092
105	-0.0411	27.9858	0.0068	0.0058
106	0.0415	43.3450	0.0242	0.0171
107	0.0330	34.0321	0.0161	0.0131
108	0.1599	44.8088	0.0272	0.0219
109	0.1422	32.5381	0.0169	0.0154
110	-0.0423	28.4293	0.0228	0.0168
111	-0.0366	28.1802	0.0113	0.0106
112	-0.0141	34.4991	0.0071	0.0058
113	-0.0219	43.7341	0.0098	0.0073
114	-0.0276	37.4733	0.0081	0.0060
115	0.0145	35.7138	0.0170	0.0115
116	0.0266	38.4609	0.0165	0.0122
117	0.0262	36.0039	0.0147	0.0105
118	0.1865	32.7865	0.0122	0.0104
119	0.2563	35.8267	0.0114	0.0102
120	0.2508	32.8217	0.0067	0.0057
121	-0.0252	51.9484	0.0135	0.0097
122	-0.0342	45.8089	0.0162	0.0127
123	-0.0396	45.4053	0.0164	0.0123
124	-0.0784	18.0561	0.0054	0.0050
125	-0.0446	10.8430	0.0032	0.0029
126	-0.0753	15.1762	0.0113	0.0113
127	-0.0764	10.2030	0.0041	0.0040
128	-0.0412	15.8391	0.0041	0.0037
129	-0.0387	14.1476	0.0026	0.0025
130	-0.0516	14.8987	0.0039	0.0037
131	-0.0515	13.0897	0.0028	0.0028
132	-0.0099	18.1982	0.0041	0.0035
133	-0.0028	20.0582	0.0028	0.0024
134	-0.0073	19.9290	0.0025	0.0021
135	-0.0111	18.4584	0.0019	0.0018
136	0.0283	9.1291	0.0290	0.0290
137	0.0271	28.4327	0.0071	0.0066
138	0.0257	30.1757	0.0388	0.0388
139	0.0003	30.0059	0.0176	0.0176
140	0.0174	33.5016	0.0221	0.0207
141	0.0019	33.7117	0.0059	0.0058
142	-0.0002	26.0707	0.0171	0.0171
143	-0.0007	26.0625	0.0093	0.0090
144	-0.0013	9.6297	0.0086	0.0079
145	-0.0014	9.4643	0.0045	0.0044
146	0.0012	2.6720	0.0076	0.0076

(continues on next page)

(continued from previous page)

```
147      0.0013             2.7077             0.0047             0.0045
148      0.1691            13.6812             0.0173             0.0169
149      0.1731            14.1731             0.0082             0.0077
150      0.1417            12.5667             0.0187             0.0182
151      0.1430            11.3357             0.0128             0.0124
152      0.1751            14.0344             0.0086             0.0074
153      0.1763            14.3255             0.0052             0.0046
154      0.1580            11.1769             0.0077             0.0075
155      0.1595            10.4699             0.0035             0.0031
156      0.1568            14.4379             0.0074             0.0060
157      0.1543            30.4767             0.0043             0.0035
158      0.1541            23.7677             0.0027             0.0029
159      0.1532            23.8466             0.0027             0.0023
160     -0.0231            50.2754             0.0254             0.0246
161     -0.0245            25.0381             0.0081             0.0076
162     -0.0240            49.9194             0.0227             0.0221
163     -0.0205            38.5338             0.0162             0.0157
164     -0.0158            37.8339             0.0162             0.0152
165     -0.0336            25.2227             0.0066             0.0062
166     -0.0165            34.1900             0.0136             0.0129
167     -0.0158            28.3005             0.0076             0.0076
168     -0.0081            21.8827             0.0070             0.0065
169     -0.0158            26.0745             0.0032             0.0033
170     -0.0035            22.3925             0.0044             0.0042
171     -0.0097            22.3095             0.0028             0.0029

     pt0_vxdiff  pt0_vxavgdiff  pt0_vydiff  pt0_vyavgdiff  pt1_vxdiff  \
0        0.0170        -0.0038      0.0959         0.0873  5.8948e-03
1       -0.2605        -0.2136      0.0059        -0.0079  1.7465e-02
2       -0.1812        -0.1725     -0.1182        -0.1165 -1.5325e-01
3       -0.0269        -0.0096      0.0044         0.0218  1.0557e-03
4       -0.2559        -0.2524     -0.0524        -0.0524  2.2109e-02
5       -0.1480        -0.1514     -0.1237        -0.1272 -1.5124e-01
6        0.0066         0.0101      0.0455         0.0525  3.3113e-03
7       -0.2584        -0.2479     -0.0466        -0.0570  1.9617e-02
8       -0.1371        -0.1336     -0.0921        -0.0990 -1.4035e-01
9        0.0156         0.0035      0.0399         0.0312  1.2364e-02
10      -0.2268        -0.2324     -0.0854        -0.0828 -3.5290e-03
11      -0.0069        -0.0069      0.0799         0.0869 -1.7947e-02
12       0.0039         0.0039      0.0369         0.0335  6.3904e-04
13      -0.2556        -0.2486     -0.0828        -0.0724 -8.8696e-03
14      -0.0097        -0.0097      0.0603         0.0603 -1.2984e-02
15       0.0057         0.0022      0.0464         0.0325  2.4364e-03
16      -0.2561        -0.2388     -0.0532        -0.0671  2.1836e-02
17      -0.0120        -0.0155      0.0854         0.0715 -1.5300e-02
18       0.0995         0.1126      0.1060         0.1222  3.4505e-02
19       0.2156         0.2237      0.2096         0.2080  3.3389e-02
20       0.0885         0.0901      0.1351         0.1335  2.3426e-02
21       0.1026         0.1124      0.1036         0.1142  2.2914e-02
22       0.2244         0.2301      0.2009         0.2025  3.4826e-02
23       0.0861         0.0943      0.1384         0.1449  2.1091e-02
24       0.1086         0.0806      0.1426         0.1214  1.7973e-02
25       0.2337        -0.5317      0.2245         0.9342  3.6784e-02
26       0.0895         0.0725      0.1638         0.1577  2.0850e-02
27       0.0689         0.0673      0.0553         0.0586  3.3213e-02
28       0.2429         0.2413      0.2031         0.2047 -2.7171e-02
```

(continues on next page)

(continued from previous page)

```
29       0.0600        0.0600      0.0938         0.0921  9.6336e-03
30       0.0699        0.0650      0.0563         0.0538  3.4176e-02
31       0.2432        0.2399      0.2002         0.1945 -2.6938e-02
32       0.0667        0.0659      0.0919         0.0911  1.6333e-02
33       0.0563        0.0453      0.0719         0.0597  3.5186e-02
34       0.2433        0.0769      0.2114         0.1048 -2.6781e-02
35       0.0583        0.0457      0.1143         0.1061  1.1610e-02
36      -1.1113       -0.8285      0.4461         0.1926  1.0767e+00
37       0.2557        0.1080     -0.1275        -0.3602         NaN
38      -0.8036       -0.5013     -0.7980        -0.7724  1.5345e+00
39      -1.2545       -0.9107      1.1561         0.6978 -8.5140e-02
40       0.2764        0.2743     -0.1255        -0.1234  1.1645e+00
41      -0.8436       -0.8429     -0.8692        -0.8490         NaN
42      -1.2566       -0.8934      1.1539         0.7324  1.4627e+00
43          NaN       -0.1075         NaN         0.2780  1.1555e+00
44      -0.8442       -0.8428     -0.8464        -0.8054         NaN
45      -0.9384       -0.4954      0.8463         0.4423  1.1012e+00
46          NaN       -1.6615         NaN         0.0849 -9.6097e-01
47      -0.0335       -0.2307      0.0086        -0.1990  6.2953e-01
48      -1.2793       -1.2765      1.1688         1.1757  1.4776e+00
49          NaN           NaN         NaN            NaN -1.0600e+00
50      -0.0492       -0.0464      0.0113         0.0134  6.3886e-01
51      -1.2813       -1.2820      1.1668         1.1605 -8.0720e-02
52          NaN        0.0715         NaN        -0.0367 -1.0561e+00
53      -0.0501       -0.1938      0.0105        -0.1444  6.3175e-01
54      -0.1641       -0.1706     -0.2251        -0.2348  1.1510e-01
55      -0.4117       -0.1956     -0.3527        -0.3397  1.3946e-02
56      -0.1412       -0.1673     -0.2014        -0.2307  6.8762e+00
57      -0.1611       -0.1464     -0.2351        -0.2318  1.1809e-01
58      -0.4131       -0.2684     -0.3365        -0.4114         NaN
59      -0.1395       -0.1362     -0.2042        -0.2124  3.5088e+00
60      -0.1514       -0.1425     -0.2374        -0.2333  1.2041e-01
61      -0.4283       -0.3803     -0.3289        -0.4477         NaN
62      -0.1540       -0.1670     -0.2303        -0.2442  1.3251e-01
63      -0.1766       -0.1766     -0.2196        -0.2196  4.4007e-02
64       0.4125        0.4028     -0.3376        -0.3279 -4.0729e-02
65      -0.2298       -0.2200     -0.2567        -0.2665 -2.7289e-01
66      -0.1724       -0.1724     -0.2062        -0.2127  6.2874e-02
67       0.4103        0.0571     -0.3399        -0.3268 -4.2939e-02
68      -0.2078       -0.2208     -0.2683        -0.2715 -2.5091e-01
69      -0.1677       -0.1701     -0.2157        -0.2230  6.0209e-02
70       0.2076        0.0018     -0.3328        -0.3287 -1.1225e-02
71      -0.2069       -0.2159     -0.2747        -0.2796 -2.6464e-01
72       0.0918        0.0309      0.3047         0.1419 -1.2166e-01
73       0.1660        0.0859      0.2165         0.1544 -6.6577e-02
74       0.1674        0.0272      0.2026         0.0382 -4.3214e-02
75      -0.2468       -0.0987     -0.2989        -0.1285 -2.3460e-01
76      -0.2725       -0.1006     -0.2716        -0.0403 -1.5342e-01
77       0.0349       -0.1355      0.2866        -0.0584 -2.9092e-01
78      -0.1380       -0.1892     -0.3882        -0.3287 -1.2620e-01
79      -0.1696       -0.2158     -0.4251        -0.3509 -2.1908e-01
80      -0.3353       -0.3220     -0.4415        -0.4148 -2.8399e-01
81      -0.3537       -0.3293     -0.3627        -0.3917 -3.8832e-01
82      -0.3516       -0.3590     -0.4839        -0.4641 -3.3464e-01
83      -0.4155       -0.3519     -0.3605        -0.4168 -4.7093e-01
84      -0.1144       -0.1574     -0.1225        -0.1420  1.0184e-01
```

(continues on next page)

(continued from previous page)

```
85      -0.1721        -0.1458      0.0778        -0.1285  4.6447e-01
86      -0.1778        -0.1642     -0.1310        -0.0980 -1.6083e-02
87      -0.1288        -0.2019      0.1627        -0.1329 -7.7555e-02
88      -0.4083        -0.3880     -0.0444        -0.0424 -1.4841e-01
89      -0.3233        -0.3568     -0.0922        -0.1053 -1.6358e-01
90      -0.0224        -0.0578     -0.0462        -0.0301  5.2352e-01
91      -0.0226        -0.0199     -0.0812        -0.0309  3.6209e-01
92      -0.0585        -0.0558     -0.0944        -0.0324  5.5653e-01
93      -0.2497        -2.5122      2.5069        11.7078  7.3513e-01
94      -0.1374        -0.2582     -0.0226        -0.1105  1.5611e-01
95      -0.1444        -0.1810     -0.2353        -0.3435 -1.0798e-01
96       0.0376         0.0292      0.0059        -0.0050 -1.1141e-05
97       0.0130         0.0103     -0.0112        -0.0147  1.2461e-01
98          NaN         1.0984         NaN         1.5620  3.8821e-01
99      -0.0414        -0.0466     -0.0172        -0.0241  7.2844e-01
100     -0.0340        -0.0391      0.2267         0.2197 -5.0500e-02
101     -0.0038        -0.0045      0.2534         0.2500 -7.5227e-02
102     -0.0064        -0.0052      0.1280         0.1248         NaN
103     -0.0365        -0.0376      0.0323         0.0272  5.9721e-01
104     -0.0035        -0.0217     -0.0207        -0.0478 -1.3471e-02
105     -0.0017        -0.0051     -0.0071        -0.0116 -1.8728e-02
106     -0.4442        -0.4391     -0.6028        -0.5953         NaN
107      0.2356         0.2288      0.2431         0.2383         NaN
108     -0.0118        -0.0113      0.2415         0.2508  3.5119e-03
109      0.0082        -0.0001      0.2345         0.2206 -1.1961e-01
110         NaN            NaN         NaN            NaN         NaN
111         NaN            NaN         NaN            NaN         NaN
112     -0.3900        -0.3952     -0.4651        -0.4715  4.5278e-02
113     -1.9091        -1.4803      1.5938         1.0851  1.2128e-01
114      0.0722         0.0692      0.1259         0.1229  7.8072e-02
115     -0.8324        -0.8644     -0.8831        -0.9199 -2.4855e-02
116     -0.1921        -0.1915     -0.1445        -0.1480 -6.9549e-01
117     -0.1456        -0.1423     -0.1177        -0.1179 -1.9687e+00
118      0.2553         0.2545      0.5209         0.5241 -5.1806e-02
119     -0.7713        -0.9439     -0.6777        -0.7220  3.7812e-02
120      0.0094         0.0100      0.0530         0.0532  3.8034e-02
121     -0.0675        -0.0660      0.1195         0.1177         NaN
122     -0.1060        -0.1021      0.0565         0.0608         NaN
123     -0.0788        -0.0756      0.0717         0.0725         NaN
124     -0.7211        -0.5291     -1.1742        -0.8682  4.8785e-02
125      0.1539         0.1070      0.1988         0.1842  4.4881e-02
126      0.0965         0.0998      0.1415         0.1578  1.6853e-02
127      0.1252         0.0893      0.1701         0.1473  1.6180e-02
128     -0.7287        -0.5367     -1.2044        -0.8984  4.1190e-02
129      0.1463         0.0994      0.1616         0.1470  3.7293e-02
130      0.0864         0.0864      0.0898         0.1061  6.6877e-03
131      0.1144         0.0785      0.1109         0.0914  5.3740e-03
132      0.1195         0.1097      0.1673         0.1542  3.9824e-02
133      0.1246         0.1051      0.1395         0.1492  4.4941e-02
134      0.0925         0.0860      0.1423         0.1423 -1.6439e-02
135      0.0935         0.0838      0.1413         0.1380  1.3838e-02
136     -0.2794        -0.4292     -0.3851        -0.8213  3.2201e-01
137     -0.2820         0.0110     -0.3876        -0.0816  3.1943e-01
138     -0.2782        -0.2782     -0.3523        -0.3523 -2.6268e-01
139     -0.2789        -0.2984     -0.3516        -0.3516 -2.6340e-01
140     -0.2731         0.0459     -0.3787        -0.1509  3.5379e-02
```

(continues on next page)

(continued from previous page)

```
141     -0.2153        -0.2446     -0.3210        -0.2917  3.4546e-02
142     -0.2898        -0.2898     -0.2817        -0.2817 -1.5708e-01
143     -0.2896        -0.2831     -0.2822        -0.2822 -1.5695e-01
144     -0.2342        -0.2472     -0.2815        -0.2750  7.4255e-02
145     -0.2343        -0.2409     -0.2201        -0.2332  7.4132e-02
146     -0.2341        -0.2341     -0.2229        -0.2294 -4.2797e-02
147     -0.2326        -0.2326     -0.2202        -0.2267 -4.1288e-02
148     -0.0809        -0.0948      0.0371         0.0024  9.5480e-03
149     -0.0809        -0.1087      0.0371        -0.0115  9.5767e-03
150     -0.1447        -0.1031     -0.0267         0.0219  8.2562e-03
151     -0.1467        -0.1120     -0.0287         0.0200  6.2942e-03
152     -0.1406        -0.1059     -0.0226        -0.0087  1.2348e-02
153     -0.1390        -0.1112     -0.0455        -0.0386  1.4004e-02
154     -0.1385        -0.0968     -0.0594        -0.0108  1.4521e-02
155     -0.1375        -0.1027     -0.0819        -0.0333  1.5527e-02
156     -0.1430        -0.1430     -0.0249        -0.0319  1.0016e-02
157     -0.1521        -0.1521     -0.0442        -0.0303  9.2328e-04
158     -0.1309        -0.1309     -0.0045         0.0094  2.2091e-02
159     -0.1305        -0.1305     -0.0048         0.0090  2.2462e-02
160     -0.0628        -0.0545      0.0473         0.0611         NaN
161     -0.0486        -0.0455      0.0330         0.0455 -3.0605e+00
162     -0.0502        -0.0557      0.0479         0.0423  6.6289e-01
163     -0.0356        -0.0495      0.0325         0.0214  6.7746e-01
164     -0.0589        -0.0451      0.0184         0.0364         NaN
165     -0.0304        -0.0249      0.0176         0.0232         NaN
166     -0.0442        -0.0497      0.0417         0.0348  6.4391e-01
167     -0.0314        -0.0453      0.0299         0.0244  6.4415e-01
168     -0.0420        -0.0309     -0.0486        -0.0402  5.2104e-01
169     -0.0286        -0.0230     -0.0435        -0.0310  5.4700e-01
170     -0.0535        -0.0521     -0.0371        -0.0343  5.3459e-01
171     -0.0407        -0.0434     -0.0499        -0.0499  5.4740e-01

     pt1_vxavgdiff  pt1_vydiff  pt1_vyavgdiff  pt2_vxdiff  pt2_vxavgdiff  \
0           0.0137      0.0822         0.0822      0.5122         0.5469
1           0.0183      0.0625         0.0660         NaN         1.3050
2          -0.1550      0.1493         0.1423      3.6578         3.5466
3           0.0011      0.0532         0.0532      0.5699         0.5490
4           0.0186      0.0589         0.0589      5.6534         0.7315
5          -0.1512      0.1126         0.1126      2.6676         2.5565
6           0.0068      0.0630         0.0630      0.5409         0.5305
7           0.0231      0.0334         0.0508      0.8072         1.9072
8          -0.1404      0.1442         0.1442      2.6785         2.7896
9           0.0236      0.0809         0.0809      0.4796         0.3893
10         -0.0061      0.0650         0.0650      7.6825         2.6356
11         -0.0179      0.1209         0.1218      0.4884        -0.0047
12          0.0006      0.0545         0.0545      0.4757         0.4861
13         -0.0054      0.0597         0.0597      8.8099         8.2440
14         -0.0130      0.0778         0.0778     -0.4129        -0.6560
15          0.0024      0.0639         0.0639      0.4775         0.4636
16          0.0079      0.0581         0.0581      7.7157         4.5143
17         -0.0153      0.1029         0.1029     -0.5715        -0.0263
18          0.0264      0.0499         0.0450         NaN            NaN
19          0.0383      0.0364         0.0397         NaN            NaN
20          0.0234      0.0791         0.0791         NaN            NaN
21          0.0229      0.0476         0.0427         NaN            NaN
22          0.0446      0.0496         0.0504         NaN            NaN
```

(continues on next page)

(continued from previous page)

```
23          0.0170      0.0897        0.0872        NaN           NaN
24          0.0233      0.0426        0.0422        NaN           NaN
25          0.0234      0.0513        0.0729        NaN           NaN
26          0.0107      0.0894        0.0903        NaN           NaN
27          0.0283      0.0432        0.0481        NaN           NaN
28         -0.0158      0.0738        0.0738        NaN           NaN
29          0.0113      0.0963        0.0963        NaN           NaN
30          0.0285      0.0442        0.0491        NaN           NaN
31         -0.0147      0.0710        0.0710        NaN           NaN
32          0.0131      0.0944        0.0944        NaN           NaN
33          0.0307      0.0489        0.0554        NaN           NaN
34         -0.0072      0.0711        0.0715        NaN           NaN
35          0.0084      0.1058        0.1058        NaN           NaN
36          0.4513      0.1845        0.8707        NaN           NaN
37         -0.3856         NaN       -0.1754        NaN           NaN
38          1.0954      0.6092        0.0197        NaN           NaN
39          0.4267      1.4274        0.8989        NaN           NaN
40          0.1652      0.8332       -0.4126        NaN           NaN
41          0.0194         NaN       -0.9292        NaN           NaN
42          0.9412     -0.1935        0.3440        NaN           NaN
43          0.1937      0.8368       -0.7000        NaN           NaN
44          1.5189         NaN        0.2358        NaN           NaN
45          0.0946      0.2300        1.2498        NaN           NaN
46         -0.6324     -1.5453       -0.7128        NaN           NaN
47          0.6248     -0.2014       -0.1867        NaN           NaN
48          0.7838     -0.1662        0.5366        NaN           NaN
49         -1.0079     -1.5912       -1.4704        NaN           NaN
50          0.6354     -0.2174       -0.2132        NaN           NaN
51          0.0915      1.4255        1.2436        NaN           NaN
52         -0.7971     -1.1686       -1.1901        NaN           NaN
53          0.6283     -0.2182       -0.2078        NaN           NaN
54          0.1379      0.1842        0.1581        NaN           NaN
55          0.0066      0.1151        0.1225        NaN           NaN
56          6.2903     -4.2746       -3.8709        NaN           NaN
57          0.1360      0.1156        0.1009        NaN           NaN
58          0.2567         NaN        0.0043        NaN           NaN
59          2.0228     -2.0802       -1.1036        NaN           NaN
60          0.0854      0.0913        0.0978        NaN           NaN
61          1.3890         NaN        0.8934        NaN           NaN
62         -0.4941      0.1496        0.5012        NaN           NaN
63          0.0440      0.0139        0.0139        NaN           NaN
64         -0.0505      0.1302        0.1302        NaN           NaN
65         -0.2534      0.1818        0.1590        NaN           NaN
66          0.0547      0.0126        0.0159        NaN           NaN
67         -0.0446      0.1280        0.1198        NaN           NaN
68         -0.2639      0.1410        0.1426        NaN           NaN
69          0.0488      0.0178        0.0064        NaN           NaN
70         -0.0316      0.0984        0.1066        NaN           NaN
71         -0.2500      0.1565        0.1256        NaN           NaN
72         -0.1119      0.0599        0.0384        NaN           NaN
73         -0.1200      0.1523        0.0776        NaN           NaN
74         -0.0999      0.0553        0.0746        NaN           NaN
75         -0.2131      0.1253        0.1493        NaN           NaN
76         -0.1945      0.1285        0.1601        NaN           NaN
77         -0.2554      0.1714        0.2011        NaN           NaN
78         -0.1929      0.3718        0.2917        NaN           NaN
```

(continues on next page)

(continued from previous page)

79	-0.2057	0.6850	0.5102	NaN	NaN
80	-0.3060	0.2271	0.2669	NaN	NaN
81	-0.3348	0.3035	0.3663	NaN	NaN
82	-0.3467	0.3072	0.3029	NaN	NaN
83	-0.5124	0.2826	0.2748	NaN	NaN
84	0.0843	0.0144	0.0403	0.2453	0.3415
85	0.2472	-0.3458	-0.0193	1.1736	0.1341
86	0.0008	0.1029	0.1056	0.2011	0.3290
87	-0.0554	0.1599	0.1115	0.3455	0.4606
88	-0.1253	0.1741	0.1958	-0.6099	-0.3913
89	-0.1878	0.1794	0.1307	-0.0596	-0.0979
90	0.5786	-0.2116	-0.2854	NaN	NaN
91	0.4942	-0.2839	-0.2111	NaN	NaN
92	0.3069	-0.1990	0.0073	NaN	NaN
93	0.3962	0.2569	-0.0211	NaN	NaN
94	0.4809	-2.8395	0.1581	NaN	NaN
95	-0.0781	-0.2016	-0.2435	NaN	NaN
96	0.0009	0.0423	0.0455	NaN	NaN
97	0.1139	0.0456	0.0491	NaN	NaN
98	0.7616	0.1324	-0.1047	NaN	NaN
99	0.7571	-0.2751	-0.2879	NaN	NaN
100	-0.0492	0.0591	0.0578	0.4655	0.7581
101	-0.0744	0.0746	0.0744	0.6170	0.6189
102	NaN	NaN	NaN	NaN	NaN
103	0.5992	-0.2309	-0.2334	NaN	NaN
104	-0.0139	0.0381	0.0417	NaN	NaN
105	-0.0176	0.0838	0.0843	NaN	NaN
106	NaN	NaN	NaN	NaN	NaN
107	-0.2646	NaN	-0.2389	NaN	NaN
108	-0.0064	0.0434	0.0462	NaN	-1.2071
109	-0.1181	0.1117	0.1112	0.7778	0.5555
110	NaN	NaN	NaN	NaN	NaN
111	NaN	NaN	NaN	NaN	NaN
112	0.0447	0.0624	0.0618	NaN	NaN
113	0.1165	0.0124	0.0135	NaN	NaN
114	0.0804	0.0510	0.0476	NaN	NaN
115	-0.4097	0.3606	0.5915	NaN	NaN
116	-0.7163	0.9321	0.6109	NaN	NaN
117	-1.9047	0.8808	0.8376	NaN	NaN
118	-0.0549	0.0448	0.0465	0.4400	0.4989
119	0.0370	-0.0223	-0.0214	0.0355	0.7619
120	0.0371	-0.0088	-0.0078	NaN	NaN
121	NaN	NaN	NaN	NaN	NaN
122	NaN	NaN	NaN	NaN	NaN
123	NaN	NaN	NaN	NaN	NaN
124	0.0293	0.0881	0.0881	NaN	NaN
125	0.0221	0.0842	0.1005	NaN	NaN
126	0.0169	0.1440	0.1343	NaN	NaN
127	0.0162	0.1434	0.1434	NaN	NaN
128	0.0282	0.0579	0.0579	NaN	NaN
129	0.0210	0.0470	0.0633	NaN	NaN
130	0.0067	0.0924	0.0891	NaN	NaN
131	0.0054	0.0841	0.0841	NaN	NaN
132	0.0170	0.0233	0.0331	NaN	NaN
133	0.0319	0.0248	0.0541	NaN	NaN
134	-0.0067	0.0569	0.0569	NaN	NaN

(continues on next page)

(continued from previous page)

```
135        0.0041      0.0559        0.0559         NaN           NaN
136        0.2309      0.1414        0.2000         NaN           NaN
137        0.0667      0.1388        0.2950         NaN           NaN
138       -0.2822      0.1741        0.1643         NaN           NaN
139       -0.2569      0.1748        0.1260         NaN           NaN
140        0.0354      0.2063        0.1770         NaN           NaN
141       -0.0424      0.2055        0.1908         NaN           NaN
142       -0.1180      0.1275        0.1080         NaN           NaN
143       -0.1114      0.1271        0.1075         NaN           NaN
144        0.0352      0.1277        0.1538         NaN           NaN
145       -0.0105      0.1305        0.1500         NaN           NaN
146       -0.0298      0.1278        0.1148         NaN           NaN
147       -0.0218      0.1304        0.1109         NaN           NaN
148        0.0165      0.0859        0.0859         NaN       -0.4060
149        0.0096      0.0859        0.0859         NaN        0.1800
150        0.0083      0.1471        0.1471     -1.2043        0.0423
151        0.0063      0.1451        0.1451      0.2314        0.2210
152        0.0054      0.0887        0.0887      0.1124        0.2235
153        0.0140      0.0657        0.0657      0.1141        0.3016
154        0.0145      0.1144        0.1144      0.3021        0.3021
155        0.0155      0.0918        0.0918      0.3031        0.3170
156        0.0447      0.0863        0.0863      0.1726        0.4921
157        0.0634      0.0671        0.0671      0.1635        0.4864
158        0.0429      0.1068        0.1068      0.6847        0.7055
159        0.0225      0.1064        0.1064      0.6851        0.6712
160       -3.0623         NaN       -2.9252         NaN           NaN
161       -3.0605     -2.9270       -2.9270         NaN           NaN
162       -0.1566     -0.1871       -0.7968         NaN           NaN
163        0.6768     -0.2025       -0.1914         NaN           NaN
164        1.3291         NaN        0.6834         NaN           NaN
165        0.7410         NaN       -0.1257         NaN           NaN
166        0.6439     -0.2058       -0.2058         NaN           NaN
167        0.6469     -0.2051       -0.1981         NaN           NaN
168        0.5502     -0.2210       -0.2363         NaN           NaN
169        0.5567     -0.2285       -0.2424         NaN           NaN
170        0.5401     -0.2596       -0.2665         NaN           NaN
171        0.5363     -0.2599       -0.2557         NaN           NaN

     pt2_vydiff  pt2_vyavgdiff  Invalid-pixel-percent
0        0.8054         0.7776                 7.8054
1           NaN         3.1347                 5.0161
2       -3.2525        -3.5824                12.3292
3        0.7139         0.6757                 5.1724
4       -5.4367        -5.4211                 3.1805
5       -4.2893        -4.4004                 8.6957
6        0.7237         0.6925                 6.2864
7        5.5691         3.5566                 4.0659
8       -4.2576        -4.1465                10.0686
9        0.6790         0.5992                 6.1368
10       5.6631         3.3428                 2.9373
11       0.3753         0.4065                10.7179
12       0.6214         0.6318                 4.0798
13       0.5641         2.4634                 1.8283
14       0.1760        -0.2615                 7.8952
15       0.6621         0.6447                 4.8960
16       5.6250         2.2812                 2.2924
```

(continues on next page)

(continued from previous page)

```
17      -0.3614         0.3434                 8.9567
18          NaN            NaN                 7.9968
19          NaN            NaN                 5.3606
20          NaN            NaN                16.2653
21          NaN            NaN                10.0900
22          NaN            NaN                 6.9775
23          NaN            NaN                19.4000
24          NaN            NaN                11.7371
25          NaN            NaN                 8.0455
26          NaN            NaN                21.9897
27          NaN            NaN                 2.7461
28          NaN            NaN                 1.4010
29          NaN            NaN                11.9498
30          NaN            NaN                 3.3578
31          NaN            NaN                 1.7428
32          NaN            NaN                13.4295
33          NaN            NaN                 3.8834
34          NaN            NaN                 2.0343
35          NaN            NaN                14.9028
36          NaN            NaN                43.5604
37          NaN            NaN                28.6727
38          NaN            NaN                41.3422
39          NaN            NaN                31.4317
40          NaN            NaN                18.7101
41          NaN            NaN                30.5405
42          NaN            NaN                36.8260
43          NaN            NaN                23.6210
44          NaN            NaN                35.0862
45          NaN            NaN                42.6848
46          NaN            NaN                23.7481
47          NaN            NaN                38.6010
48          NaN            NaN                32.4413
49          NaN            NaN                15.4606
50          NaN            NaN                29.5631
51          NaN            NaN                36.8436
52          NaN            NaN                19.2125
53          NaN            NaN                33.0943
54          NaN            NaN                 6.4834
55          NaN            NaN                 6.2516
56          NaN            NaN                14.5197
57          NaN            NaN                 8.1546
58          NaN            NaN                 8.0007
59          NaN            NaN                16.5318
60          NaN            NaN                 9.3372
61          NaN            NaN                 9.0537
62          NaN            NaN                18.2080
63          NaN            NaN                 4.8015
64          NaN            NaN                 3.7828
65          NaN            NaN                12.4996
66          NaN            NaN                 5.6545
67          NaN            NaN                 4.6980
68          NaN            NaN                13.5827
69          NaN            NaN                 6.3116
70          NaN            NaN                 5.2927
71          NaN            NaN                14.6222
72          NaN            NaN                 0.0000
```

(continues on next page)

(continued from previous page)

```
73           NaN           NaN                0.0000
74           NaN           NaN                0.0000
75           NaN           NaN                0.0000
76           NaN           NaN                0.0000
77           NaN           NaN                0.0000
78           NaN           NaN                0.0000
79           NaN           NaN                0.0000
80           NaN           NaN                0.0000
81           NaN           NaN                0.0002
82           NaN           NaN                0.0000
83           NaN           NaN                0.0000
84        0.2964        0.6304                0.0000
85       -0.0538       -0.3457                0.0000
86        0.5162        0.6503                0.0000
87        0.7332        0.8116                0.0000
88        0.0093        0.0062                0.0000
89        0.3611       -0.2225                0.0001
90           NaN           NaN                0.0000
91           NaN           NaN                0.0000
92           NaN           NaN                0.0000
93           NaN           NaN                0.0002
94           NaN           NaN                0.0000
95           NaN           NaN                0.0000
96           NaN           NaN               13.6845
97           NaN           NaN               10.1788
98           NaN           NaN               14.1996
99           NaN           NaN               12.8649
100       0.7203        0.8243               12.3226
101       0.7597        0.7563                8.9881
102          NaN           NaN               35.3302
103          NaN           NaN               16.5616
104          NaN           NaN               40.3571
105          NaN           NaN               26.3141
106          NaN           NaN               38.5181
107          NaN           NaN               29.0936
108       0.5887        0.5887               29.1637
109       0.6129        0.6129               18.4418
110          NaN           NaN               85.1256
111          NaN           NaN               67.8554
112          NaN           NaN               16.7852
113          NaN           NaN               13.7038
114          NaN           NaN               17.1474
115          NaN           NaN               16.4240
116          NaN           NaN               16.4751
117          NaN           NaN               17.4662
118       0.7080        0.7377               13.2756
119       0.4521        0.9156               10.2991
120          NaN           NaN               13.5545
121          NaN           NaN               43.7494
122          NaN           NaN               35.6635
123          NaN           NaN               37.4170
124          NaN           NaN               10.4465
125          NaN           NaN               21.2842
126          NaN           NaN                2.7206
127          NaN           NaN                7.5009
128          NaN           NaN                7.5823
```

(continues on next page)

(continued from previous page)

```
129         NaN            NaN                13.5491
130         NaN            NaN                 1.5149
131         NaN            NaN                 3.3336
132         NaN            NaN                 5.8797
133         NaN            NaN                 8.3062
134         NaN            NaN                 0.5931
135         NaN            NaN                 1.0652
136         NaN            NaN                11.9963
137         NaN            NaN                19.6043
138         NaN            NaN                 4.6293
139         NaN            NaN                10.6445
140         NaN            NaN                 7.3143
141         NaN            NaN                11.9107
142         NaN            NaN                 2.4309
143         NaN            NaN                 5.3829
144         NaN            NaN                 7.5362
145         NaN            NaN                 9.3943
146         NaN            NaN                 2.8453
147         NaN            NaN                 4.4327
148      0.2153         0.1389                 6.2205
149      0.2153         0.5140                13.1742
150     -0.4110         0.2418                 1.9720
151      0.5871         0.4690                 5.5130
152      0.5306         0.6070                 4.8452
153      0.5077         0.6188                 9.9877
154      0.6188         0.6188                 1.6604
155      0.5963         0.6033                 3.8936
156      0.5283         0.8096                 3.9320
157      0.5091         0.7417                 5.6876
158      0.8613         0.8891                 0.6131
159      0.8609         0.8540                 1.0475
160         NaN            NaN                22.0370
161         NaN            NaN                45.0053
162         NaN            NaN                 7.1902
163         NaN            NaN                19.7589
164         NaN            NaN                16.3373
165         NaN            NaN                29.8371
166         NaN            NaN                 4.3482
167         NaN            NaN                 9.3208
168         NaN            NaN                16.3125
169         NaN            NaN                21.4989
170         NaN            NaN                 3.0184
171         NaN            NaN                 4.9545
```

5.1 Abbreviations in Table S2

- LS8: Landsat 8
- Sen2: Sentinel-2
- SAV-uncertainty-x: correct-match uncertainty of the static terrain velocity (Metric 1; x component, m/day)
- SAV-uncertainty-y: correct-match uncertainty of the static terrain velocity (Metric 1; y component, m/day)
- SAV-peak-x: correct-match peak location (x component, m/day)

- SAV-peak-y: correct-match peak location (y component, m/day)
- SAV-outlier-percent: amount of incorrect matches (in percent of all the static terrain pixels)
- LSR-uncertainty-nm: variability of longitudinal normal strain rate (1/day)
- LSR-uncertainty-sh: variability of longitudinal shear strain rate (Metric 2; 1/day)
- pt0: the easternmost GNSS station
- pt1: the GNSS station in the middle
- pt2: the westernmost GNSS station
- vxdiff: difference between the GNSS measurement and the value sampled (nearest neighbor) from the velocity map (x component, m/day)
- vydiff: difference between the GNSS measurement and the value sampled (nearest neighbor) from the velocity map (x component, m/day)
- vxavgdiff: difference between the GNSS measurement and the value sampled (averaged from the nearest 3x3 array) from the velocity map (x component, m/day)
- vyavgdiff: difference between the GNSS measurement and the value sampled (averaged from the nearest 3x3 array) from the velocity map (x component, m/day)
- Invalid-pixel-percent: amount of invalid pixels (in percent of all pixels in the velocity map)

CHAPTER

SIX

TABLE S3: LIST OF ALL 35 ITS_LIVE MAPS

The machine-readable CSV file is available at `notebooks/manifest_ITSLIVE.csv`.

```
                     Label Start date  End date Duration (days)
0    LS8-20180304-20180405   20180304  20180405              32
1    LS8-20180405-20180421   20180405  20180421              16
2    LS8-20180421-20180523   20180421  20180523              32
3    LS8-20180523-20180608   20180523  20180608              16
4    LS8-20180412-20180428   20180412  20180428              16
5    LS8-20180428-20180802   20180428  20180802              96
6    LS8-20180802-20180818   20180802  20180818              16
7    LS8-20180818-20180903   20180818  20180903              16
8    LS8-20180903-20181005   20180903  20181005              32
9   Sen2-20180306-20180316   20180306  20180316              10
10  Sen2-20180316-20180515   20180316  20180515              60
11  Sen2-20180329-20180508   20180329  20180508              40
12  Sen2-20180508-20180518   20180508  20180518              10
13  Sen2-20180508-20180627   20180508  20180627              50
14  Sen2-20180515-20180619   20180515  20180619              35
15  Sen2-20180518-20180523   20180518  20180523               5
16  Sen2-20180627-20180722   20180627  20180722              25
17  Sen2-20180704-20180724   20180704  20180724              20
18  Sen2-20180724-20180729   20180724  20180729               5
19  Sen2-20180727-20180801   20180727  20180801               5
20  Sen2-20180304-20180314   20180304  20180314              10
21  Sen2-20180314-20180329   20180314  20180329              15
22  Sen2-20180523-20180612   20180523  20180612              20
23  Sen2-20180612-20180627   20180612  20180627              15
24  Sen2-20180619-20180704   20180619  20180704              15
25  Sen2-20180722-20180727   20180722  20180727               5
26  Sen2-20180729-20180818   20180729  20180818              20
27  Sen2-20180801-20180811   20180801  20180811              10
28  Sen2-20180811-20180831   20180811  20180831              20
29  Sen2-20180818-20180917   20180818  20180917              30
30  Sen2-20180831-20180910   20180831  20180910              10
31  Sen2-20180910-20180920   20180910  20180920              10
32  Sen2-20180917-20181002   20180917  20181002              15
33  Sen2-20180920-20180930   20180920  20180930              10
34  Sen2-20180930-20181005   20180930  20181005               5
```

6.1 Abbreviations in Table S3

- LS8: Landsat 8
- Sen2: Sentinel-2

6.2 Feature-tracking parameters of the ITS_LIVE velocity maps (same for all)

- Template size: varying (240-480 m)
- Pixel spacing: 120 m
- Prefilter: 5x5 Wallis operator
- Subpixel sampling: 16-node oversampling
- Processing software: autoRIFT

CHAPTER

SEVEN

TABLE S4: METRICS FOR THE ITS_LIVE VELOCITY MAPS

The machine-readable CSV file is available at `notebooks/results_ITSLIVE.csv`.

```
                     Label  Assigned-x-error  Assigned-y-error  \
0    LS8-20180304-20180405              47.7              32.6
1    LS8-20180405-20180421             119.6              71.7
2    LS8-20180421-20180523              92.5              48.2
3    LS8-20180523-20180608              75.5              64.0
4    LS8-20180412-20180428             132.2              68.8
5    LS8-20180428-20180802              32.0              18.5
6    LS8-20180802-20180818              48.3              44.9
7    LS8-20180818-20180903              49.9              47.6
8    LS8-20180903-20181005              38.7              30.8
9   Sen2-20180306-20180316             102.3              62.4
10  Sen2-20180316-20180515              43.0              18.5
11  Sen2-20180329-20180508              52.1              25.2
12  Sen2-20180508-20180518             175.7             106.0
13  Sen2-20180508-20180627              37.1              21.5
14  Sen2-20180515-20180619              45.3              30.4
15  Sen2-20180518-20180523             133.3             110.6
16  Sen2-20180627-20180722              39.0              29.1
17  Sen2-20180704-20180724              36.7              28.2
18  Sen2-20180724-20180729              70.3              66.5
19  Sen2-20180727-20180801              67.1              60.1
20  Sen2-20180304-20180314             121.4              73.1
21  Sen2-20180314-20180329              72.3              45.5
22  Sen2-20180523-20180612              50.4              39.7
23  Sen2-20180612-20180627              80.5              61.2
24  Sen2-20180619-20180704              61.6              52.5
25  Sen2-20180722-20180727              76.0              71.5
26  Sen2-20180729-20180818              24.8              20.0
27  Sen2-20180801-20180811              39.1              33.8
28  Sen2-20180811-20180831              33.4              29.6
29  Sen2-20180818-20180917              32.9              22.8
30  Sen2-20180831-20180910              65.7              52.0
31  Sen2-20180910-20180920              73.9              71.4
32  Sen2-20180917-20181002              54.4              45.0
33  Sen2-20180920-20180930              78.2              63.7
34  Sen2-20180930-20181005             103.5              93.5

    SAV-uncertainty-x  SAV-uncertainty-y  SAV-peak-x  SAV-peak-y  \
0             51.1488            44.5490    -10.7502     12.6500
1            124.2358           108.7063     32.1766     -3.5882
2            147.2743            81.3884      5.1295     -6.8756
3             87.4032            76.1254     12.4584     -7.8195
```

(continues on next page)

(continued from previous page)

```
4          113.6569           98.9915    43.3318    -31.3336
5           35.0748           35.0748   -30.0474      0.3716
6           32.0912           32.0912   -20.0697     -1.0697
7           45.6125           42.6698    -2.1278     -5.4141
8           34.5922           29.1871   -18.4050      4.9190
9          111.2489           97.3428    44.3826      1.5235
10          71.8003           40.7515     2.2271      0.0595
11          85.5989           50.8967    15.4483     -7.3135
12         111.9817           90.9852    23.4994     -2.4994
13          71.0687           52.2564     8.0903      0.9097
14          80.1844           63.1756    21.5702     -0.4298
15         143.2905          117.2377    18.0264     18.3421
16          31.6944           26.7422     6.0286     -0.0095
17          22.6684           24.2317    -4.9083     -6.0917
18          92.5666          118.4852    24.8920     22.4867
19          56.4143           58.3596     1.9453     -1.9453
20         100.8888           77.1503    33.9020     -1.0327
21          79.2852           66.8969   -10.5670      5.4777
22          71.8164           56.5826     4.8237      4.1763
23         141.1083          103.7561    -0.7512    -11.4507
24          93.4423           78.8419    -0.6004      0.9201
25          58.9596           63.0258    -1.9669     -0.0331
26          25.6879           28.4402    -4.5871      3.7523
27          41.7041           23.0470     1.0975     -0.9025
28          37.8486           36.5870   -11.7849     -2.2616
29          40.7117           32.0759   -13.6358      9.7011
30          82.0364           87.6941    -2.8288      0.1712
31          95.2788           95.2788   -18.5279      3.1760
32          73.4694           83.9651   -32.1195    -18.8717
33          89.8632           83.8723    -3.9863      9.9954
34          75.8879           75.8879   -22.7664      1.4704

    SAV-outlier-percent  LSR-uncertainty-nm  LSR-uncertainty-sh
0               25.5099              1.0753              0.8147
1               30.4657              2.0622              1.4141
2               15.3570              2.5138              1.7745
3               24.2239              2.6042              2.1307
4               25.0583              3.3615              2.4011
5               24.1731              1.0058              0.9084
6               31.8074              1.7956              1.4147
7               26.3946              1.6979              1.3892
8               30.1054              1.4017              1.0619
9               21.8180              1.2218              0.8727
10              19.0379              1.3350              1.1264
11              15.4072              1.7981              1.3221
12              29.5978              3.6587              3.0870
13              15.2372              1.5157              1.3690
14              15.5889              1.2131              1.0236
15              28.0330              5.2090              4.4198
16              39.4703              0.9104              1.0046
17              35.2196              0.8499              0.8792
18              28.0406              3.2260              2.3721
19              15.9744              1.9864              1.6761
20              26.7112              1.1858              0.9068
21              23.0105              1.1267              0.9155
22              25.0130              1.7165              1.5504
```

(continues on next page)

(continued from previous page)

```
23              22.5063              1.4392              1.4392
24              30.9668              1.1808              1.1414
25              22.2547              2.3483              1.9814
26              32.3215              1.2633              1.1411
27              25.7202              1.5543              1.3036
28              30.6416              0.8849              0.9481
29              23.3886              0.6939              0.7178
30              25.8497              1.7262              1.5104
31              25.0593              1.8709              1.3757
32              20.5361              1.3447              1.2579
33              26.1195              2.0611              1.6863
34              31.5486              2.1947              1.9823
```

7.1 Abbreviations in Table S4

- LS8: Landsat 8
- Sen2: Sentinel-2
- Assigned-x-error: the value of the `Vx_err` flag that comes with each velocity map (m/yr)
- Assigned-y-error: the value of the `Vy_err` flag that comes with each velocity map (m/yr)
- SAV-uncertainty-x: correct-match uncertainty of the static terrain velocity (Metric 1; x component, m/yr)
- SAV-uncertainty-y: correct-match uncertainty of the static terrain velocity (Metric 1; y component, m/yr)
- SAV-peak-x: correct-match peak location (x component, m/yr)
- SAV-peak-y: correct-match peak location (y component, m/yr)
- SAV-outlier-percent: amount of incorrect matches (in percent of all the static terrain pixels)
- LSR-uncertainty-nm: variability of longitudinal normal strain rate (1/yr)
- LSR-uncertainty-sh: variability of longitudinal shear strain rate (Metric 2; 1/yr)

CHAPTER

EIGHT

FIGURES S1-S8: ALL 172 TEST VELOCITY MAPS

The label of each panel in Figures S1-S8 indicates the corresponding parameter combination, formatted as (Software)-(Template size)-(Pixel spacing)-(Prefilter). For Vmap results, the subpixel method is also shown in the label. See Table S1 for parameter abbrevations.

To reproduce these figures, see bottom of this page.

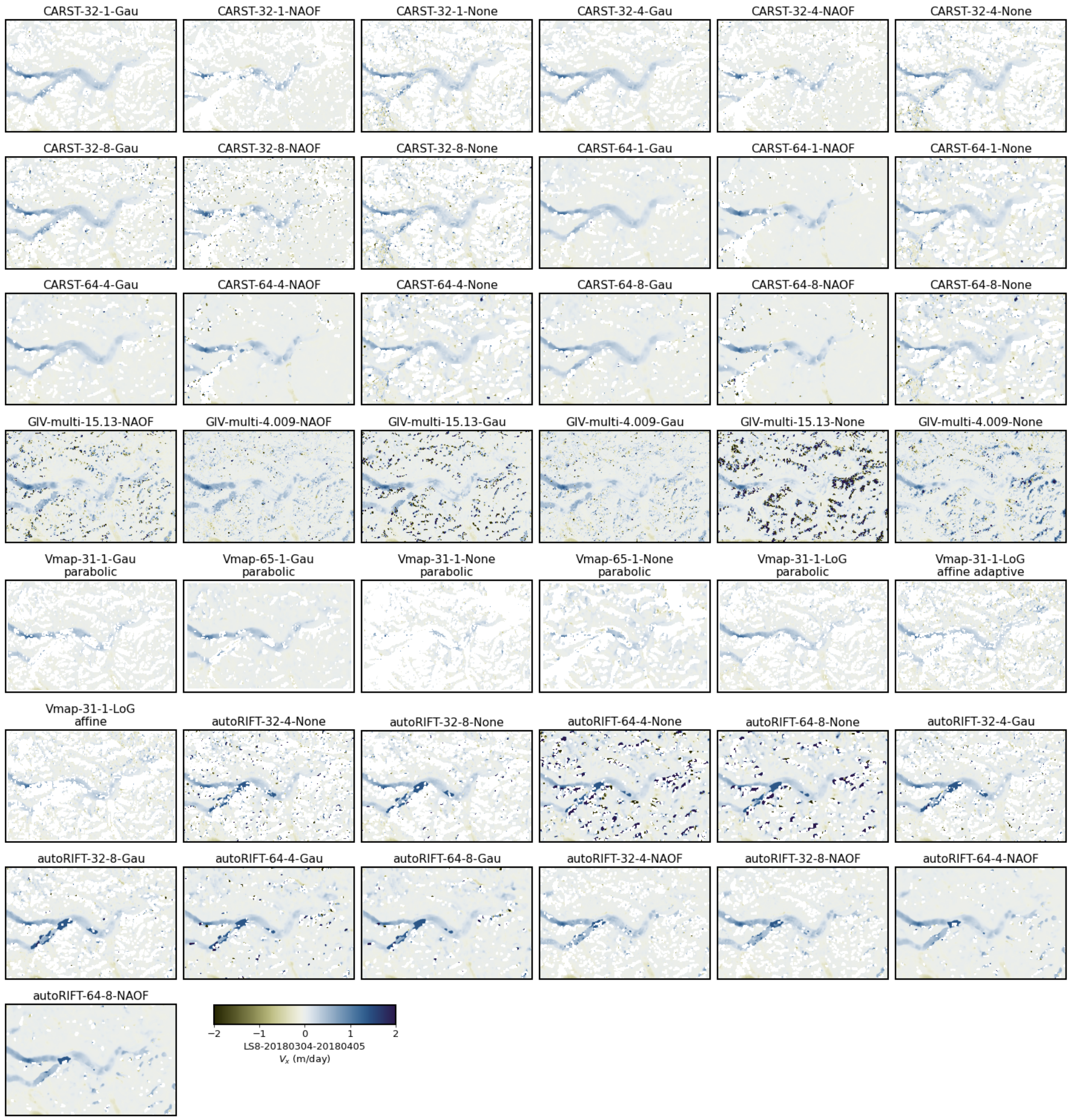


Figure S1. V_x (positive toward image east) of the pair `LS8-20180304-20180405`.

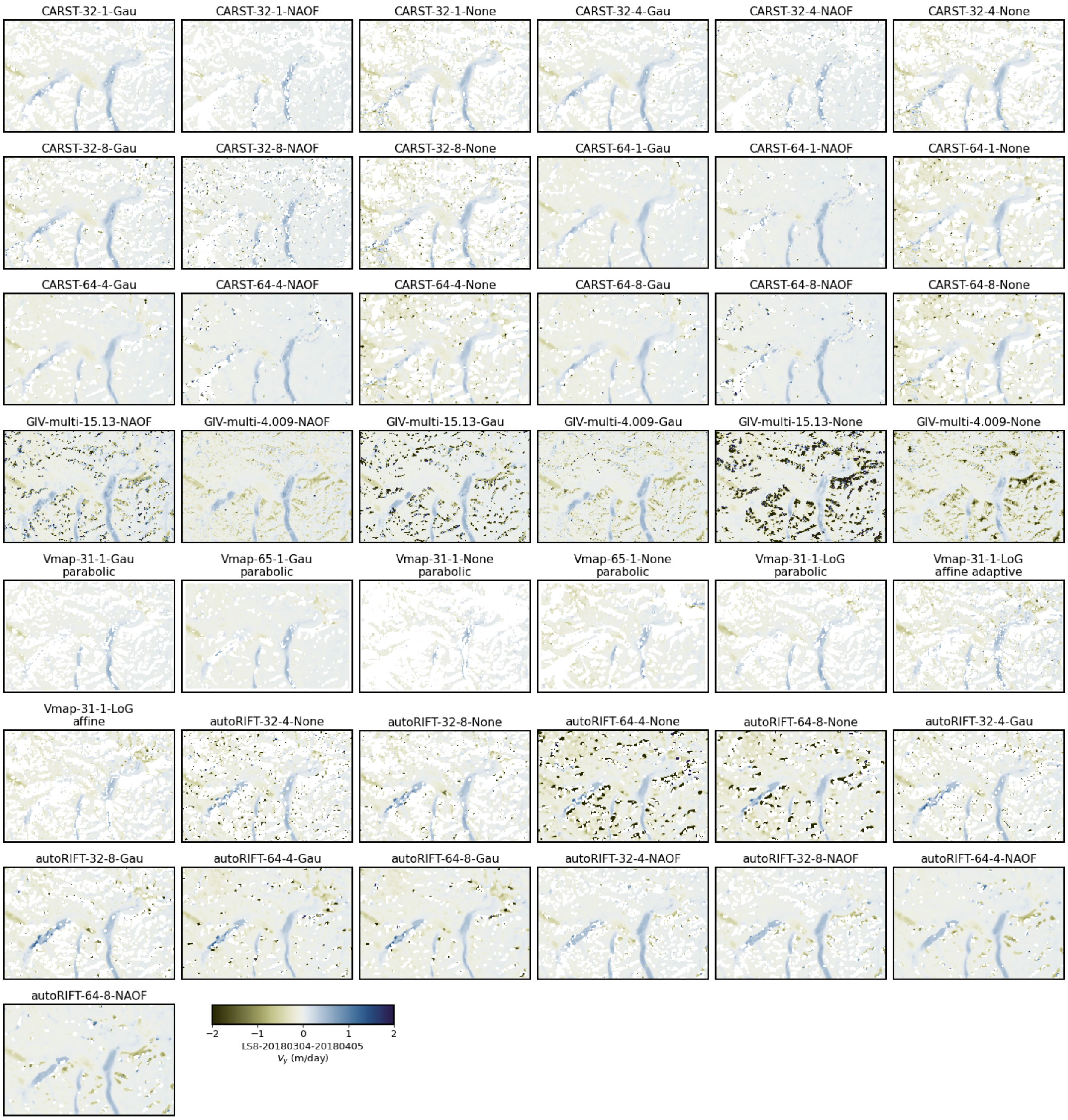


Figure S2. V_y (positive toward image north) of the pair `LS8-20180304-20180405`.

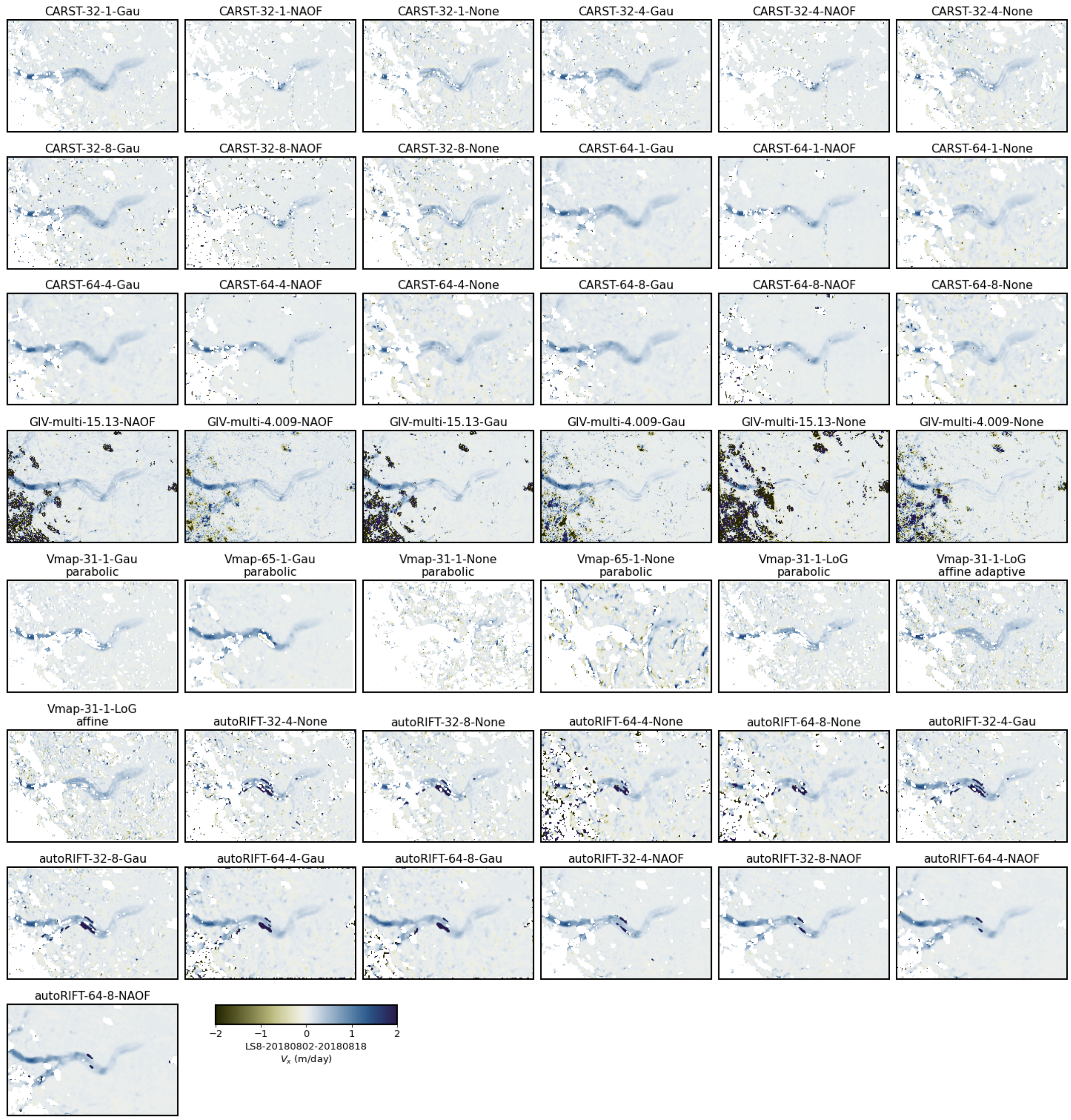


Figure S3. V_x (positive toward image east) of the pair `LS8-20180802-20180818`.

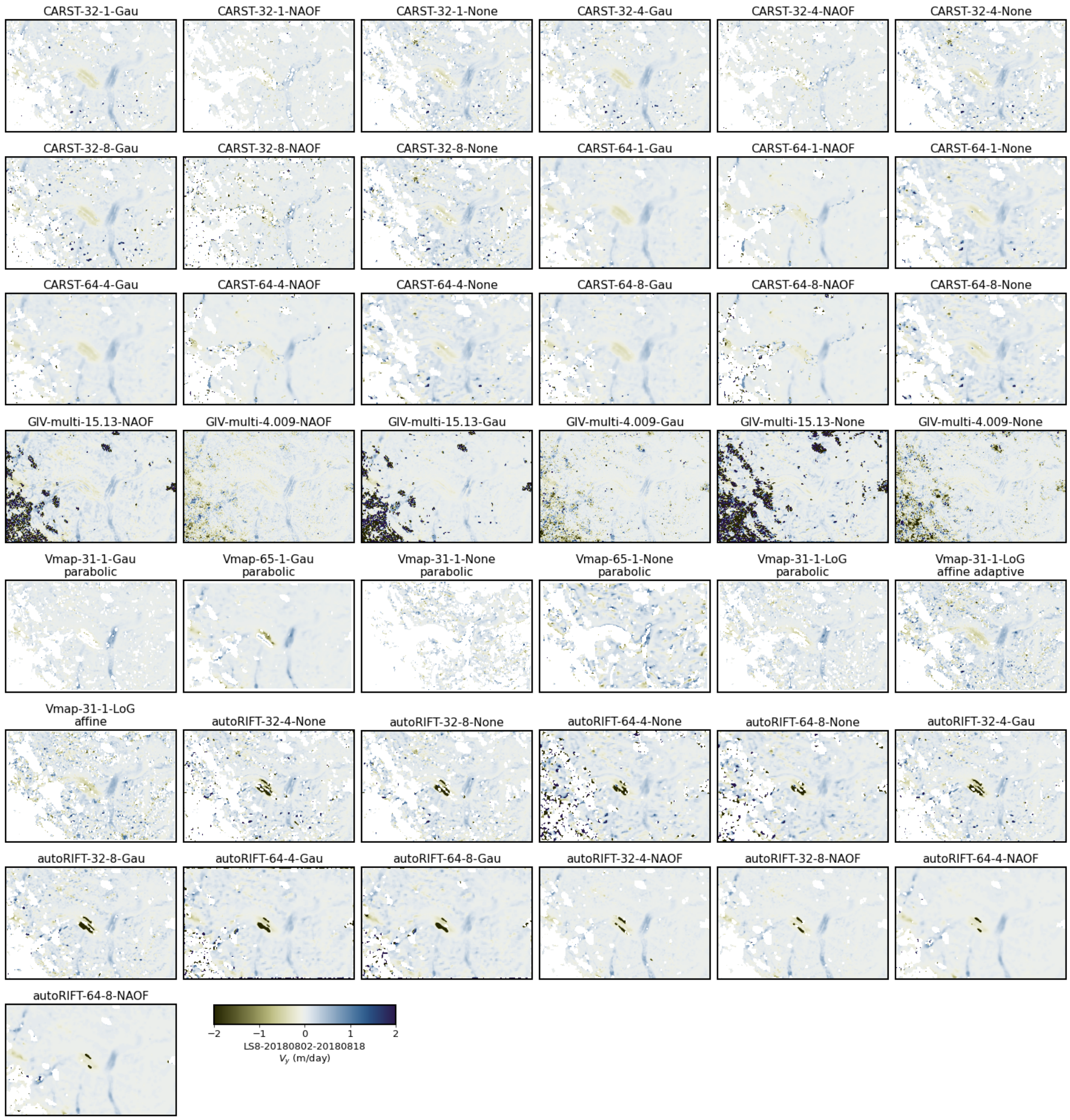


Figure S4. V_y (positive toward image north) of the pair `LS8-20180802-20180818`.

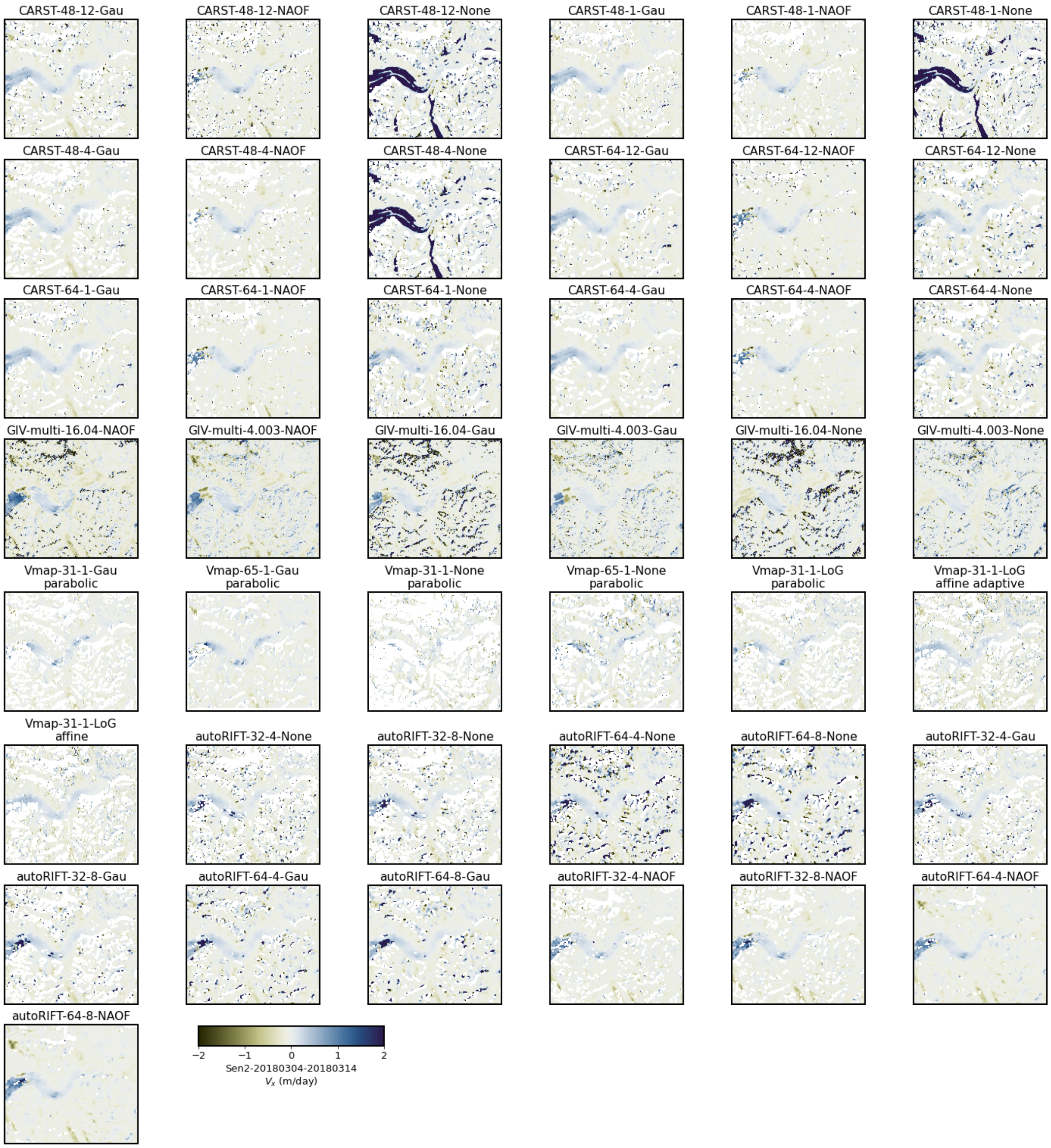


Figure S5. V_x (positive toward image east) of the pair `Sen2-20180304-20180314`.

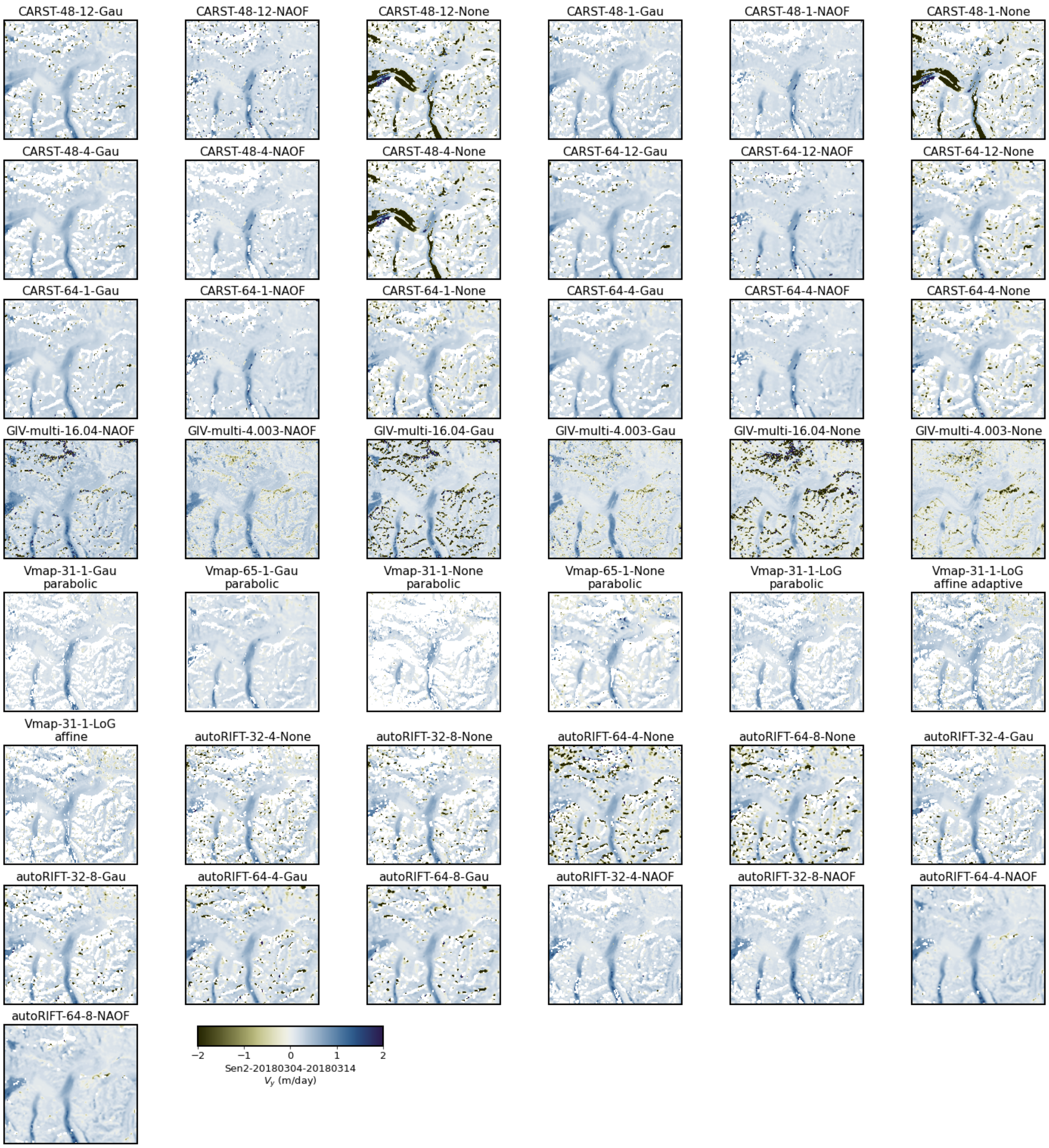


Figure S6. V_y (positive toward image north) of the pair `Sen2-20180304-20180314`.

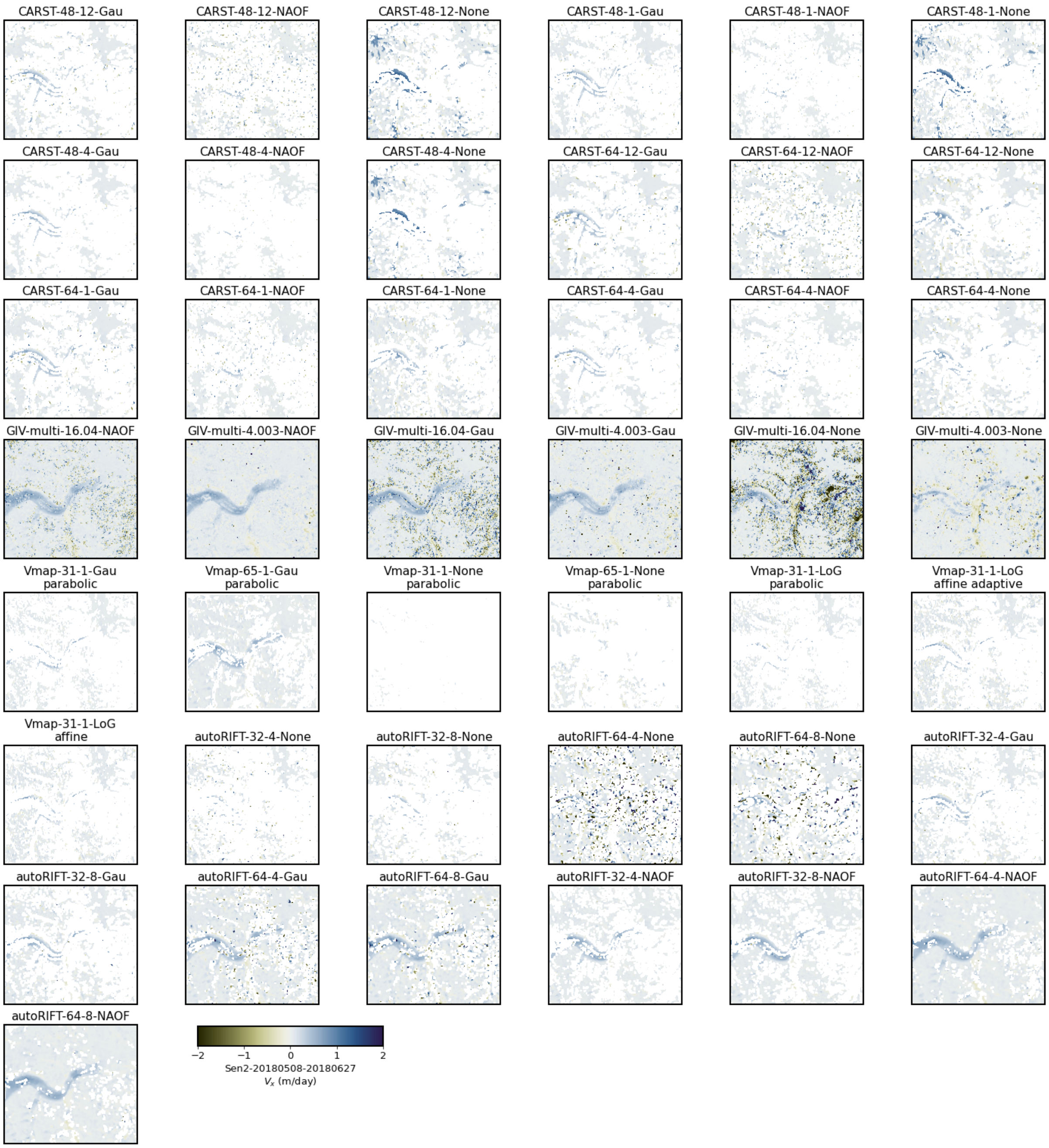


Figure S7. V_x (positive toward image east) of the pair `Sen2-20180508-20180627`.

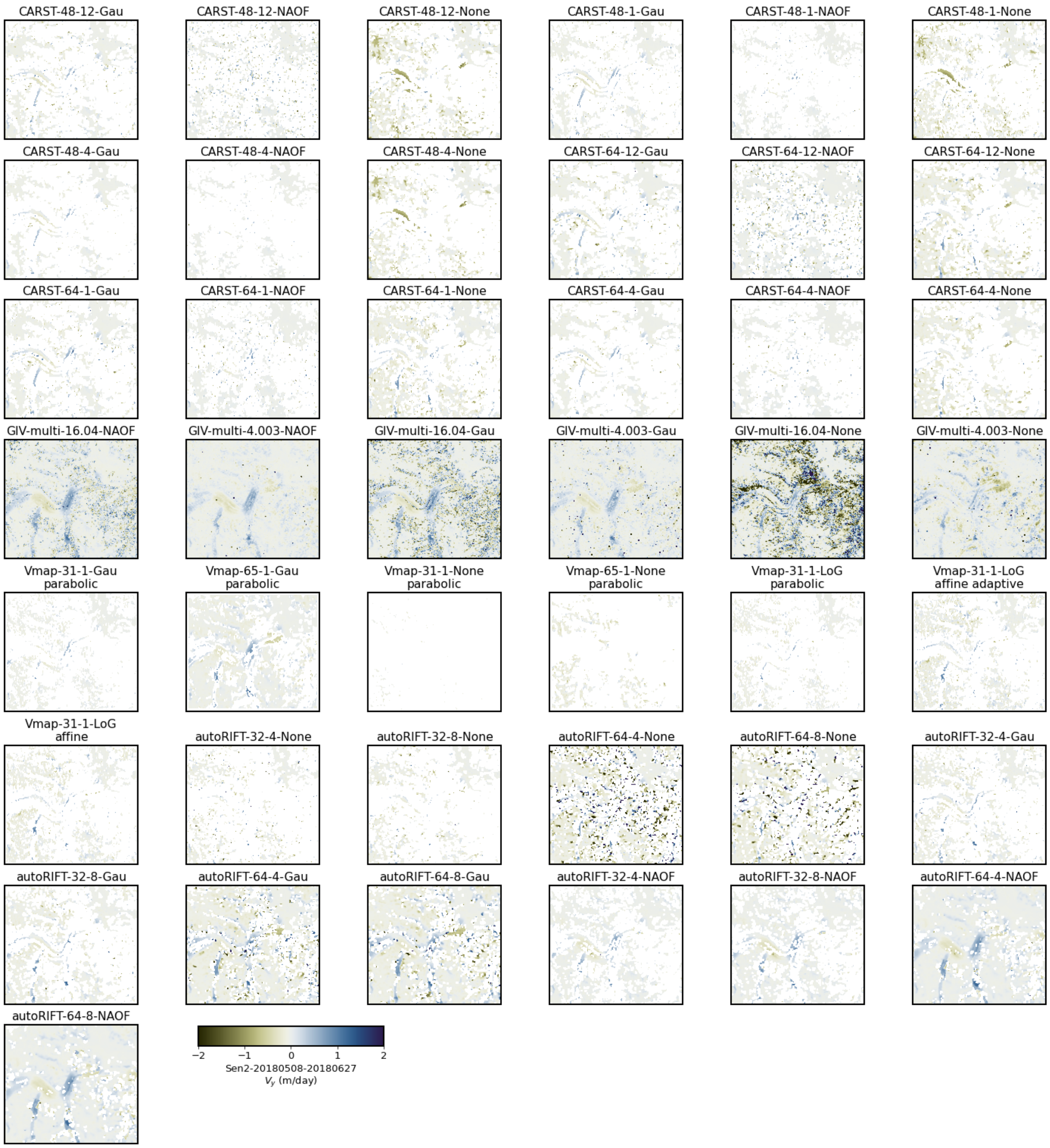


Figure S8. V_y (positive toward image north) of the pair `Sen2-20180508-20180627`.

8.1 Code for reproducing the figures

```
import pandas as pd
import glaft
import matplotlib as mpl
import matplotlib.pyplot as plt
```

```
#### Font and line width settings ####
font = {'size'    : 13}
mpl.rc('font', **font)
axes_settings = {'linewidth'    : 2}
mpl.rc('axes', **axes_settings)

def plot_batch(sub_df, component: str='Vx', datestr: str=''):
    """
    Plot all Vx or Vy maps from the same image pair.
    """
    fig, axs = plt.subplots(8, 6, figsize=(20, 21), constrained_layout=True)
    n = 0
    for idx, row in sub_df.iterrows():
        templatesize = row['Template size (px)']
        # change long GIV label "varying: multi-pass" to "multi"
        templatesize = 'multi' if templatesize == 'varying: multi-pass' else templatesize
        if row.Software == 'Vmap':
            label = '-'.join((row.Software, templatesize, row['Pixel spacing (px)'], row.Prefilter)) + '\n' + row.Subpixel
        else:
            label = '-'.join((row.Software, templatesize, row['Pixel spacing (px)'], row.Prefilter))
        ax_sel = axs[n // 6, n % 6]
        glaft.show_velocomp(row[component], ax=ax_sel)
        ax_sel.set_title(label)
        n += 1

    # delete empty axes
    for i in range(n, 48):
        ax_sel = axs[i // 6, i % 6]
        fig.delaxes(ax_sel)

    # add a colorbar in the bottom
    if component == 'Vx':
        cbar_label = '$V_x$ (m/day)'
    elif component == 'Vy':
        cbar_label = '$V_y$ (m/day)'
    cbar_label = datestr + '\n' + cbar_label
    cax = fig.add_axes([0.2, 0.09, 0.17, 0.017])

    mappable = glaft.prep_colorbar_mappable()
    fig.colorbar(mappable, cax=cax, orientation='horizontal', label=cbar_label)

    return fig, axs
```

To reproduce the figures:

1. download the source velocity maps from https://doi.org/10.17605/OSF.IO/HE7YR

2. locate `notebooks/manifest.csv`
3. update the `Vx` and `Vy` columns with the downloaded file paths
4. uncomment and run the cell below.

```
# df = pd.read_csv('../manifest.csv', dtype=str)
# datestrs = ['LS8-20180304-20180405',
#             'LS8-20180802-20180818',
#             'Sen2-20180304-20180314',
#             'Sen2-20180508-20180627']

# for datestr in datestrs:
#     sub_df = df.loc[df['Date'] == datestr]
#     for component in ['Vx', 'Vy']:
#         fig, axs = plot_batch(sub_df, component=component, datestr=datestr)
#         fig.patch.set_facecolor('xkcd:white')
#         fig.savefig('figs/{}-{}.png'.format(datestr, component))
```

CHAPTER

NINE

FIGURES S9-S16: STATIC AREA VELOCITY ANALYSIS FOR ALL TESTS

This notebook shows the analysis of static area velocity (abbreviated as **SAV** in Table S2) with the supplemental figures in the bottom.

9.1 Basic information, importing modules, load data list and static-area shapefile

See Table S1 for all the Kaskawulsh glacier images and parameter sets used in this study.

```
import glaft
import matplotlib as mpl
import matplotlib.pyplot as plt
import pandas as pd
```

We start by loading the data list. Whichever line in the cell blow works for reproducing the figures.

- `../manifest.csv` contains only the parameter table (**Table S1**)
- `../results_2022.csv` contains both the parameter table and all the metrics calculated (**Table S2**) in this study.

If you want to reproduce the workflow and the figures, make sure you have downloaded all necessary input files from https://doi.org/10.17605/OSF.IO/HE7YR and have updated the `Vx` and `Vy` columns in either csv file with the downloaded file paths before starting the analysis.

```
# df = pd.read_csv('../manifest.csv', dtype=str)
df = pd.read_csv('../results_2022.csv', dtype=str)
```

Specify static area. Change the path to the downloaded shapefile from https://doi.org/10.17605/OSF.IO/HE7YR before running the cell.

```
in_shp = '/home/jovyan/Projects/PX_comparison/shapefiles/bedrock_V2.shp'
```

9.2 Perform analysis

```
exps = {}

for idx, row in df.iterrows():
    exp = glaft.Velocity(vxfile=row.Vx, vyfile=row.Vy, static_area=in_shp, kde_
↪gridsize=60, thres_sigma=2.0)
    exp.static_terrain_analysis()
    exps[idx] = exp
```

9.3 Visualize results

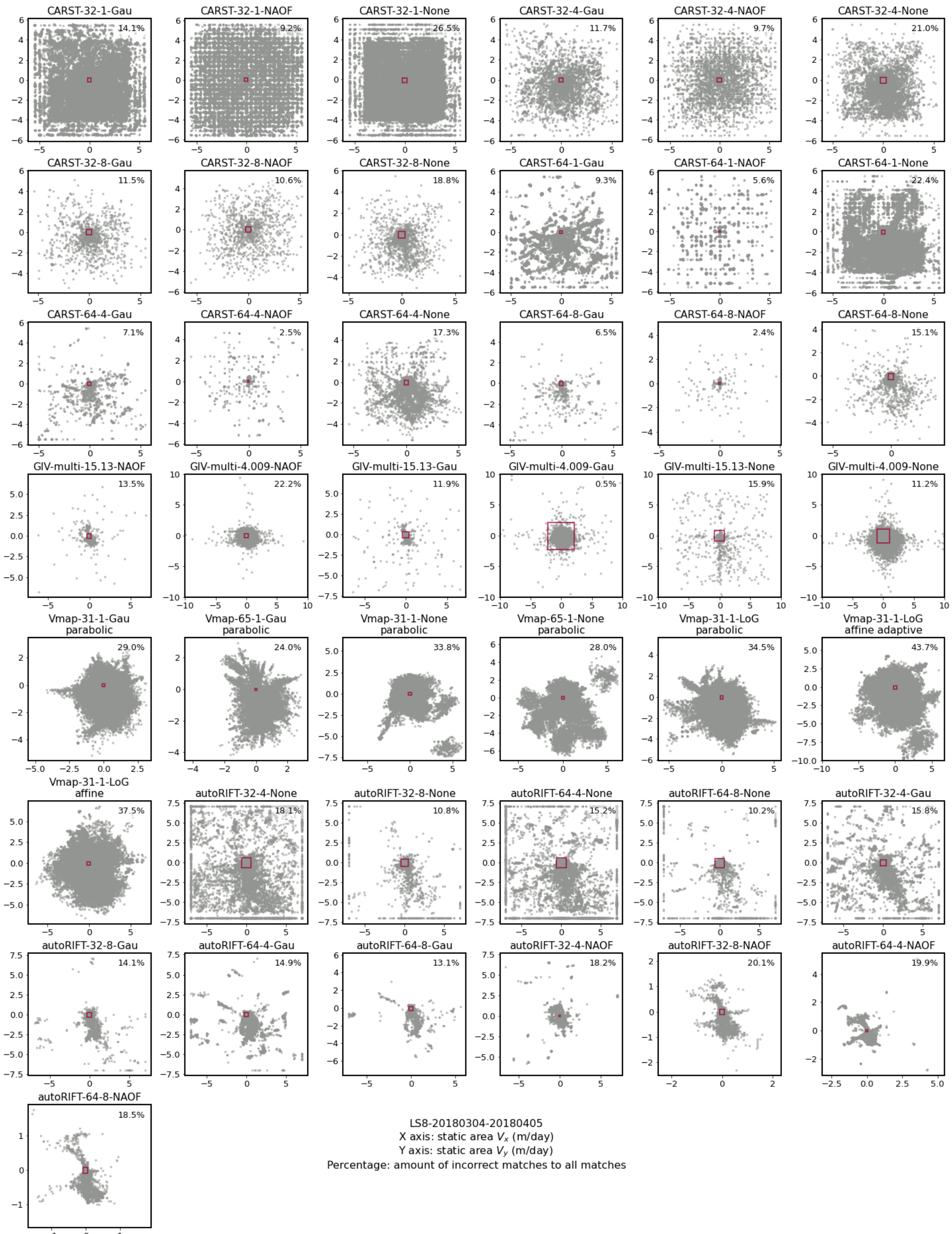


Figure S9. Static terrain velocity distribution of the pair `LS8-20180304-20180405` (full extent).

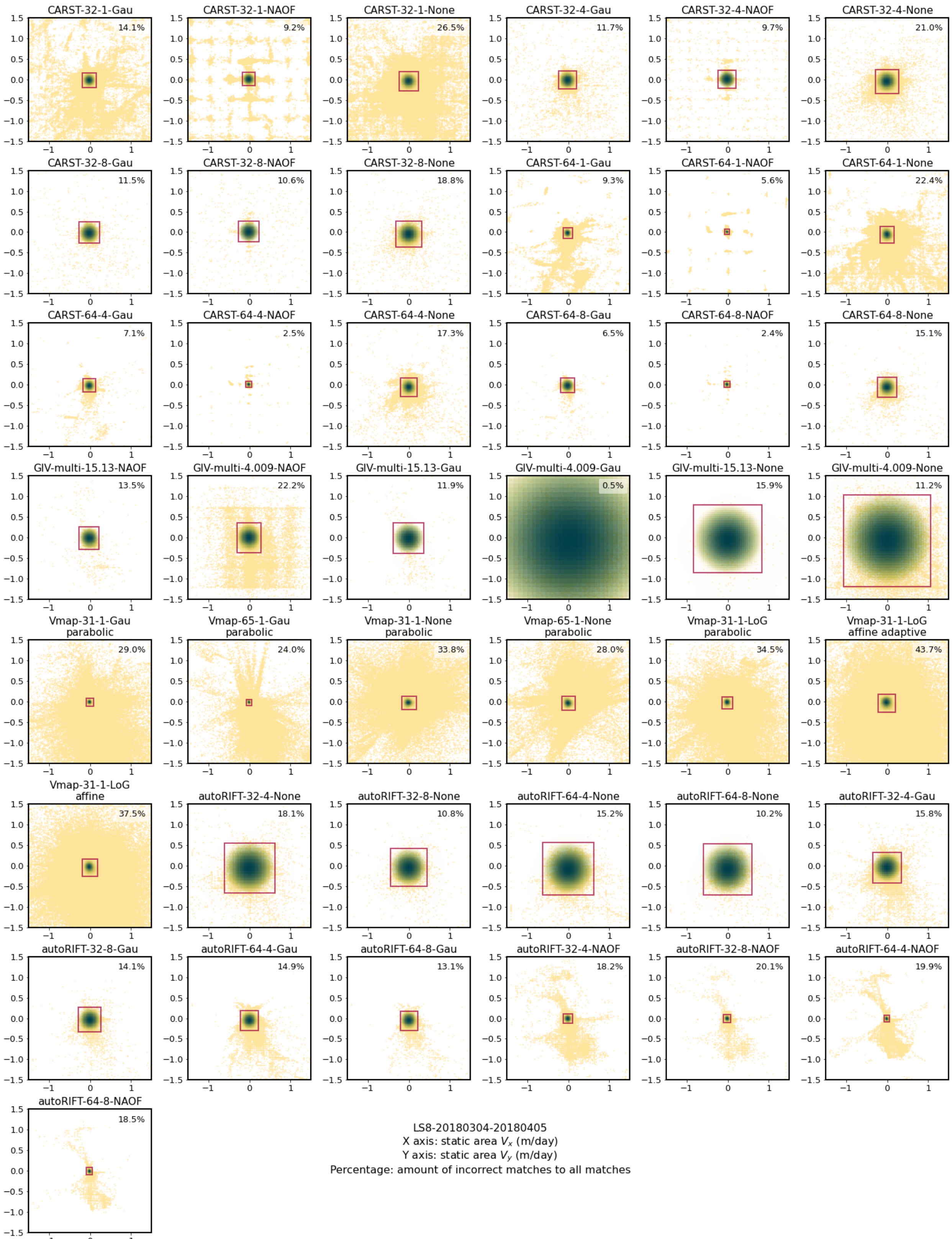


Figure S10. Static terrain velocity distribution of the pair `LS8-20180304-20180405` (zoomed with kernel density estimation).

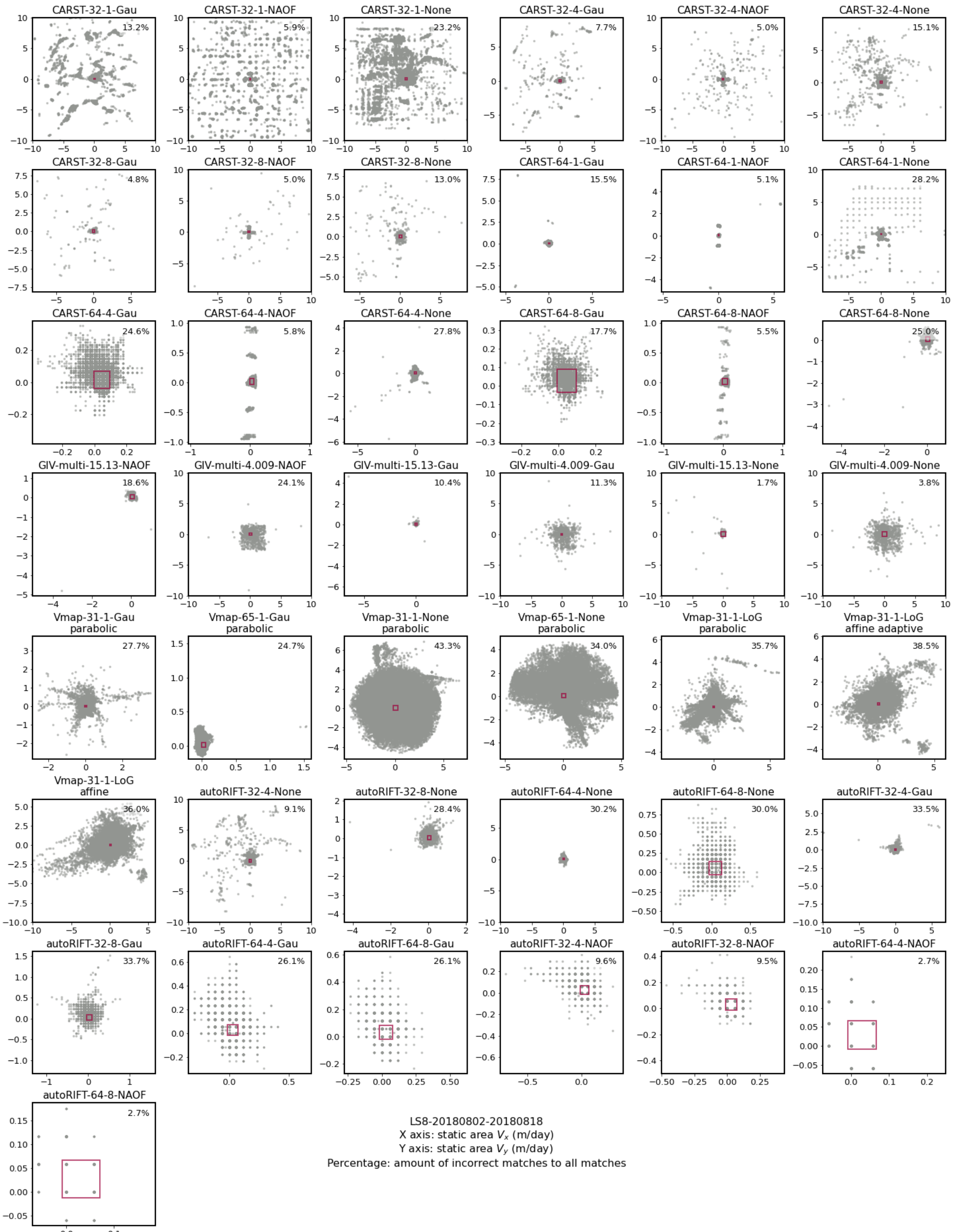


Figure S11. Static terrain velocity distribution of the pair `LS8-20180802-20180818`. (full extent).

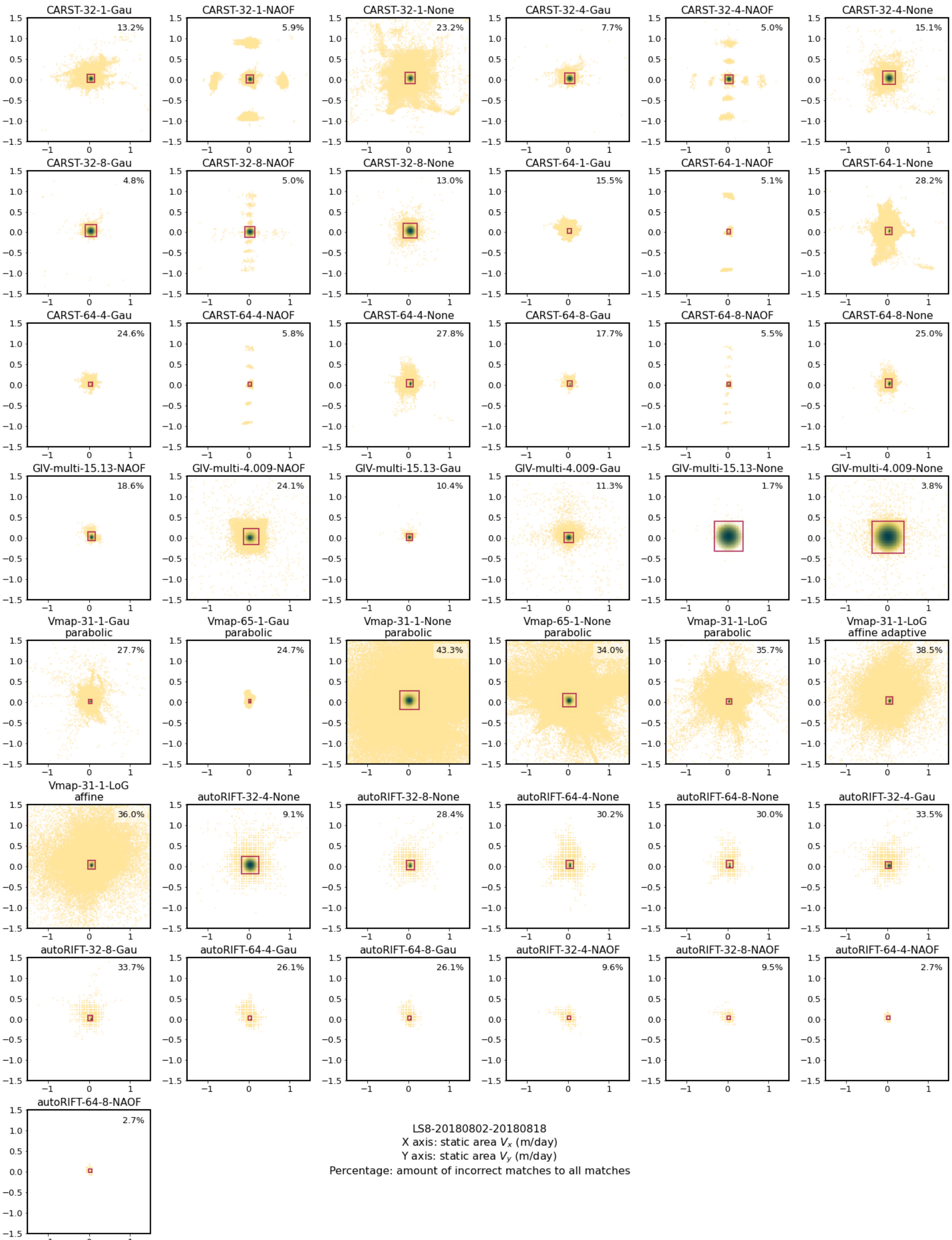


Figure S12. Static terrain velocity distribution of the pair `LS8-20180802-20180818`. (zoomed with kernel density estimation).

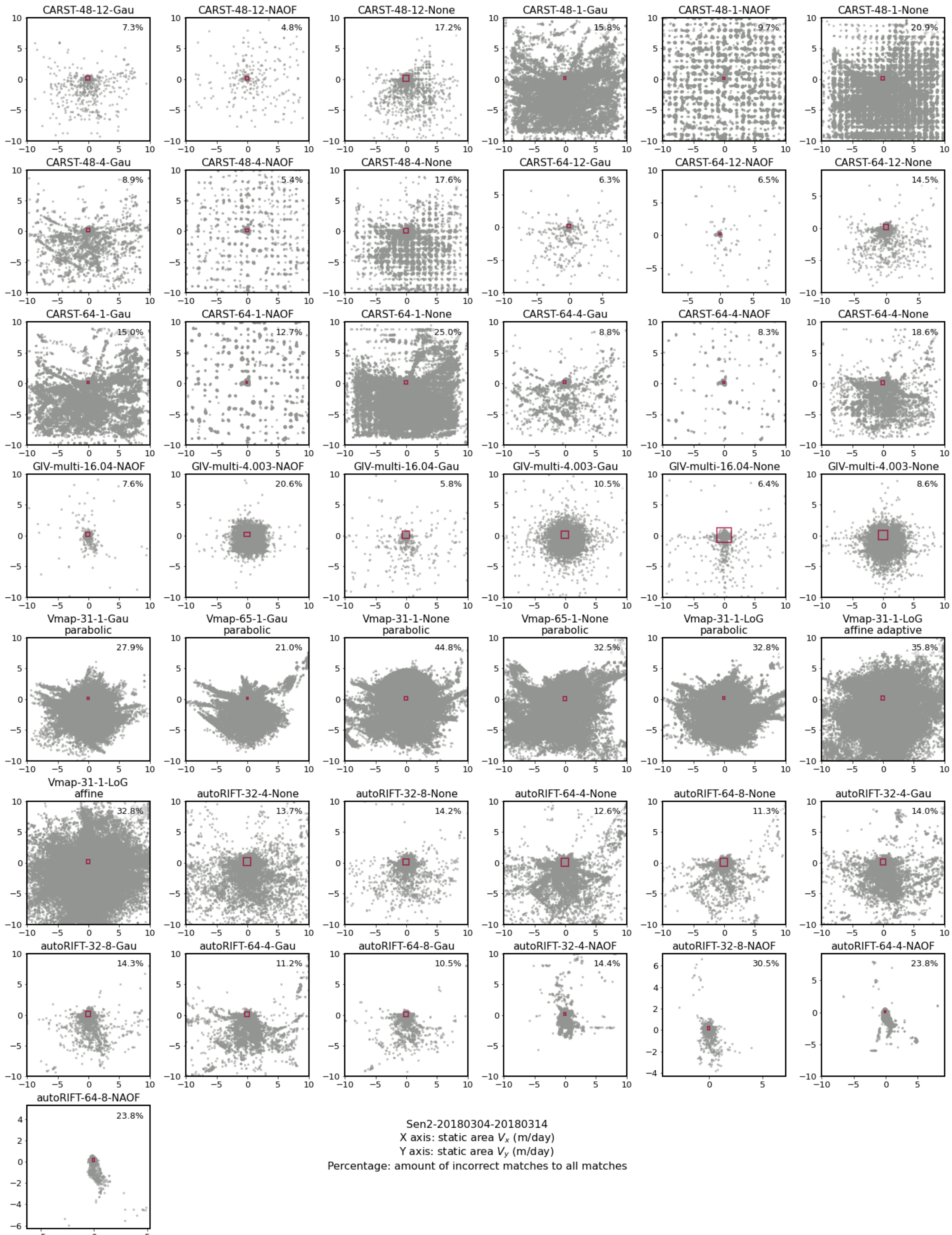


Figure S13. Static terrain velocity distribution of the pair `Sen2-20180304-20180314`. (full extent).

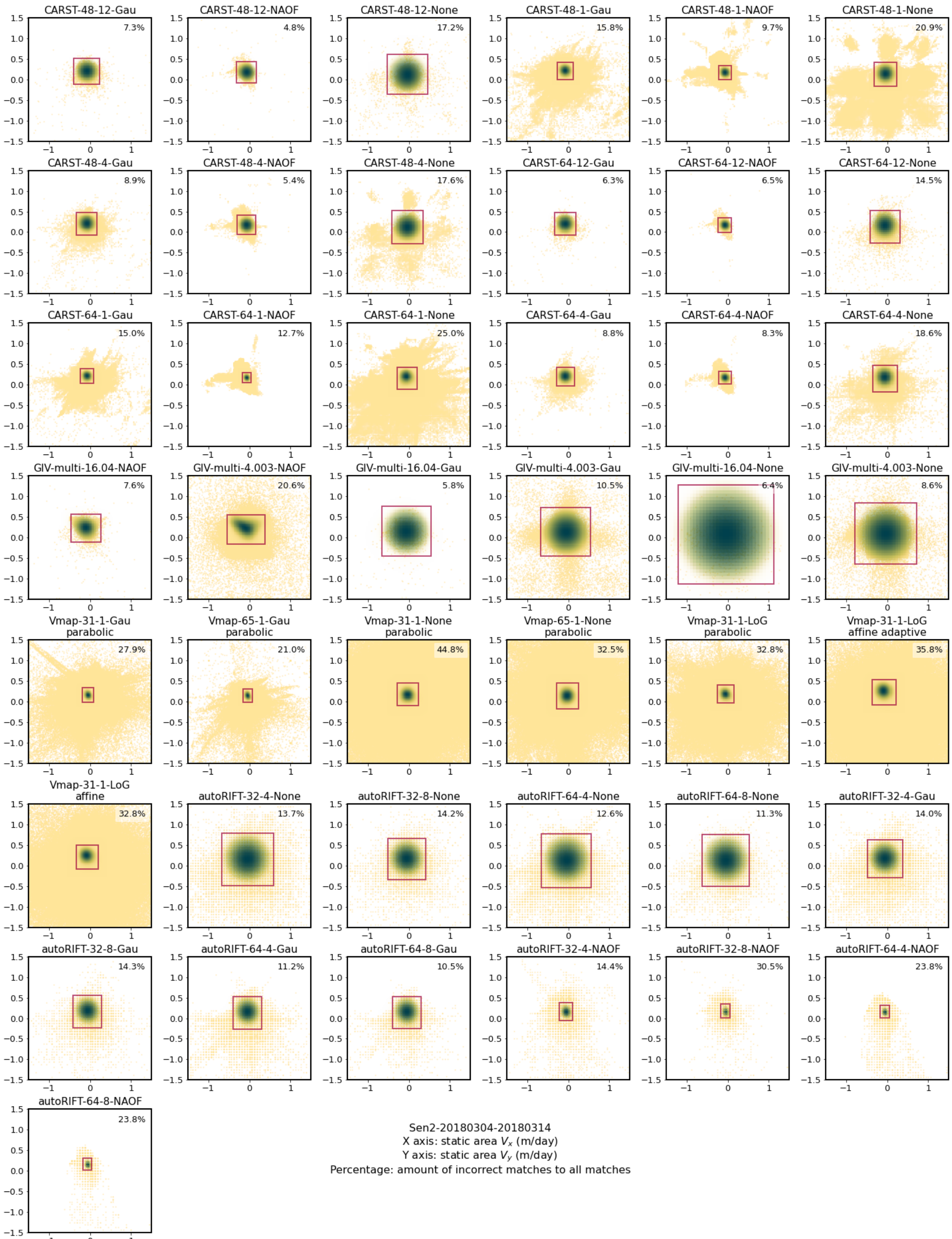


Figure S14. Static terrain velocity distribution of the pair `Sen2-20180304-20180314`. (zoomed with kernel density estimation).

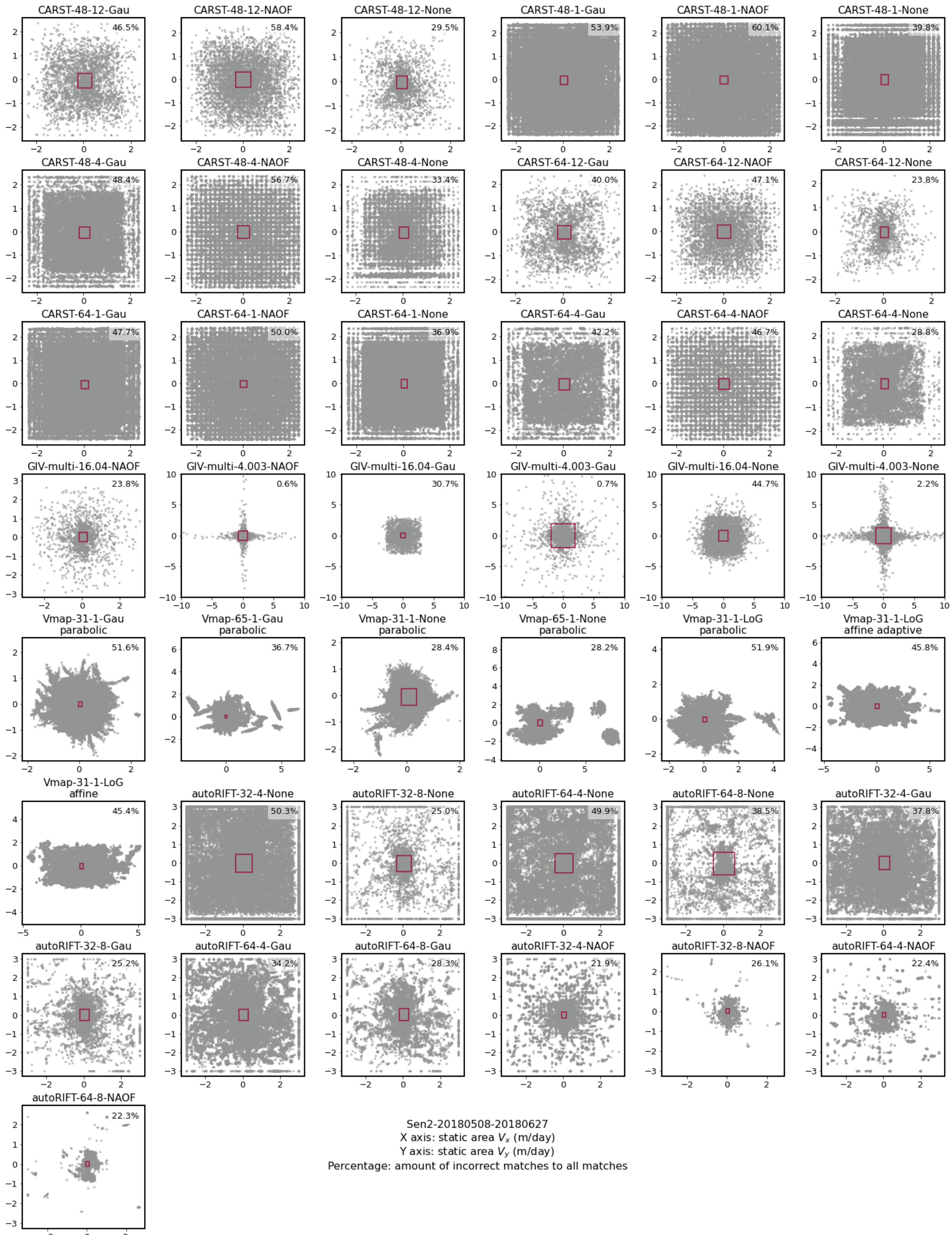


Figure S15. Static terrain velocity distribution of the pair `Sen2-20180508-20180627`. (full extent).

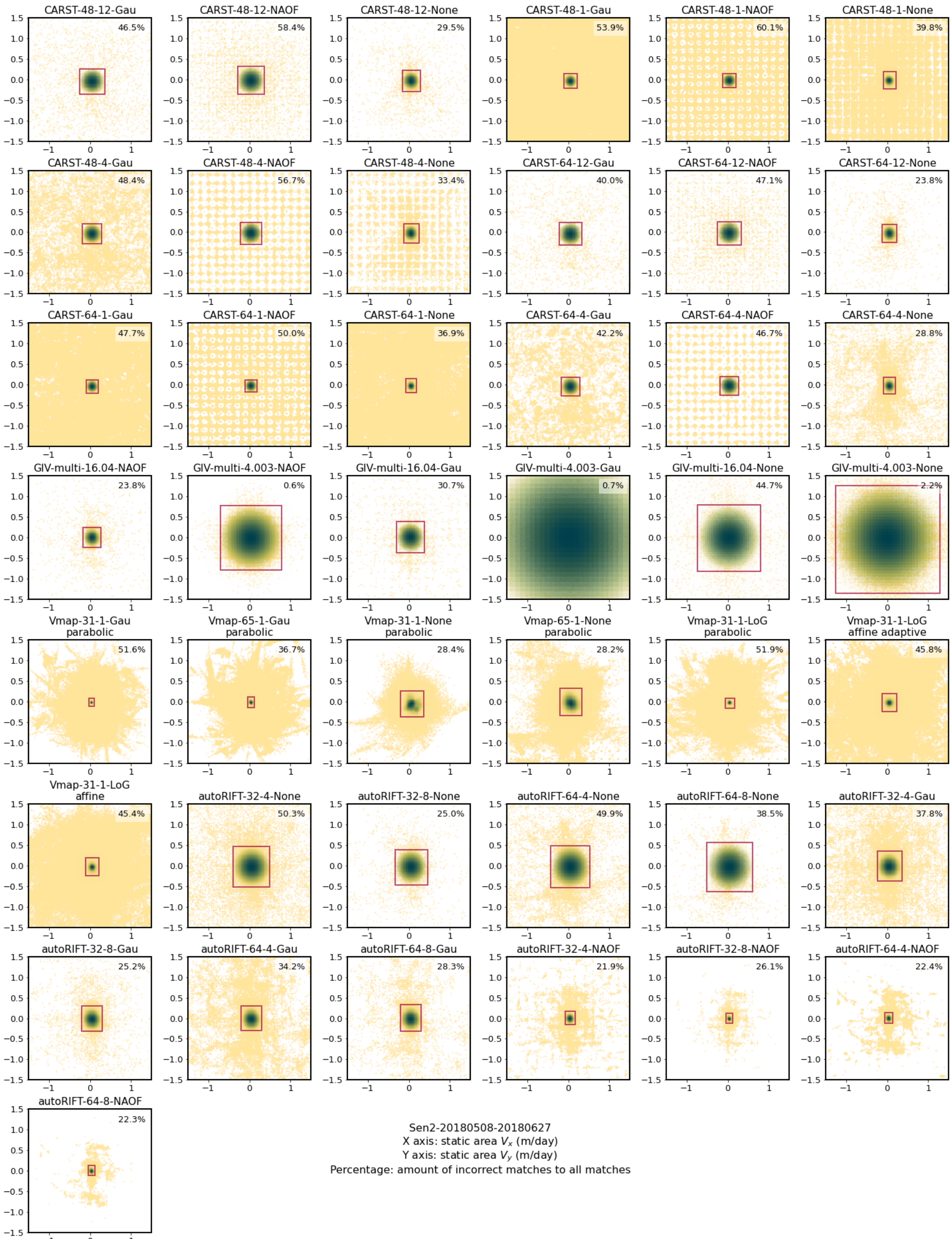


Figure S16. Static terrain velocity distribution of the pair `Sen2-20180508-20180627`. (zoomed with kernel density estimation).

9.4 Save results

```
for idx, exp in exps.items():
    df.loc[idx, 'SAV-uncertainty-x'] = exp.metric_static_terrain_x
    df.loc[idx, 'SAV-uncertainty-y'] = exp.metric_static_terrain_y
    df.loc[idx, 'SAV-peak-x'] = exp.kdepeak_x
    df.loc[idx, 'SAV-peak-y'] = exp.kdepeak_y
    df.loc[idx, 'SAV-outlier-percent'] = exp.outlier_percent * 100

df.to_csv('../results_2022.csv', index=False)
```

CHAPTER

TEN

FIGURES S17-S28: LONGITUDINAL STRAIN RATE ANALYSIS FOR ALL TESTS

This notebooks shows the analysis of longitudinal strain rate (abbreviated as **LSR** in Table S2) with the supplemental figures in the bottom.

10.1 Basic information, importing modules, load data list and flow-area shapefile

See Table S1 for all the Kaskawulsh glacier images and parameter sets used in this study.

```
import glaft
import matplotlib as mpl
import matplotlib.pyplot as plt
import pandas as pd
from cmcrameri import cm as cramericm
```

We start by loading the data list. Whichever line in the cell blow works for reproducing the figures.

- `../manifest.csv` contains only the parameter table (**Table S1**)
- `../results_2022.csv` contains both the parameter table and all the metrics calculated (**Table S2**) in this study.

If you want to reproduce the workflow and the figures, make sure you have downloaded all necessary input files from https://doi.org/10.17605/OSF.IO/HE7YR and have updated the `Vx` and `Vy` columns in either csv file with the downloaded file paths before starting the analysis.

```
# df = pd.read_csv('../manifest.csv', dtype=str)
df = pd.read_csv('../results_2022.csv', dtype=str)
```

Specify flow area. Change the path to the downloaded shapefile from https://doi.org/10.17605/OSF.IO/HE7YR before running the cell.

```
in_shp = '/home/jovyan/Projects/PX_comparison/shapefiles/glacier_V1_Kaskawulsh_s_
↪inwardBuffer600m.shp'
```

10.2 Perform analysis

```
exps = {}

for idx, row in df.iterrows():
    exp = glaft.Velocity(vxfile=row.Vx, vyfile=row.Vy, on_ice_area=in_shp, kde_
↪gridsize=60, thres_sigma=2.0)
    exp.longitudinal_shear_analysis()
    exps[idx] = exp
```

10.3 Visualize results

10.3.1 3.1. Distribution of longitudinal strain rate

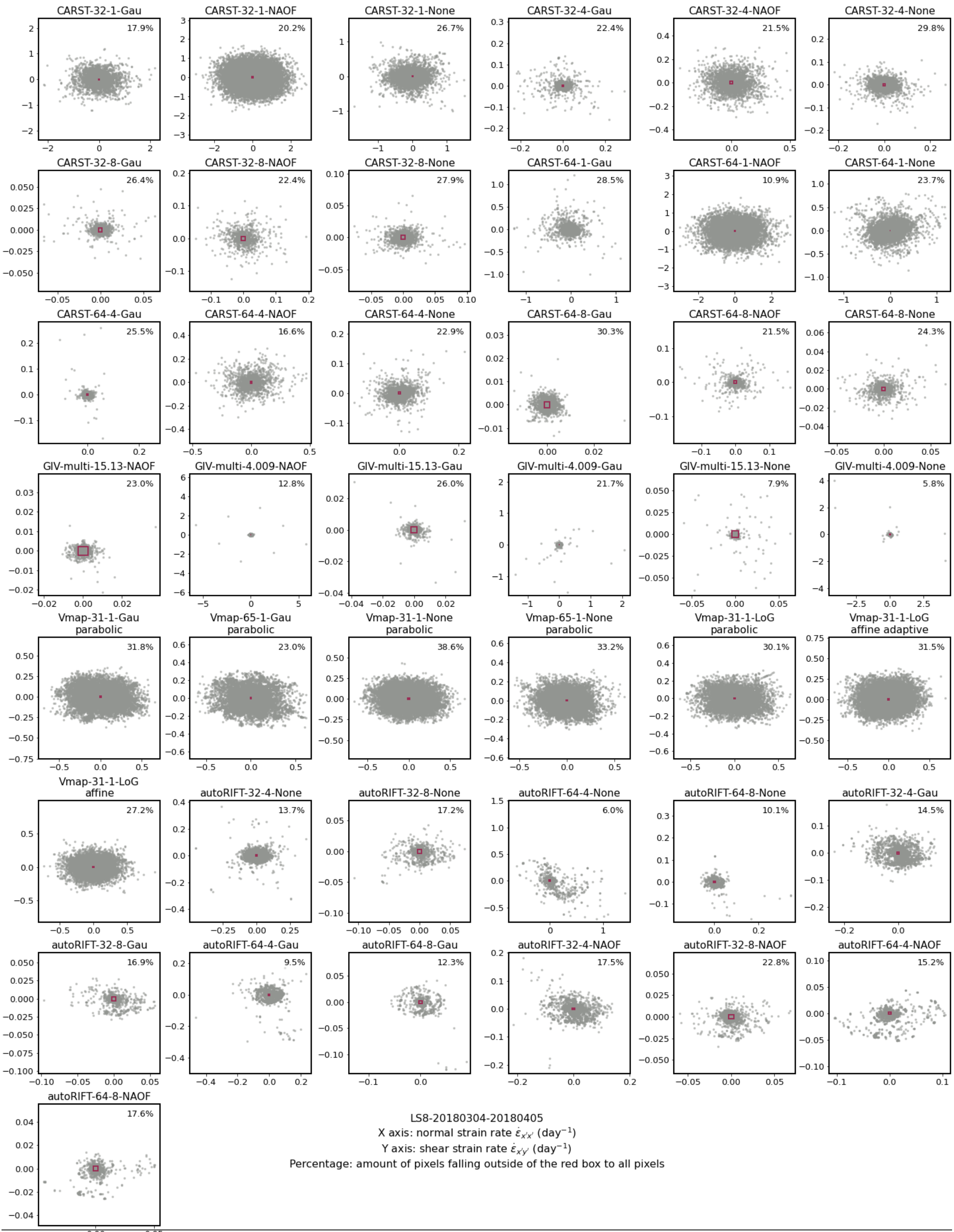


Figure S17. Longitudinal strain rate distribution of the pair `LS8-20180304-20180405` (full extent).

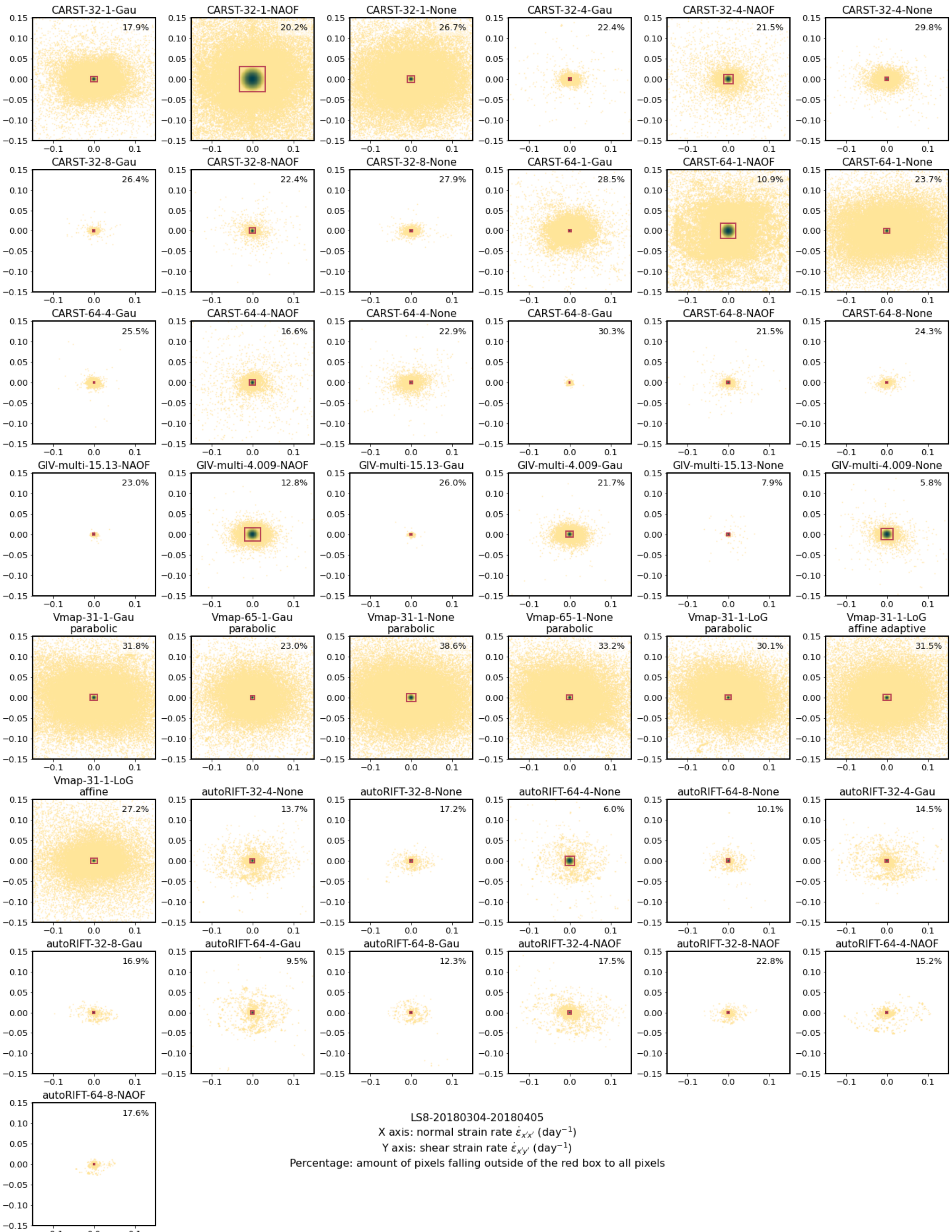


Figure S18. Longitudinal strain rate distribution of the pair `LS8-20180304-20180405` (zoomed with kernel density estimation).

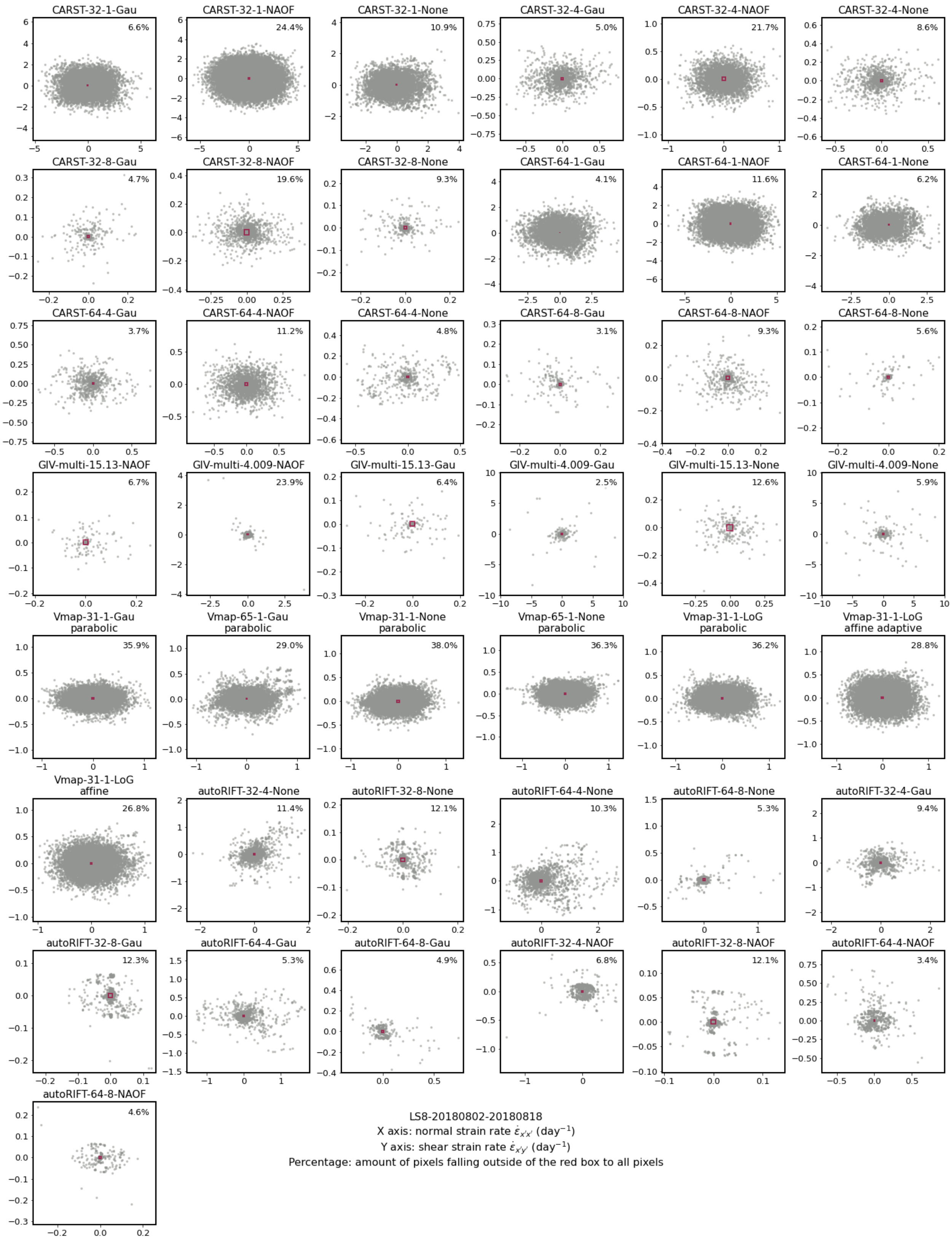


Figure S19. Longitudinal strain rate distribution of the pair `LS8-20180802-20180818`. (full extent).

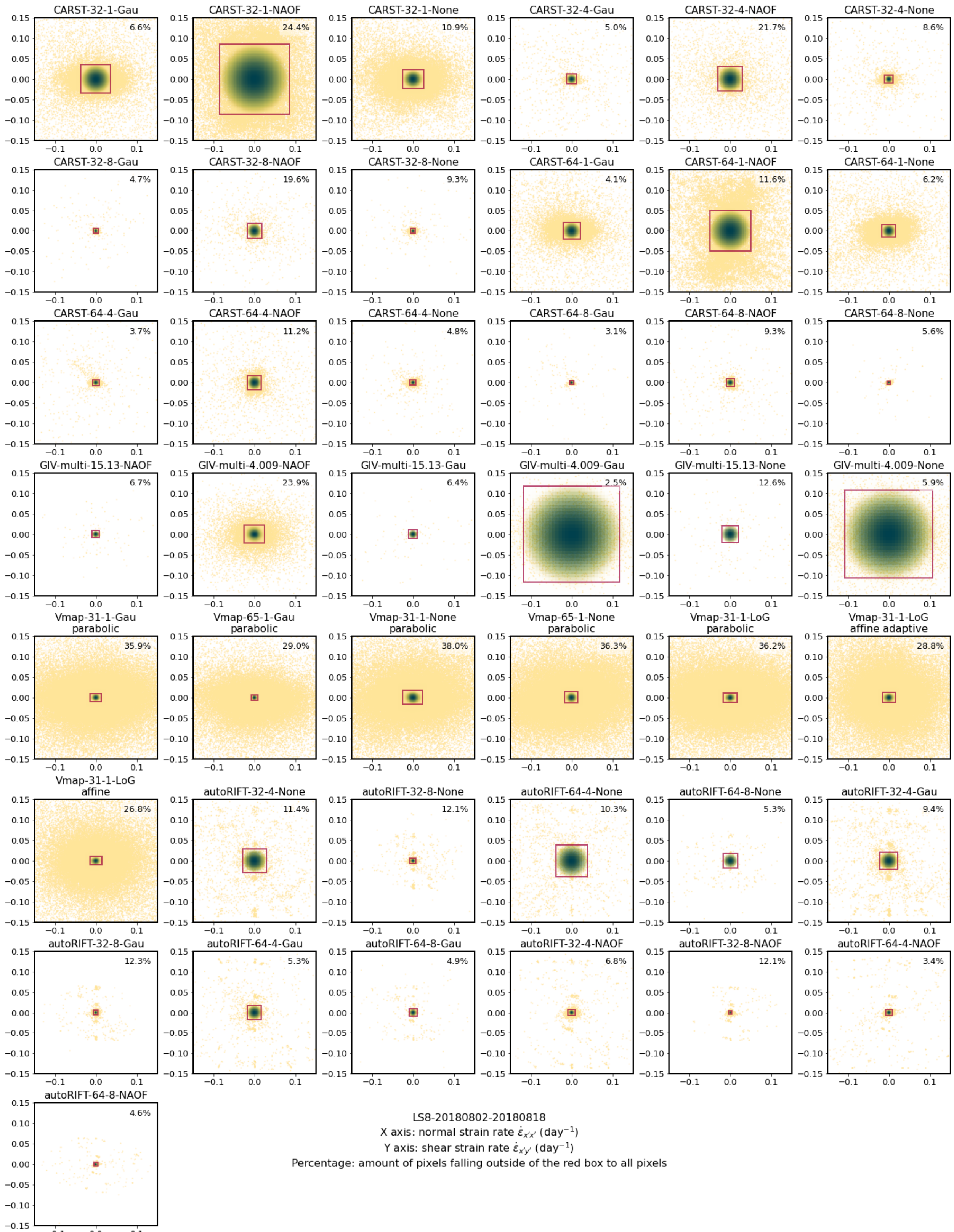


Figure S20. Longitudinal strain rate distribution of the pair `LS8-20180802-20180818`. (zoomed with kernel density estimation).

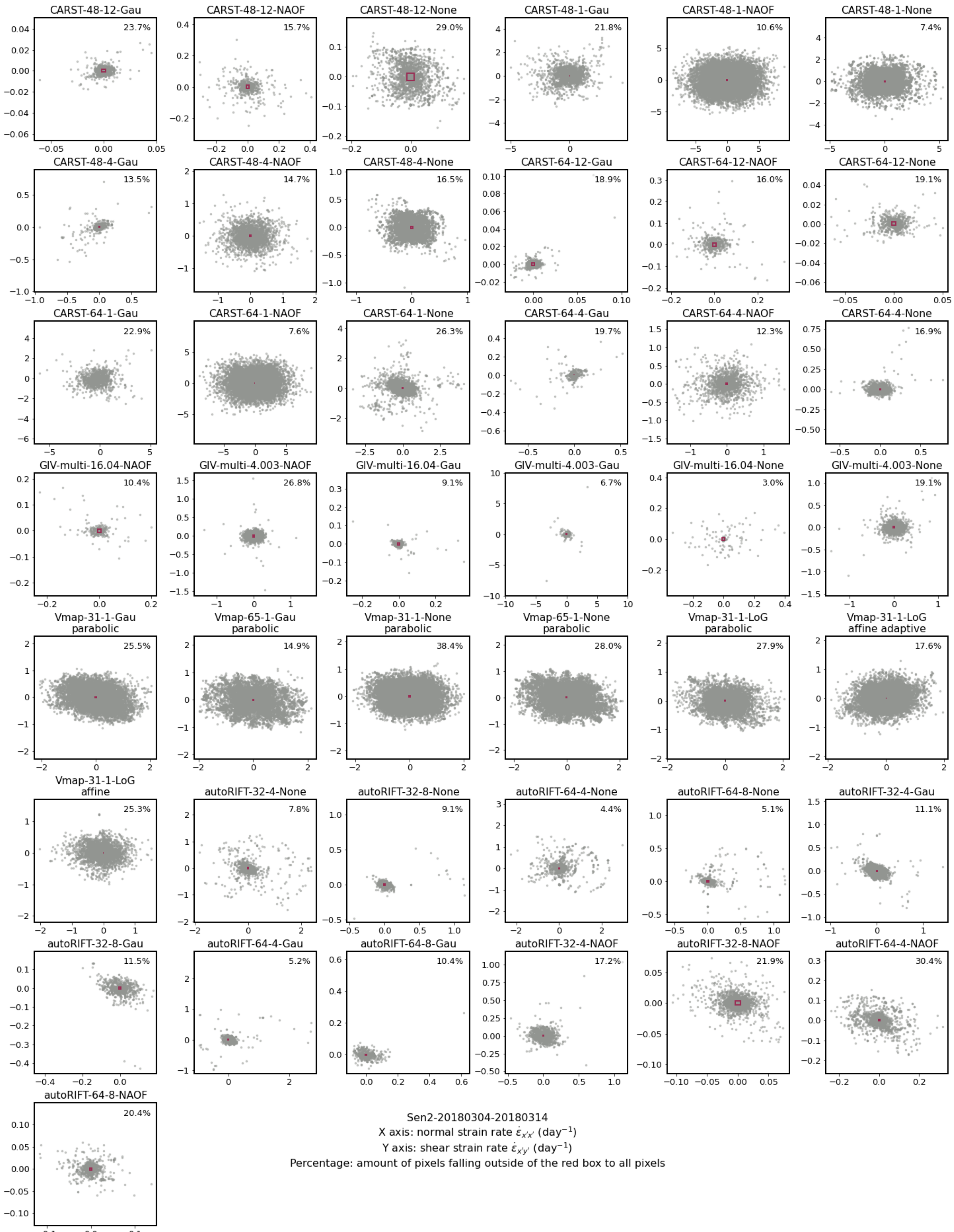


Figure S21. Longitudinal strain rate distribution of the pair `Sen2-20180304-20180314`. (full extent).

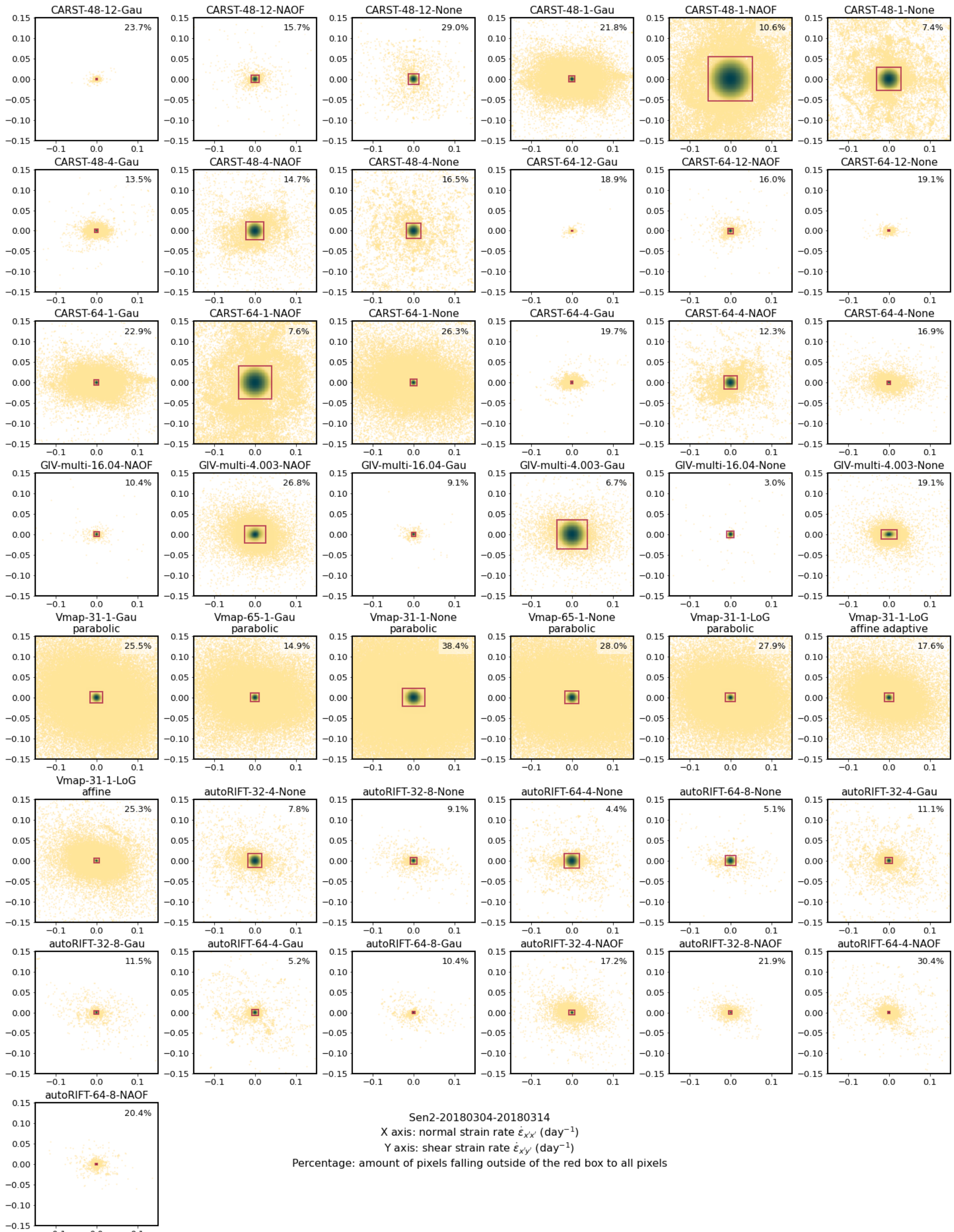


Figure S22. Longitudinal strain rate distribution of the pair `Sen2-20180304-20180314`. (zoomed with kernel density estimation).

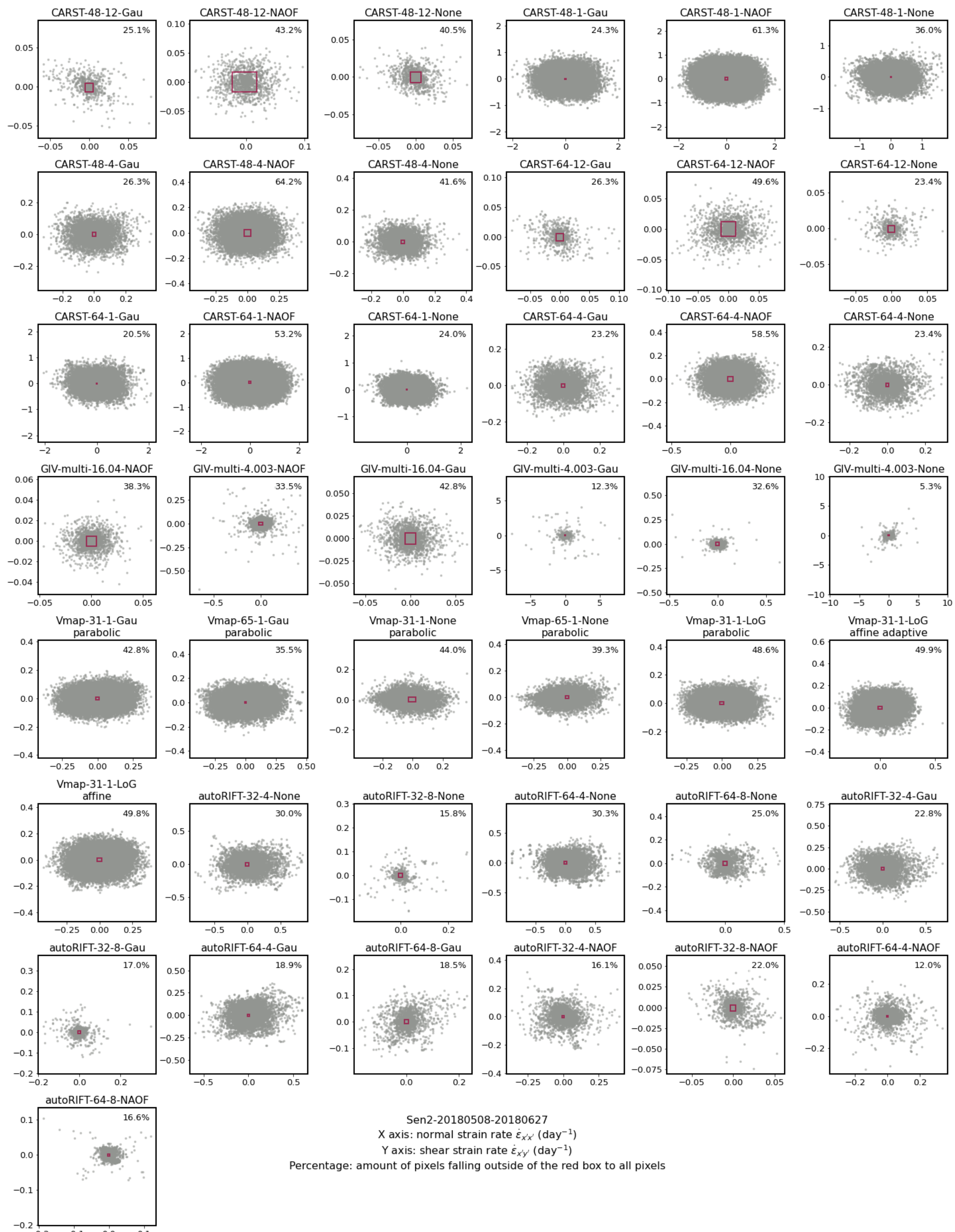


Figure S23. Longitudinal strain rate distribution of the pair `Sen2-20180508-20180627`. (full extent).

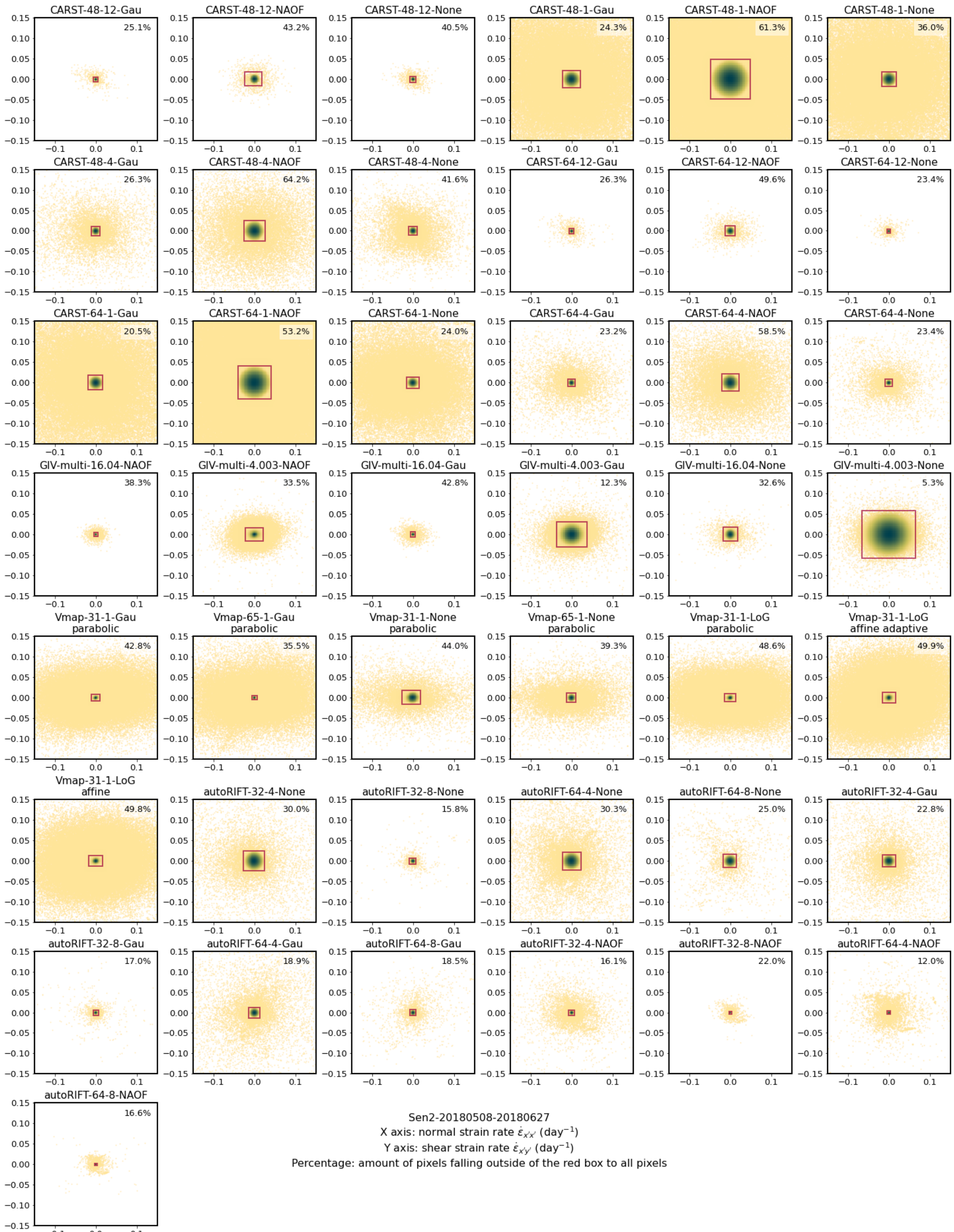


Figure S24. Longitudinal strain rate distribution of the pair `Sen2-20180508-20180627`. (zoomed with kernel density estimation).

10.3.2 3.2. Map of longitudinal strain rate

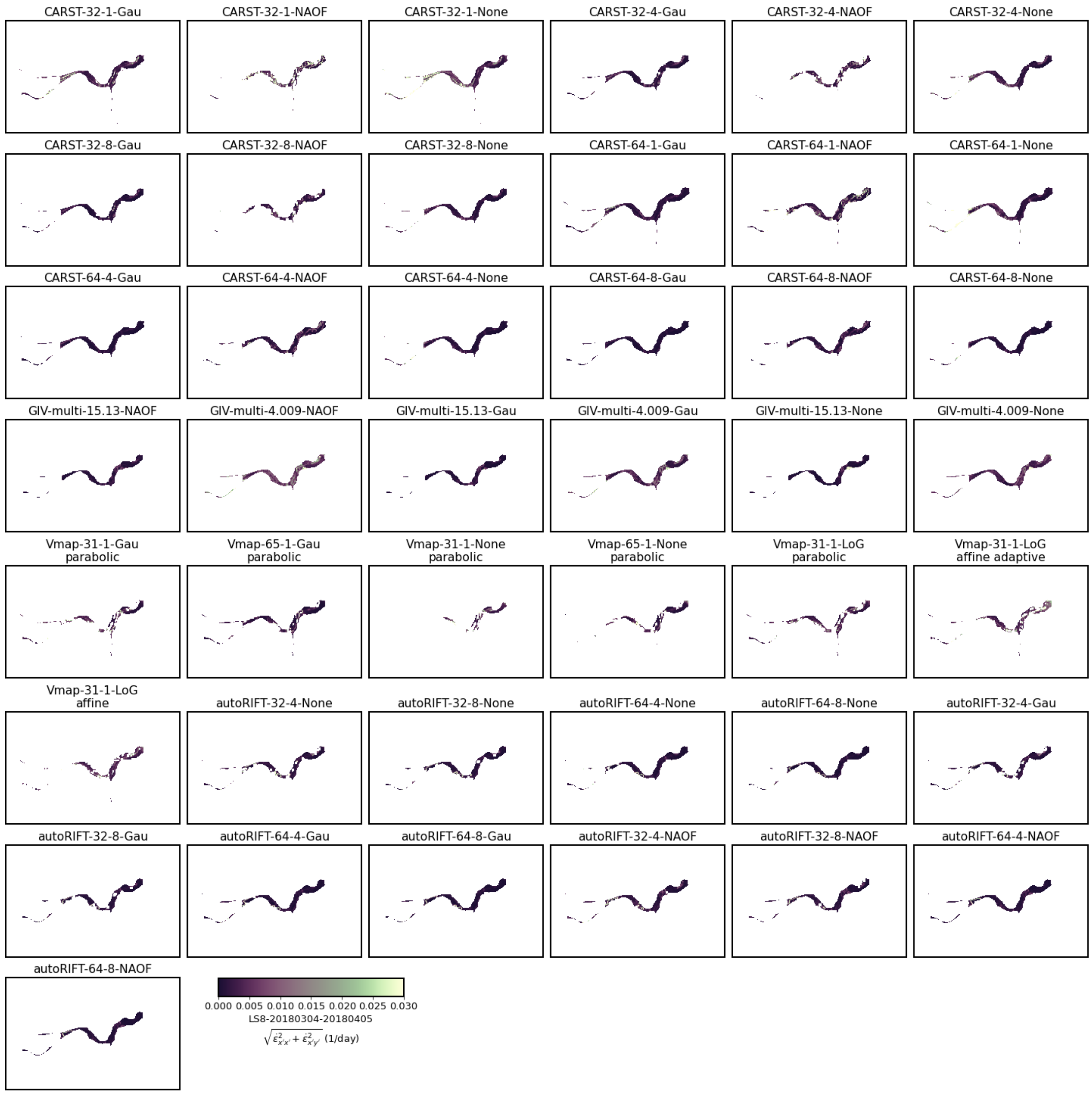


Figure S25. Longitudinal strain rate map of the pair `LS8-20180304-20180405` full extent).

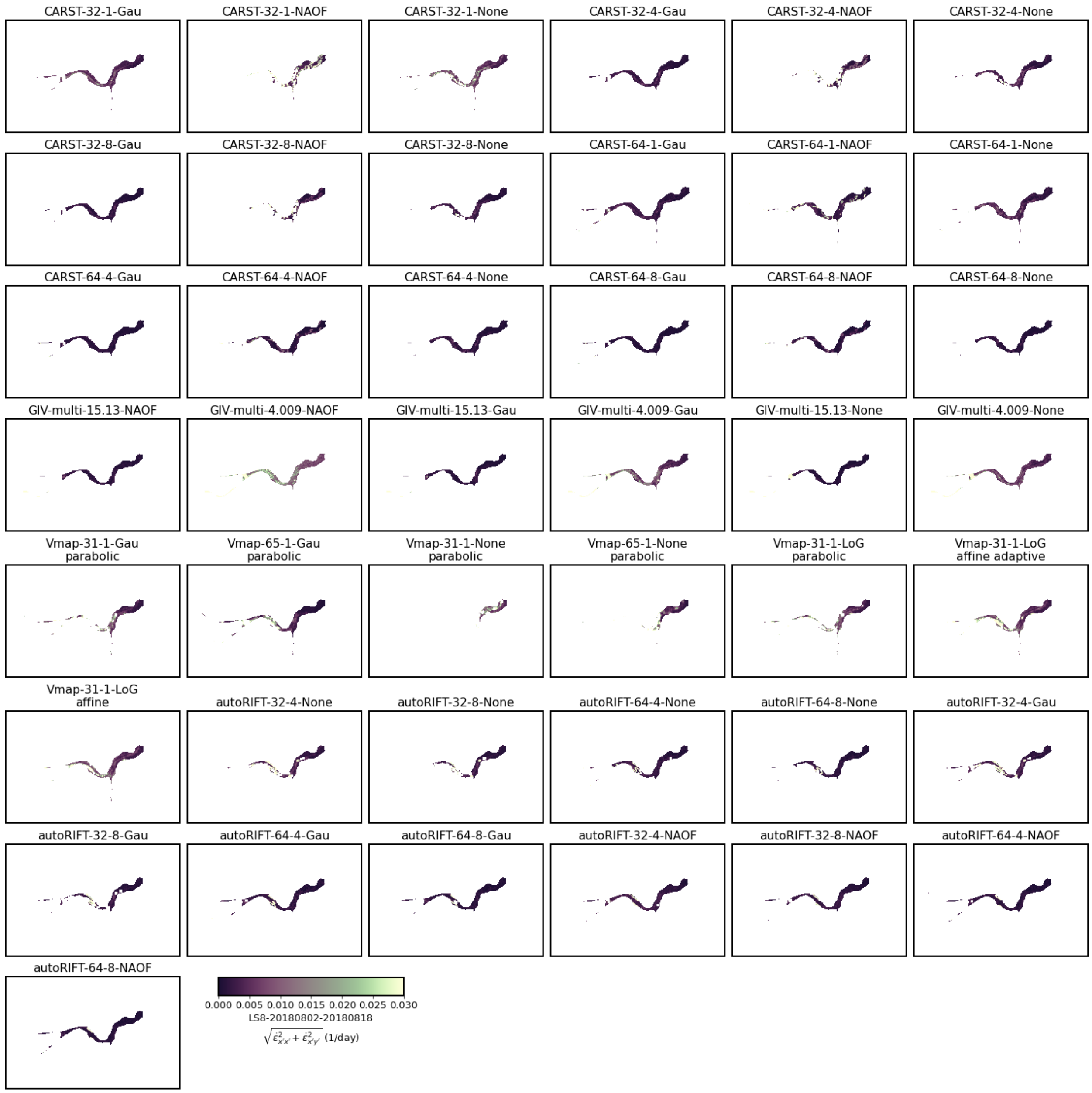


Figure S26. Longitudinal strain rate map of the pair `LS8-20180802-20180818`. full extent).

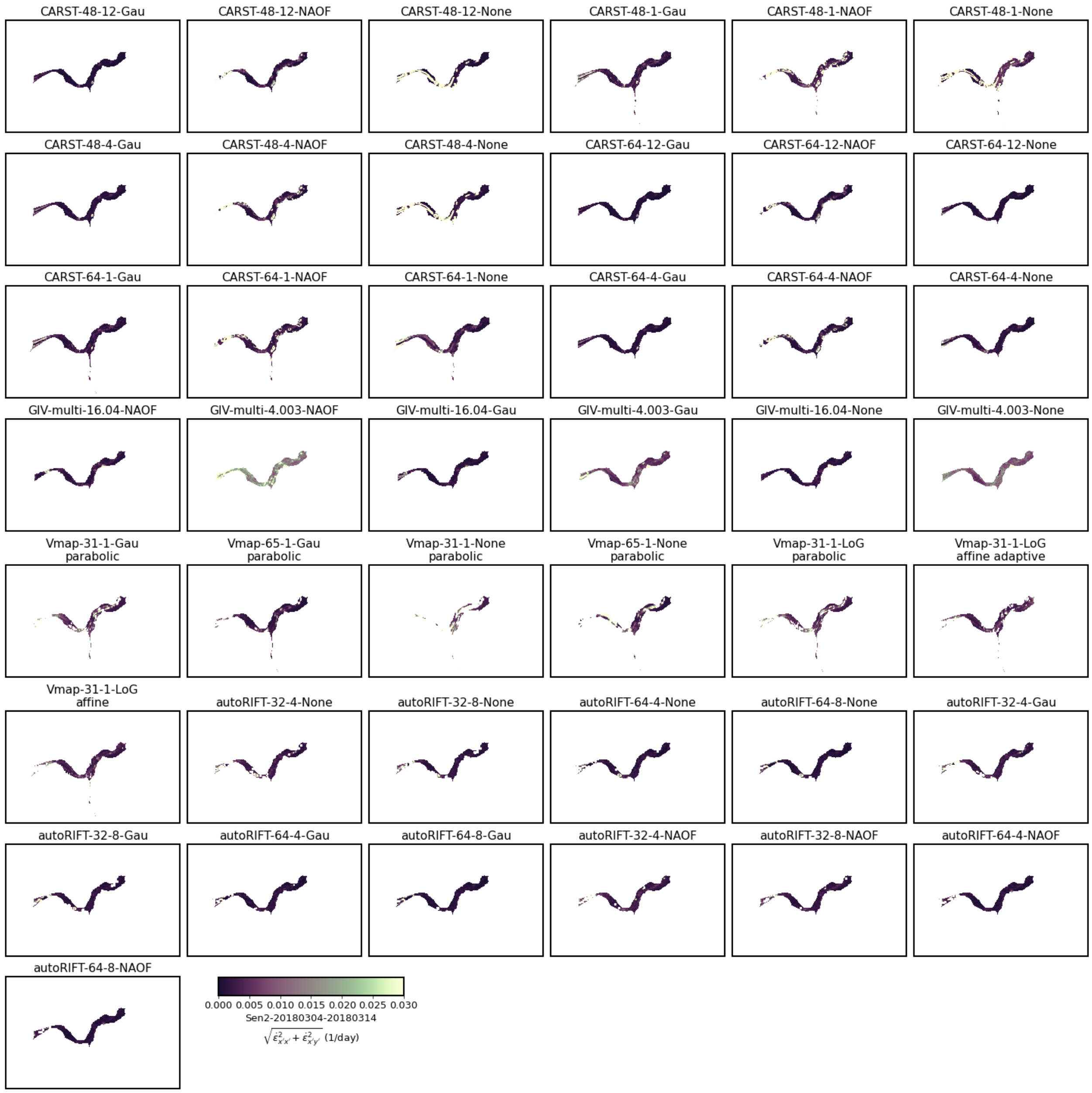


Figure S27. Longitudinal strain rate map of the pair `Sen2-20180304-20180314`. full extent).

Figure S28. Longitudinal strain rate map of the pair `Sen2-20180508-20180627`. full extent).

10.4 Save results

```
for idx, exp in exps.items():
    df.loc[idx, 'LSR-uncertainty-nm'] = exp.metric_alongflow_normal
    df.loc[idx, 'LSR-uncertainty-sh'] = exp.metric_alongflow_shear

df.to_csv('../results_2022.csv', index=False)
```

Part III

Figure scripts

CHAPTER

ELEVEN

FIGURE 2 SCRIPT

To reproduce this figure, make sure you have downloaded all necessary input files (velocity maps and static terrain geometries) from https://doi.org/10.17605/OSF.IO/HE7YR and have updated the `Vx` and `Vy` columns in `notebooks/manifest.csv` with the downloaded file paths before starting the analysis.

```
import glaft
import pandas as pd
import matplotlib as mpl
from matplotlib.ticker import FormatStrFormatter
import matplotlib.pyplot as plt
import numpy as np
```

```
# font and linewidth settings
font = {'size'   : 14}
mpl.rc('font', **font)
axes_settings = {'linewidth'   : 2}
mpl.rc('axes', **axes_settings)

# read and select data
df = pd.read_csv('../manifest.csv', dtype=str)
in_shp = '/home/jovyan/Projects/PX_comparison/shapefiles/bedrock_V2.shp'
selected_cases = df.loc[[126, 130, 134, 97, 73, 28]]
selected_cases
```

```
                        Date Duration (days)   Template size (px)  \
126  LS8-20180304-20180405              32                   64
130  LS8-20180304-20180405              32                   64
134  LS8-20180304-20180405              32                   64
97   LS8-20180304-20180405              32                   65
73   LS8-20180304-20180405              32  varying: multi-pass
28   LS8-20180304-20180405              32                   64

       Template size (m) Pixel spacing (px) Pixel spacing (m) Prefilter  \
126                  960                  4                60      None
130                  960                  4                60       Gau
134                  960                  4                60      NAOF
97                   975                  1                15       Gau
73   varying: multi-pass              4.009             60.14      NAOF
28                   960                  1                15      NAOF

                   Subpixel  Software  \
126                   pyrUP  autoRIFT
130                   pyrUP  autoRIFT
```

(continues on next page)

(continued from previous page)

```
134                  pyrUP   autoRIFT
97                parabolic      Vmap
73     interest point groups       GIV
28      16-node oversampling     CARST

                                                     Vx   \
126   /home/jovyan/Projects/PX_comparison/PX/autoRIF...
130   /home/jovyan/Projects/PX_comparison/PX/autoRIF...
134   /home/jovyan/Projects/PX_comparison/PX/autoRIF...
97    /home/jovyan/Projects/PX_comparison/PX/Vmap/pa...
73    /home/jovyan/Projects/PX_comparison/PX/GIV/u_l...
28    /home/jovyan/Projects/PX_comparison/PX/CARST/2...

                                                     Vy
126   /home/jovyan/Projects/PX_comparison/PX/autoRIF...
130   /home/jovyan/Projects/PX_comparison/PX/autoRIF...
134   /home/jovyan/Projects/PX_comparison/PX/autoRIF...
97    /home/jovyan/Projects/PX_comparison/PX/Vmap/pa...
73    /home/jovyan/Projects/PX_comparison/PX/GIV/v_l...
28    /home/jovyan/Projects/PX_comparison/PX/CARST/2...
```

This cell performs the static terrain analysis and calculates the corresponding metrics.

```
exps = {}

for idx, row in selected_cases.iterrows():
    exp = glaft.Velocity(vxfile=row.Vx, vyfile=row.Vy, static_area=in_shp,
                         kde_gridsize=60, thres_sigma=2.0)
    exp.static_terrain_analysis()
    exps[idx] = exp
```

The following functions plot the braces on the axes with annotations, which is necessary for Figure 2. They are modified from guzey's answer to this StackOverflow thread.

```
def draw_brace_x(ax, xspan: tuple=(None, None), yy: float=0.0, text: str=''):
    """
    ax: axes to be drawn.
    xspan: x coordinates of the two brace ending points.
    yy: y coordinate of the two brace ending points.
        (The brace will be placed horizontally)
    text: annotation text.
    """

    xmin, xmax = xspan
    xspan = xmax - xmin
    ax_xmin, ax_xmax = ax.get_xlim()
    xax_span = ax_xmax - ax_xmin

    ymin, ymax = ax.get_ylim()
    yspan = ymax - ymin
    resolution = int(xspan / xax_span * 100) * 2 + 1 # sampling resolution of the␣
↪sigmoid brace
    beta = 200. / xax_span                          # the higher this is, the␣
↪sharper the sigmoid
```

(continues on next page)

(continued from previous page)

```
    x = np.linspace(xmin, xmax, resolution)
    x_half = x[:int(resolution / 2) + 1]
    y_half_brace = (1 / (1. + np.exp(-beta * (x_half - x_half[0] )))
                    + 1 / (1. + np.exp(-beta * (x_half - x_half[-1]))))
    y = np.concatenate((y_half_brace, y_half_brace[-2::-1]))
    y = yy - (.05 * y - .01) * yspan                  # adjust vertical stretch and␣
↪position

    ax.plot(x, y, color='black', lw=1)
    ax.text((xmax + xmin) / 2., yy - .07 * yspan, text, ha='left', va='top')

def draw_brace_y(ax, yspan: tuple=(None, None), xx: float=0.0, text: str=''):
    """
    ax: axes to be drawn.
    yspan: y coordinates of the two brace ending points.
    xx: x coordinate of the two brace ending points.
        (The brace will be placed vertically)
    text: annotation text.
    """

    ymin, ymax = yspan
    yspan = ymax - ymin
    ax_ymin, ax_ymax = ax.get_ylim()
    yax_span = ax_ymax - ax_ymin

    xmin, xmax = ax.get_xlim()
    xspan = xmax - xmin
    resolution = int(yspan / yax_span * 100) * 2 + 1 # sampling resolution of the␣
↪sigmoid brace
    beta = 200. / yax_span                            # the higher this is, the␣
↪sharper the sigmoid

    y = np.linspace(ymin, ymax, resolution)
    y_half = y[:int(resolution / 2) + 1]
    x_half_brace = (1 / (1. + np.exp(-beta * (y_half - y_half[0] )))
                    + 1 / (1. + np.exp(-beta * (y_half - y_half[-1]))))
    x = np.concatenate((x_half_brace, x_half_brace[-2::-1]))
    x = xx - (.05 * x - .01) * xspan                  # adjust vertical stretch and␣
↪position

    ax.plot(x, y, color='black', lw=1)
    ax.text(xx - .05 * xspan, (ymax + ymin) / 2., text, rotation=90, ha='right', va=
↪'bottom')
```

Now starting to make the figure:

```
fig = plt.figure(figsize=(13, 13))
subfigs = fig.subfigures(2, 4, wspace=0, hspace=0, width_ratios=(0.3, 0.3, 0.3, 0.1))␣
↪    # last column is for colorbar
all_axs = np.empty((2,4), dtype='object')

# create two subplots in each subfigure
for i, row in enumerate(subfigs[:, :3]):
    for j, subfig in enumerate(row):
        all_axs[i, j] = subfig.subplots(2, 1, gridspec_kw = {'hspace':0, 'height_
↪ratios':(0.3962, 0.6038)})
```

(continues on next page)

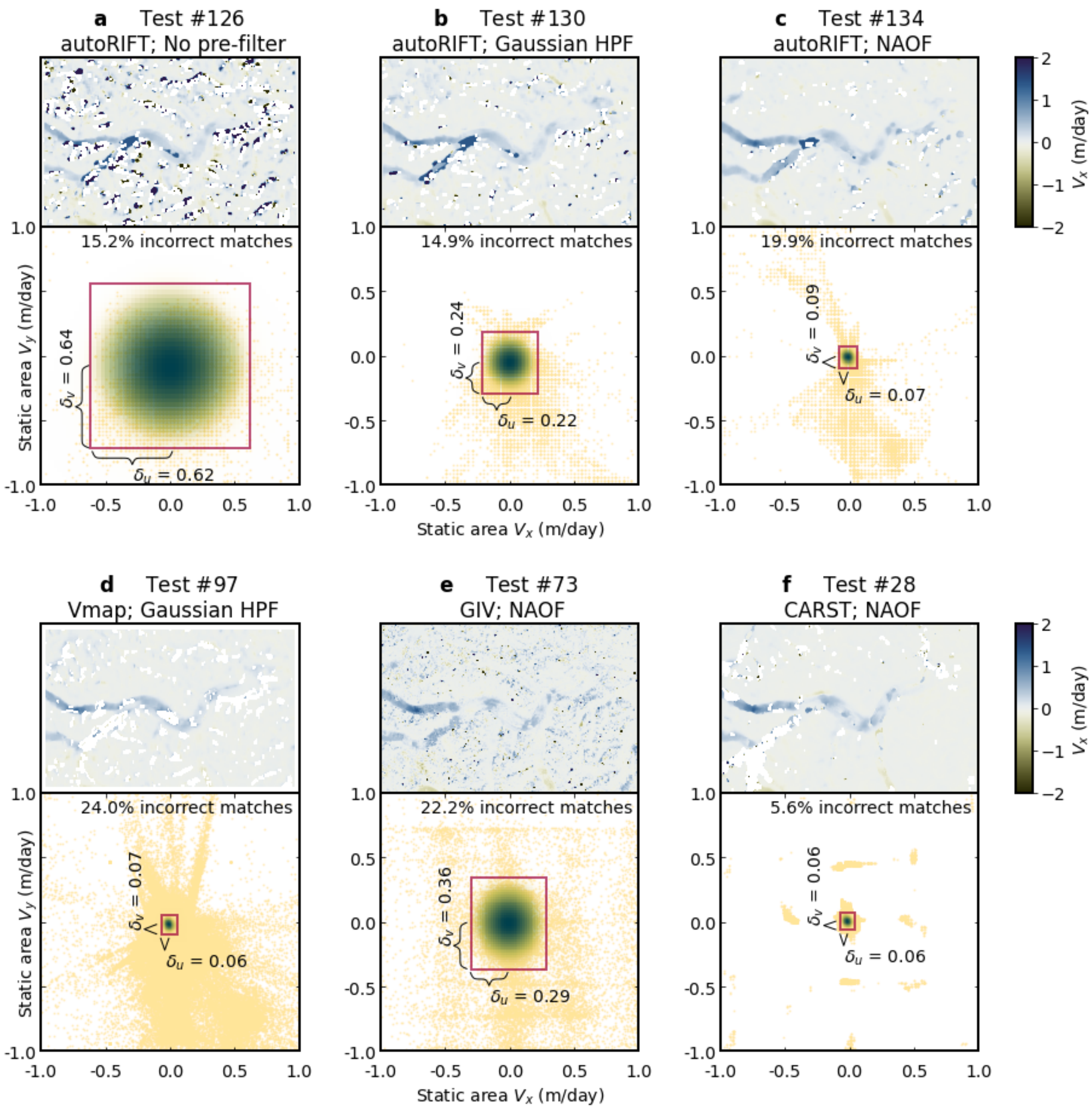
(continued from previous page)

```
title_labels = np.array([['$\mathbf{a}$ \t Test #126 \n autoRIFT; No pre-filter',
                          '$\mathbf{b}$ \t Test #130 \n autoRIFT; Gaussian HPF',
                          '$\mathbf{c}$ \t Test #134 \n autoRIFT; NAOF'],
                         ['$\mathbf{d}$ \t Test #97 \n Vmap; Gaussian HPF',
                          '$\mathbf{e}$ \t Test #73 \n GIV; NAOF',
                          '$\mathbf{f}$ \t Test #28 \n CARST; NAOF']])

for idx, i, j in [[126, 0, 0], [130, 0, 1], [134, 0, 2],
                  [97,  1, 0], [73,  1, 1], [28,  1, 2]]:
    exp = exps[idx]

    # top panel
    ax_sel = all_axs[i, j][0]
    cm_settings = glaft.show_velocomp(exp.vxfile, ax=ax_sel)
    ax_sel.set_aspect('equal', adjustable='datalim')
    ax_sel.set_title(title_labels[i, j])

    # bottom panel
    ax_sel = all_axs[i, j][1]
    exp.plot_zoomed_extent(ax=ax_sel)
    ax_sel.set_aspect('equal', adjustable='box')
    ax_sel.set_xlim(-1, 1)
    ax_sel.set_ylim(-1, 1)
    ax_sel.set_title(None)

    # bottom panel ticks
    ax_sel.tick_params(direction="in", bottom=True, top=True, left=True, right=True)
    ax_sel.tick_params(axis='x', pad=10)
    ax_sel.yaxis.set_major_formatter(FormatStrFormatter('%.1f'))
    ax_sel.xaxis.set_major_formatter(FormatStrFormatter('%.1f'))
    ax_sel.set_yticks([-1.0, -0.5, 0.0, 0.5, 1.0])
    ax_sel.set_xticks([-1.0, -0.5, 0.0, 0.5, 1.0])

    # show percentage of incorrent matches
    ax_sel.text(0.95, 0.95, '{:.1f}% incorrect matches'.format(exp.outlier_percent *␣
↪100), ha='right', va='top')

    # annotations of du and dv
    draw_brace_x(ax_sel,
                 xspan=(exp.kdepeak_x - exp.metric_static_terrain_x, exp.kdepeak_x),
                 yy=exp.kdepeak_y - exp.metric_static_terrain_y,
                 text='$\delta_u$ = {:.2f}'.format(exp.metric_static_terrain_x))
    draw_brace_y(ax_sel,
                 yspan=(exp.kdepeak_y - exp.metric_static_terrain_y, exp.kdepeak_y),
                 xx=exp.kdepeak_x - exp.metric_static_terrain_x,
                 text='$\delta_v$ = {:.2f}'.format(exp.metric_static_terrain_y))


# fine-tune positions of the top panels
bbox_top    = all_axs[0, 0][0].get_position()
bbox_bottom = all_axs[0, 0][1].get_position()
bbox_top.x0 = bbox_bottom.x0
bbox_top.x1 = bbox_bottom.x1
bbox_top.y0 = bbox_bottom.y1
new_bbox_top_y1 = bbox_top.y0 + (bbox_top.y1 - bbox_top.y0) * ((bbox_bottom.x1 - bbox_
↪bottom.x0) / (bbox_top.x1 - bbox_top.x0))
```

(continues on next page)

(continued from previous page)

```
bbox_top.y1 = new_bbox_top_y1

for i, row in enumerate(all_axs[:, :3]):
    for j, subfig in enumerate(row):
        all_axs[i, j][0].set_position(bbox_top)

# add colorbars
mappable = glaft.prep_colorbar_mappable(**cm_settings)
cax = subfigs[0, 3].add_axes([0.0, bbox_top.y0, 0.15, bbox_top.y1 - bbox_top.y0])
subfigs[0, 3].colorbar(mappable, cax=cax, orientation='vertical', label='$V_x$ ({})'.
↪format(exp.velocity_unit), ticks=[-2, -1, 0, 1, 2])
cax = subfigs[1, 3].add_axes([0.0, bbox_top.y0, 0.15, bbox_top.y1 - bbox_top.y0])
subfigs[1, 3].colorbar(mappable, cax=cax, orientation='vertical', label='$V_x$ ({})'.
↪format(exp.velocity_unit), ticks=[-2, -1, 0, 1, 2])

# add axis labels
all_axs[0, 0][1].set_yticklabels(['-1.0', '', '', '', '1.0'])                          ␣
↪               # to prevent label bleeding
all_axs[1, 0][1].set_yticklabels(['-1.0', '', '', '', '1.0'])                          ␣
↪               # ditto
all_axs[0, 0][1].set_ylabel('Static area $V_y$ ({})'.format(exp.velocity_unit),␣
↪labelpad=-30.0)     # ditto
all_axs[1, 0][1].set_ylabel('Static area $V_y$ ({})'.format(exp.velocity_unit),␣
↪labelpad=-30.0)     # ditto

all_axs[0, 1][1].set_xlabel('Static area $V_x$ ({})'.format(exp.velocity_unit))
all_axs[1, 1][1].set_xlabel('Static area $V_x$ ({})'.format(exp.velocity_unit))

# save figure
fig.patch.set_facecolor('xkcd:white')
fig.savefig('Fig2.png', dpi=200)
```


a Test #126
autoRIFT; No pre-filter
15.2% incorrect matches
δ_v = 0.64
δ_u = 0.62
b Test #130
autoRIFT; Gaussian HPF
14.9% incorrect matches
δ_v = 0.24
δ_u = 0.22
c Test #134
autoRIFT; NAOF
19.9% incorrect matches
δ_v = 0.09
δ_u = 0.07
d Test #97
Vmap; Gaussian HPF
24.0% incorrect matches
δ_v = 0.07
δ_u = 0.06
e Test #73
GIV; NAOF
22.2% incorrect matches
δ_v = 0.36
δ_u = 0.29
f Test #28
CARST; NAOF
5.6% incorrect matches
δ_v = 0.06
δ_u = 0.06
Static area V_y (m/day)
Static area V_x (m/day)
V_x (m/day)

CHAPTER TWELVE

FIGURE 3 SCRIPT

To reproduce this figure, make sure you have downloaded all necessary input files (velocity maps and static terrain geometries) from https://doi.org/10.17605/OSF.IO/HE7YR and have updated the `Vx` and `Vy` columns in `notebooks/manifest.csv` with the downloaded file paths before starting the analysis.

```
import glaft
import pandas as pd
import matplotlib as mpl
from matplotlib.ticker import FormatStrFormatter
import matplotlib.pyplot as plt
import numpy as np
from cmcrameri import cm as camericm
```

```
# font and linewidth settings
font = {'size'   : 14}
mpl.rc('font', **font)
axes_settings = {'linewidth'   : 2}
mpl.rc('axes', **axes_settings)

# read and select data
df = pd.read_csv('../manifest.csv', dtype=str)
in_shp = '/home/jovyan/Projects/PX_comparison/shapefiles/glacier_V1_Kaskawulsh_s_
↪inwardBuffer600m.shp'
selected_cases = df.loc[[64, 79, 117]]
selected_cases
```

```
                      Date Duration (days)   Template size (px)  \
64   LS8-20180802-20180818              16                   64
79   LS8-20180802-20180818              16  varying: multi-pass
117  LS8-20180802-20180818              16                   31

       Template size (m) Pixel spacing (px) Pixel spacing (m) Prefilter  \
64                   960                  1                15      NAOF
79   varying: multi-pass              4.009             60.14      NAOF
117                  465                  1                15       LoG

                   Subpixel Software  \
64    16-node oversampling    CARST
79   interest point groups      GIV
117                 affine     Vmap

                                                  Vx  \
64   /home/jovyan/Projects/PX_comparison/PX/CARST/2...
```

(continues on next page)

(continued from previous page)

```
79   /home/jovyan/Projects/PX_comparison/PX/GIV/u_l...
117  /home/jovyan/Projects/PX_comparison/PX/Vmap/su...

                                                    Vy
64   /home/jovyan/Projects/PX_comparison/PX/CARST/2...
79   /home/jovyan/Projects/PX_comparison/PX/GIV/v_l...
117  /home/jovyan/Projects/PX_comparison/PX/Vmap/su...
```

This cell performs the static terrain analysis and calculates the corresponding metrics.

```
exps = {}

for idx, row in selected_cases.iterrows():
    exp = glaft.Velocity(vxfile=row.Vx, vyfile=row.Vy, on_ice_area=in_shp, kde_
↪gridsize=60, thres_sigma=2.0)
    exp.longitudinal_shear_analysis()
    exps[idx] = exp
```

The following functions plot the braces on the axes with annotations, which is necessary for Figure 3. They are modified from guzey's answer to this StackOverflow thread.

```
def draw_brace_x(ax, xspan: tuple=(None, None), yy: float=0.0, text: str=''):
    """
    ax: axes to be drawn.
    xspan: x coordinates of the two brace ending points.
    yy: y coordinate of the two brace ending points.
        (The brace will be placed horizontally)
    text: annotation text.
    """

    xmin, xmax = xspan
    xspan = xmax - xmin
    ax_xmin, ax_xmax = ax.get_xlim()
    xax_span = ax_xmax - ax_xmin

    ymin, ymax = ax.get_ylim()
    yspan = ymax - ymin
    resolution = int(xspan / xax_span * 100) * 2 + 1 # sampling resolution of the␣
↪sigmoid brace
    beta = 200. / xax_span                           # the higher this is, the␣
↪sharper the sigmoid

    x = np.linspace(xmin, xmax, resolution)
    x_half = x[:int(resolution / 2) + 1]
    y_half_brace = (1 / (1. + np.exp(-beta * (x_half - x_half[0] )))
                  + 1 / (1. + np.exp(-beta * (x_half - x_half[-1]))))
    y = np.concatenate((y_half_brace, y_half_brace[-2::-1]))
    y = yy - (.05 * y - .01) * yspan                 # adjust vertical stretch and␣
↪position

    ax.plot(x, y, color='black', lw=1)
    ax.text((xmax + xmin) / 2., yy - .07 * yspan, text, ha='left', va='top')

def draw_brace_y(ax, yspan: tuple=(None, None), xx: float=0.0, text: str=''):
    """
```

(continues on next page)

(continued from previous page)

```
    ax: axes to be drawn.
    yspan: y coordinates of the two brace ending points.
    xx: x coordinate of the two brace ending points.
        (The brace will be placed vertically)
    text: annotation text.
    """

    ymin, ymax = yspan
    yspan = ymax - ymin
    ax_ymin, ax_ymax = ax.get_ylim()
    yax_span = ax_ymax - ax_ymin

    xmin, xmax = ax.get_xlim()
    xspan = xmax - xmin
    resolution = int(yspan / yax_span * 100) * 2 + 1 # sampling resolution of the␣
↪sigmoid brace
    beta = 200. / yax_span                              # the higher this is, the␣
↪sharper the sigmoid

    y = np.linspace(ymin, ymax, resolution)
    y_half = y[:int(resolution / 2) + 1]
    x_half_brace = (1 / (1. + np.exp(-beta * (y_half - y_half[0] )))
                    + 1 / (1. + np.exp(-beta * (y_half - y_half[-1]))))
    x = np.concatenate((x_half_brace, x_half_brace[-2::-1]))
    x = xx - (.05 * x - .01) * xspan                    # adjust vertical stretch and␣
↪position

    ax.plot(x, y, color='black', lw=1)
    ax.text(xx - .05 * xspan, (ymax + ymin) / 2., text, rotation=90, ha='right', va=
↪'bottom')
```

Now starting to make the figure:

```
# vmax = exps[117].metric_alongflow_normal
vmax = 0.03

fig = plt.figure(figsize=(13, 9.5))
subfigs = fig.subfigures(1, 4, wspace=0, hspace=0, width_ratios=(0.3, 0.3, 0.3, 0.1))␣
↪    # last column is for colorbar
all_axs = np.empty(4, dtype='object')

# create three subplots in each subfigure
for i, subfig in enumerate(subfigs[:3]):
    all_axs[i] = subfig.subplots(3, 1, gridspec_kw = {'hspace':0, 'height_ratios':(0.
↪28377, 0.28377, 0.43246)})
# a = 0.3962, b = 0.6038 (from Figure 2)
# x = a/(2a + b) = 0.28377
# y = b/(2a + b) = 0.43246

title_labels = np.array(['$\mathbf{a}$ \t Test #64 \n CARST; NAOF',
                         '$\mathbf{b}$ \t Test #79 \n GIV; NAOF',
                         '$\mathbf{c}$ \t Test #117 \n Vmap; LoG'])

for idx, i in [[64, 0], [79, 1], [117, 2]]:

    exp = exps[idx]
```

(continues on next page)

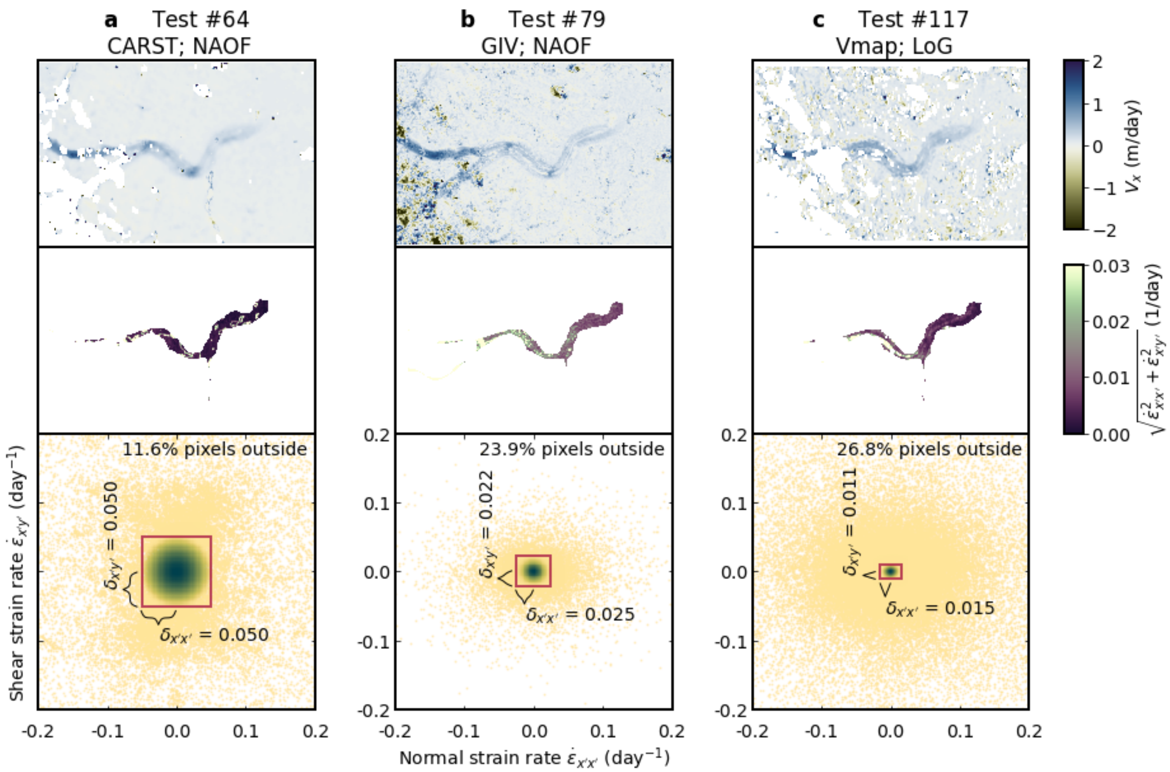
(continued from previous page)

```
    # top panel
    ax_sel = all_axs[i][0]
    cm_settings = glaft.show_velocomp(exp.vxfile, ax=ax_sel)
    ax_sel.set_aspect('equal', adjustable='datalim')
    ax_sel.set_title(title_labels[i])

    # middle panel
    ax_sel = all_axs[i][1]
    mappable_strain = exp.plot_strain_map(ax=ax_sel, vmax=vmax, base_
↪colormap=crameri cm.tokyo)
    ax_sel.set_aspect('equal', adjustable='datalim')

    # bottom panel
    ax_sel = all_axs[i][2]
    exp.plot_zoomed_extent(metric=2, ax=ax_sel)
    ax_sel.set_aspect('equal', adjustable='box')
    ax_sel.set_xlim(-0.2, 0.2)
    ax_sel.set_ylim(-0.2, 0.2)
    ax_sel.set_title(None)

    # bottom panel ticks
    ax_sel.tick_params(direction="in", bottom=True, top=True, left=True, right=True)
    ax_sel.tick_params(axis='x', pad=10)
    ax_sel.yaxis.set_major_formatter(FormatStrFormatter('%.1f'))
    ax_sel.xaxis.set_major_formatter(FormatStrFormatter('%.1f'))
    ax_sel.set_yticks([-0.2, -0.1, 0.0, 0.1, 0.2])
    ax_sel.set_xticks([-0.2, -0.1, 0.0, 0.1, 0.2])

    # show percentage of pixels outside
    ax_sel.text(0.19, 0.19, '{:.1f}% pixels outside'.format(exp.outlier_percent *␣
↪100), ha='right', va='top')

    # annotations of du and dv
    draw_brace_x(ax_sel,
                 xspan=(exp.kdepeak_x - exp.metric_alongflow_normal, exp.kdepeak_x),
                 yy=exp.kdepeak_y - exp.metric_alongflow_shear,
                 text="$\delta_{x'x'}$" + " = {:.3f}".format(exp.metric_alongflow_
↪normal))
    draw_brace_y(ax_sel,
                 yspan=(exp.kdepeak_y - exp.metric_alongflow_shear, exp.kdepeak_y),
                 xx=exp.kdepeak_x - exp.metric_alongflow_normal,
                 text="$\delta_{x'y'}$" + " = {:.3f}".format(exp.metric_alongflow_
↪shear))



# fine-tune positions of the top panels
# x0 = left, x1 = right, y0 = bottom, y1 = top
bbox_top    = all_axs[0][0].get_position()
bbox_middle = all_axs[0][1].get_position()
bbox_bottom = all_axs[0][2].get_position()

bbox_bottom_vertical_shift = bbox_middle.y0 - bbox_bottom.y1
bbox_bottom.y0 = bbox_bottom.y0 + bbox_bottom_vertical_shift
bbox_bottom.y1 = bbox_bottom.y1 + bbox_bottom_vertical_shift
```

(continues on next page)

(continued from previous page)

```
for i, subfig in enumerate(all_axs[:3]):
    all_axs[i][2].set_position(bbox_bottom)

# add colorbars
mappable_velo = glaft.prep_colorbar_mappable(**cm_settings)
cax = subfigs[3].add_axes([0.0, bbox_top.y0 + 0.02, 0.15, bbox_top.y1 - bbox_top.y0 -␣
↪0.02])
subfigs[3].colorbar(mappable_velo, cax=cax, orientation='vertical', label='$V_x$ ({})
↪'.format(exp.velocity_unit), ticks=[-2, -1, 0, 1, 2])

cax2 = subfigs[3].add_axes([0.0, bbox_middle.y0, 0.15, bbox_middle.y1 - bbox_middle.
↪y0 - 0.02])
strain_cmap_label = "$\sqrt{\dot{\epsilon}_{x'x'}^2 + \dot{\epsilon}_{x'y'}^2}$ (1/
↪day)"
subfigs[3].colorbar(mappable_strain, cax=cax2, orientation='vertical', label=strain_
↪cmap_label, ticks=[0, 0.01, 0.02, 0.03])

# add axis labels
all_axs[0][2].set_yticklabels(['', '', '', '', ''])                                  ␣
↪   # to prevent label bleeding
all_axs[0][2].set_ylabel("Shear strain rate $\dot{\epsilon}_{x'y'}$ (day$^{-1}$)",␣
↪labelpad=-0.0)     # ditto

all_axs[1][2].set_xlabel("Normal strain rate $\dot{\epsilon}_{x'x'}$ (day$^{-1}$)")

# save figure
fig.patch.set_facecolor('xkcd:white')
fig.savefig('Fig3.png', dpi=200)
```


a Test #64
CARST; NAOF
b Test #79
GIV; NAOF
c Test #117
Vmap; LoG
V_x (m/day)
$\sqrt{\dot\epsilon_{x'x'}^2 + \dot\epsilon_{x'y'}^2}$ (1/day)
11.6% pixels outside
23.9% pixels outside
26.8% pixels outside
$\delta_{x'y'}$ = 0.050
$\delta_{x'x'}$ = 0.050
$\delta_{x'y'}$ = 0.022
$\delta_{x'x'}$ = 0.025
$\delta_{x'y'}$ = 0.011
$\delta_{x'x'}$ = 0.015
Shear strain rate $\dot\epsilon_{x'y'}$ (day^{-1})
Normal strain rate $\dot\epsilon_{x'x'}$ (day^{-1})

CHAPTER THIRTEEN

FIGURE 4 SCRIPT

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib as mpl
```

```
# font and linewidth settings
font = {'size'   : 20}
mpl.rc('font', **font)
mpl.rc('legend', fontsize=16)
axes_settings = {'linewidth'   : 2}
mpl.rc('axes', **axes_settings)
```

Now read the data:

```
df = pd.read_csv('../results_2022.csv', dtype=str)
df = df.replace('varying: multi-pass', 0)   # replace some values in 'Template size␣
↪(m)' with another flag
for field in ['Pixel spacing (m)',
              'Template size (m)',
              'SAV-uncertainty-x',
              'SAV-uncertainty-y',
              'SAV-peak-x',
              'SAV-peak-y',
              'LSR-uncertainty-nm',
              'LSR-uncertainty-sh',
              'pt0_vxdiff',
              'pt0_vydiff',
              'pt1_vxdiff',
              'pt1_vydiff',
              'pt2_vxdiff',
              'pt2_vydiff',
              'pt0_vxavgdiff',
              'pt0_vyavgdiff',
              'pt1_vxavgdiff',
              'pt1_vyavgdiff',
              'pt2_vxavgdiff',
              'pt2_vyavgdiff',
              'SAV-outlier-percent',
              'Invalid-pixel-percent']:
    df[field] = df[field].astype(float)
```

(continues on next page)

(continued from previous page)

```
datestrs = ['LS8-20180304-20180405',
            'LS8-20180802-20180818',
            'Sen2-20180304-20180314',
            'Sen2-20180508-20180627']

# df
```

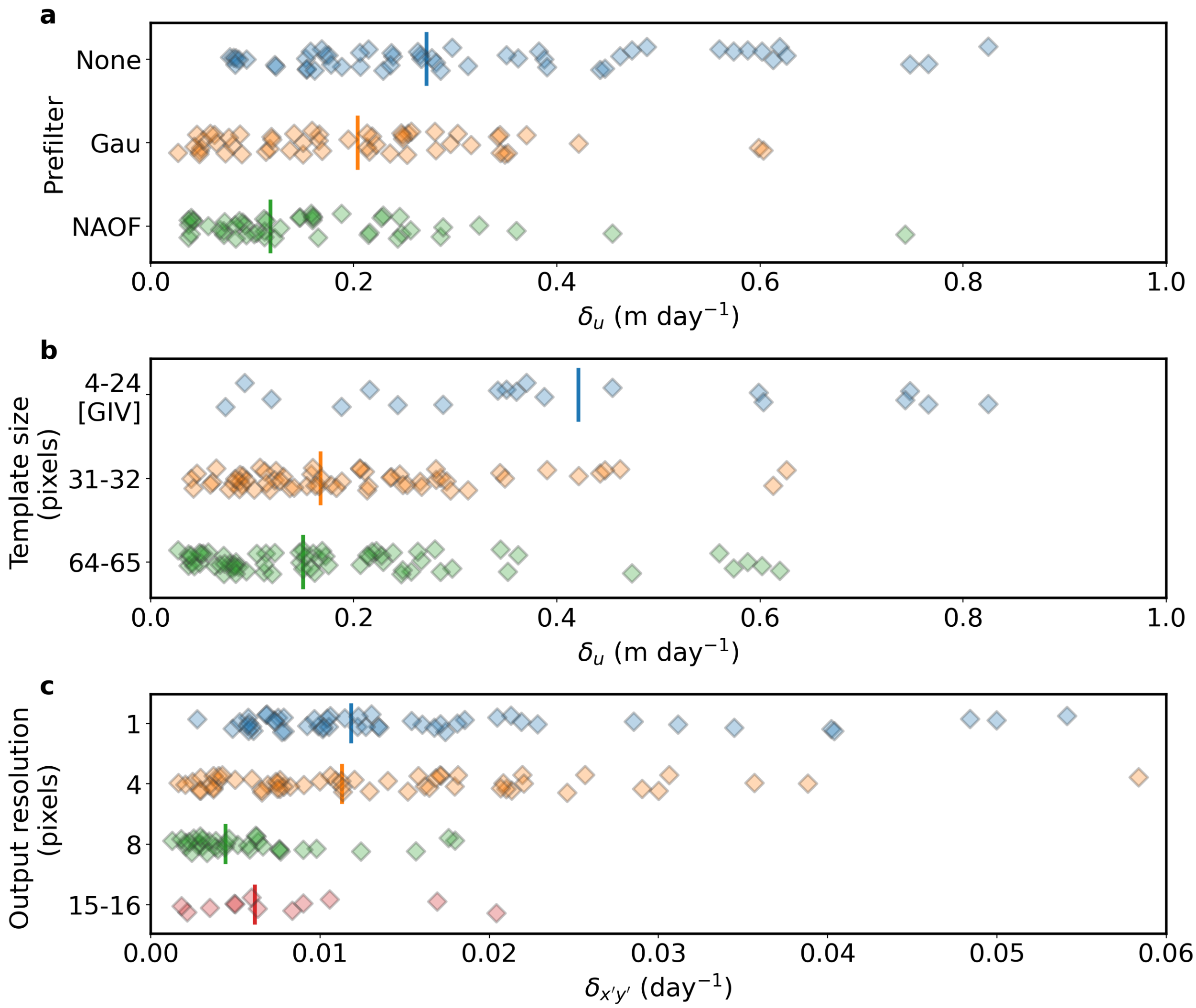
Now plot the figure:

```
demo1 = df[df['Prefilter'] != 'LoG']
demo2 = df[df['Template size (px)'] != "48"]
demo3 = df[df['Pixel spacing (px)'] != "12"]

for idx, row in demo2.iterrows():
    if row['Template size (px)'] in ["64", "65"]:
        demo2.loc[idx, 'Template size (pixels)'] = "64-65"
    elif row['Template size (px)'] in ["31", "32"]:
        demo2.loc[idx, 'Template size (pixels)'] = "31-32"
    elif row['Template size (px)'] == 0:
        demo2.loc[idx, 'Template size (pixels)'] = "multi"
    else:
        demo2.loc[idx, 'Template size (pixels)'] = "else"

for idx, row in demo3.iterrows():
    if row['Pixel spacing (px)'] == "1":
        demo3.loc[idx, 'Pixel spacing (pixels)'] = "1"
    elif row['Pixel spacing (px)'] in ["4", "4.009", "4.003"]:
        demo3.loc[idx, 'Pixel spacing (pixels)'] = "~4"
    elif row['Pixel spacing (px)'] == "8":
        demo3.loc[idx, 'Pixel spacing (pixels)'] = "8"
    elif row['Pixel spacing (px)'] in ["15.13", "16.04"]:
        demo3.loc[idx, 'Pixel spacing (pixels)'] = "~16"
    else:
        demo3.loc[idx, 'Pixel spacing (pixels)'] = "else"

fig, axs = plt.subplots(3, 1, figsize=(13, 11), constrained_layout=True)

kawgs = {'jitter': 0.15, 'marker': 'D', 's': 10, 'alpha': 0.3, 'linewidth': 2, }

sns.stripplot(data=demo1, x="SAV-uncertainty-x", y="Prefilter", ax=axs[0],
              order=['None', 'Gau', 'NAOF'], **kawgs)
sns.stripplot(data=demo2, x="SAV-uncertainty-x", y="Template size (pixels)",␣
↪ax=axs[1],
              order=['multi', '31-32', '64-65'], **kawgs)
sns.stripplot(data=demo3, x="LSR-uncertainty-sh", y="Pixel spacing (pixels)",␣
↪ax=axs[2], **kawgs)

kawgs2 = {'linewidth': 3, }

tmp = demo1[demo1['Prefilter'] == 'None']
tmp_mean = tmp['SAV-uncertainty-x'].median()
axs[0].plot([tmp_mean, tmp_mean], [-0.3, 0.3], **kawgs2)

tmp = demo1[demo1['Prefilter'] == 'Gau']
tmp_mean = tmp['SAV-uncertainty-x'].median()
axs[0].plot([tmp_mean, tmp_mean], [0.7, 1.3], **kawgs2)
```

(continues on next page)

(continued from previous page)

```
tmp = demo1[demo1['Prefilter'] == 'NAOF']
tmp_mean = tmp['SAV-uncertainty-x'].median()
axs[0].plot([tmp_mean, tmp_mean], [1.7, 2.3], **kawgs2)
axs[0].set_xlabel('$\delta_u$ (m day$^{-1}$)')

tmp = demo2[demo2['Template size (pixels)'] == 'multi']
tmp_mean = tmp['SAV-uncertainty-x'].median()
axs[1].plot([tmp_mean, tmp_mean], [-0.3, 0.3], **kawgs2)

tmp = demo2[demo2['Template size (pixels)'] == '31-32']
tmp_mean = tmp['SAV-uncertainty-x'].median()
axs[1].plot([tmp_mean, tmp_mean], [0.7, 1.3], **kawgs2)

tmp = demo2[demo2['Template size (pixels)'] == '64-65']
tmp_mean = tmp['SAV-uncertainty-x'].median()
axs[1].plot([tmp_mean, tmp_mean], [1.7, 2.3], **kawgs2)
axs[1].set_xlabel('$\delta_u$ (m day$^{-1}$)')

tmp = demo3[demo3['Pixel spacing (pixels)'] == '1']
tmp_mean = tmp['LSR-uncertainty-sh'].median()
axs[2].plot([tmp_mean, tmp_mean], [-0.3, 0.3], **kawgs2)

tmp = demo3[demo3['Pixel spacing (pixels)'] == '~4']
tmp_mean = tmp['LSR-uncertainty-sh'].median()
axs[2].plot([tmp_mean, tmp_mean], [0.7, 1.3], **kawgs2)

tmp = demo3[demo3['Pixel spacing (pixels)'] == '8']
tmp_mean = tmp['LSR-uncertainty-sh'].median()
axs[2].plot([tmp_mean, tmp_mean], [1.7, 2.3], **kawgs2)

tmp = demo3[demo3['Pixel spacing (pixels)'] == '~16']
tmp_mean = tmp['LSR-uncertainty-sh'].median()
axs[2].plot([tmp_mean, tmp_mean], [2.7, 3.3], **kawgs2)
axs[2].set_xlabel("$\delta_{x'y'}$ (day$^{-1}$)")

axs[1].set_yticklabels(['4-24\n[GIV]', '31-32', '64-65'])
axs[2].set_yticklabels(['1', '4', '8', '15-16'])

axs[0].set_xlim(0, 1)
axs[1].set_xlim(0, 1)
axs[2].set_xlim(0, 0.06)

axs[0].text(-0.11, 1, "$\mathbf{a}$", transform=axs[0].transAxes)
axs[1].text(-0.11, 1, "$\mathbf{b}$", transform=axs[1].transAxes)
axs[2].text(-0.11, 1, "$\mathbf{c}$", transform=axs[2].transAxes)
axs[1].set_ylabel("Template size \n (pixels)")
axs[2].set_ylabel("Output resolution \n (pixels)")

# save figure
fig.patch.set_facecolor('xkcd:white')
fig.savefig('Fig4.png', dpi=200)
```


a
None
Gau
NAOF
Prefilter
0.0
0.2
0.4
0.6
0.8
1.0
δ_u (m day^{-1})
b
4-24 [GIV]
31-32
64-65
Template size (pixels)
δ_u (m day^{-1})
c
1
4
8
15-16
Output resolution (pixels)
0.00
0.01
0.02
0.03
0.04
0.05
0.06
$\delta_{x'y'}$ (day^{-1})

CHAPTER FOURTEEN

FIGURE 5 SCRIPT

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib as mpl
```

```
# font and linewidth settings
font = {'size'   : 20}
mpl.rc('font', **font)
mpl.rc('legend', fontsize=16)
axes_settings = {'linewidth'   : 2}
mpl.rc('axes', **axes_settings)
```

Now read the data:

```
df = pd.read_csv('../results_2022.csv', dtype=str)
df = df.replace('varying: multi-pass', 0)   # replace some values in 'Template size␣
↪(m)' with another flag
for field in ['Pixel spacing (m)',
              'Template size (m)',
              'SAV-uncertainty-x',
              'SAV-uncertainty-y',
              'SAV-peak-x',
              'SAV-peak-y',
              'LSR-uncertainty-nm',
              'LSR-uncertainty-sh',
              'pt0_vxdiff',
              'pt0_vydiff',
              'pt1_vxdiff',
              'pt1_vydiff',
              'pt2_vxdiff',
              'pt2_vydiff',
              'pt0_vxavgdiff',
              'pt0_vyavgdiff',
              'pt1_vxavgdiff',
              'pt1_vyavgdiff',
              'pt2_vxavgdiff',
              'pt2_vyavgdiff',
              'SAV-outlier-percent',
              'Invalid-pixel-percent']:
    df[field] = df[field].astype(float)
```

(continues on next page)

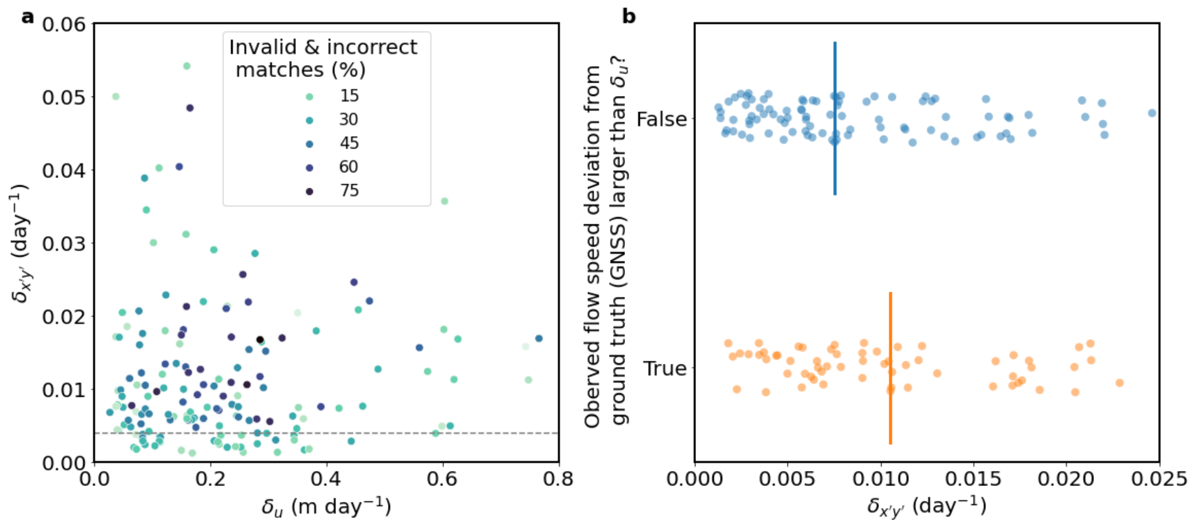
(continued from previous page)

```
datestrs = ['LS8-20180304-20180405',
            'LS8-20180802-20180818',
            'Sen2-20180304-20180314',
            'Sen2-20180508-20180627']

# df
```

Create additional column fields for grouping data:

```
df['large_vxdiff'] = np.abs(df['pt0_vxdiff']) > df['SAV-uncertainty-x']
df['large_vydiff'] = np.abs(df['pt0_vydiff']) > df['SAV-uncertainty-y']
df['large_vxavgdiff'] = np.abs(df['pt0_vxavgdiff']) > df['SAV-uncertainty-x']
df['large_vyavgdiff'] = np.abs(df['pt0_vyavgdiff']) > df['SAV-uncertainty-y']
df['Invalid+Incorrect'] = df['Invalid-pixel-percent'] + df['SAV-outlier-percent'] *␣
↪(1 - df['Invalid-pixel-percent'] / 100)

df['large_vxdiff'] = df['large_vxdiff'].astype(str)
df['large_vxavgdiff'] = df['large_vxavgdiff'].astype(str)
```

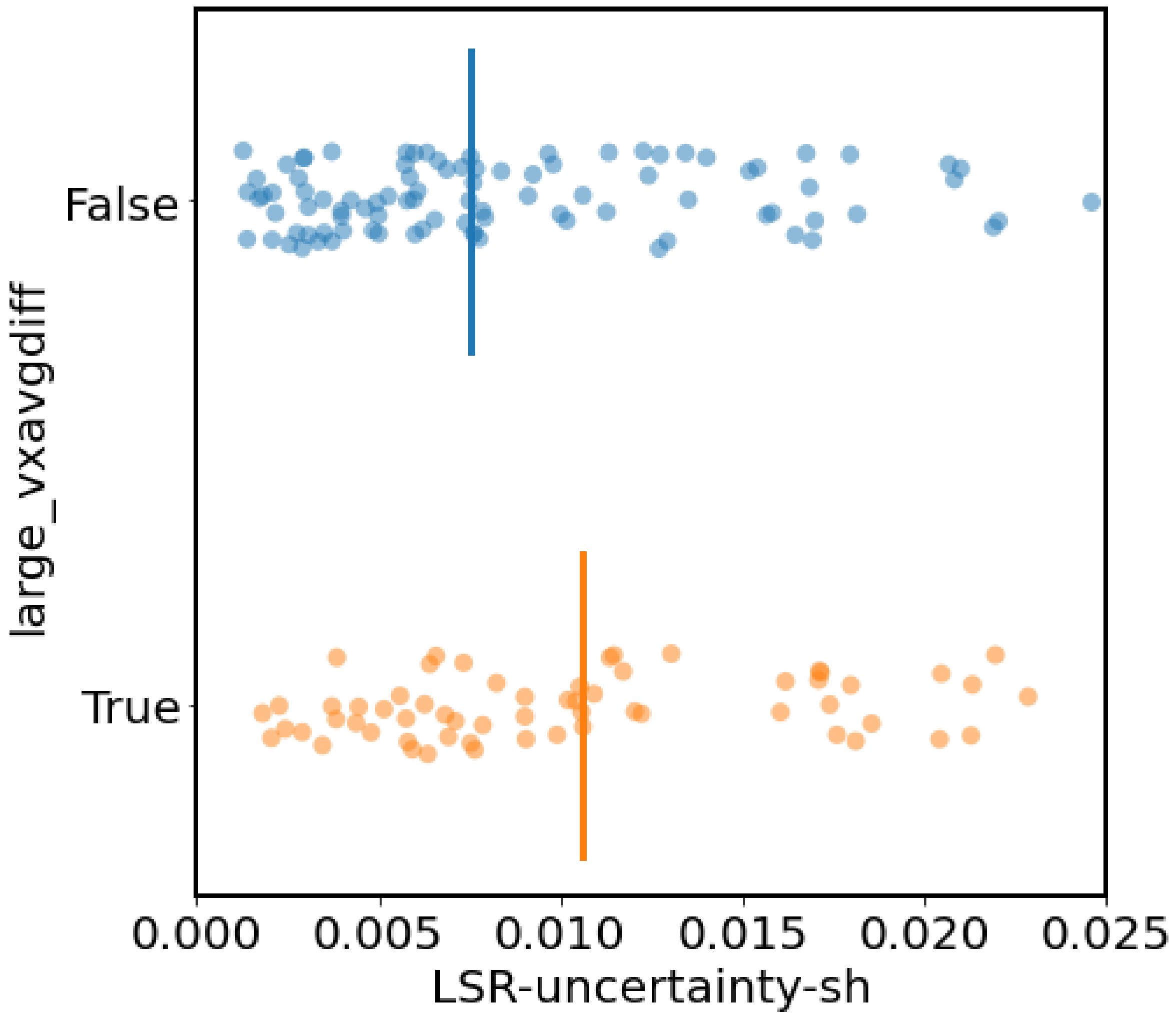
```
fig, axs = plt.subplots(1, 2, figsize=(16, 7), constrained_layout=True)
axs[0].axhline(y=0.004, linestyle='--', color='gray')
sns.scatterplot(data=df, x='SAV-uncertainty-x', y='LSR-uncertainty-sh', hue=
↪'Invalid+Incorrect', palette="mako_r", ax=axs[0], s=60)
axs[0].set_ylim(0, 0.06)
axs[0].set_xlim(0, 0.8)
sns.stripplot(data=df, x="LSR-uncertainty-sh", y="large_vxdiff", s=8, alpha=0.5,
             ax=axs[1])
kawgs2 = {'linewidth': 3, }

tmp = df[df['large_vxdiff'] == 'False']
tmp_mean = tmp['LSR-uncertainty-sh'].median()
axs[1].plot([tmp_mean, tmp_mean], [-0.3, 0.3], **kawgs2)

tmp = df[df['large_vxdiff'] == 'True']
tmp_mean = tmp['LSR-uncertainty-sh'].median()
axs[1].plot([tmp_mean, tmp_mean], [0.7, 1.3], **kawgs2)

axs[1].set_xlim(0, 0.025)

axs[0].text(-0.16, 1, "$\mathbf{a}$", transform=axs[0].transAxes)
axs[1].text(-0.16, 1, "$\mathbf{b}$", transform=axs[1].transAxes)

axs[0].set_xlabel('$\delta_u$ (m day$^{-1}$)')
axs[0].set_ylabel("$\delta_{x'y'}$ (day$^{-1}$)")
axs[0].get_legend().set_title("Invalid & incorrect \n matches (%)")
axs[1].set_xlabel("$\delta_{x'y'}$ (day$^{-1}$)")
axs[1].set_ylabel("Oberved flow speed deviation from \n ground truth (GNSS) larger␣
↪than $\delta_u$?")

# save figure
fig.patch.set_facecolor('xkcd:white')
fig.savefig('Fig5.png', dpi=200)
```



14.1 Additional notes

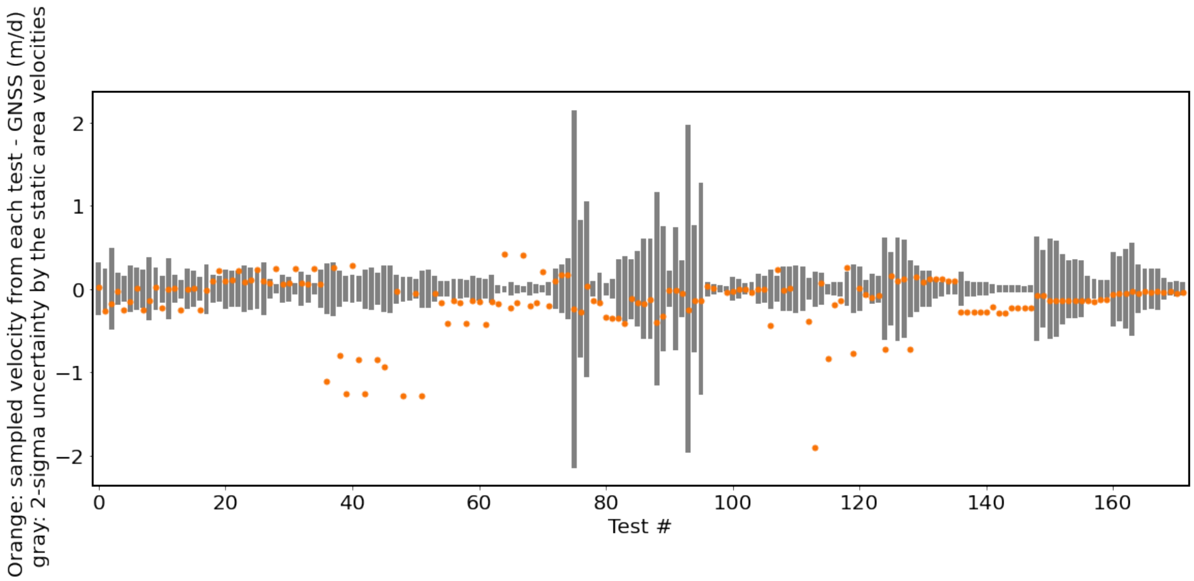
1. If we replace `large_vxdiff` with `large_vxavgdiff` (which is the status averaged from the nearest 3x3 array), the results won't change much.

```
kawgs2 = {'linewidth': 3, }
fig, ax = plt.subplots(1, 1, figsize=(7, 7))
sns.stripplot(data=df, x="LSR-uncertainty-sh", y="large_vxavgdiff", s=8, alpha=0.5,␣
↪ax=ax)
tmp = df[df['large_vxavgdiff'] == 'False']
tmp_mean = tmp['LSR-uncertainty-sh'].median()
ax.plot([tmp_mean, tmp_mean], [-0.3, 0.3], **kawgs2)

tmp = df[df['large_vxavgdiff'] == 'True']
tmp_mean = tmp['LSR-uncertainty-sh'].median()
ax.plot([tmp_mean, tmp_mean], [0.7, 1.3], **kawgs2)
ax.set_xlim(0, 0.025);
```



2. Here shows how the static-area correct-match uncertainty (δ_x, gray bar) compares to the ground truth deviation at the GNSS station (pt0) for each test pair.

```
fig, ax = plt.subplots(1, 1, figsize=(19, 7))
plt.bar(df.index, df['SAV-uncertainty-x'], color='gray')
plt.bar(df.index, -df['SAV-uncertainty-x'], color='gray')
plt.plot(df.index, df['pt0_vxdiff'], '.', color='xkcd:orange', markersize=10)
plt.xlim(-1, 172)
plt.xlabel('Test #')
plt.ylabel('Orange: sampled velocity from each test - GNSS (m/d) \n gray: 2-sigma␣
↪uncertainty by the static area velocities');
```


Orange: sampled velocity from each test - GNSS (m/d)
gray: 2-sigma uncertainty by the static area velocities
2
1
0
−1
−2
0
20
40
60
80
100
120
140
160
Test #

CHAPTER

FIFTEEN

FIGURE 6 SCRIPT

This notebook requires additional steps to be reproduced:

1. Make sure you have installed all necessary packages, including vmap and others specified in the cell below.
2. Update the input file paths so they point to the actual files on your local machine. These files include the source Landsat 8 image (`LC08_L1TP_061018_20180304_20180319_01_T1_B8_s.TIF`, downloadable at https://doi.org/10.17605/OSF.IO/HE7YR) and the RGI glacier outline (`01_rgi60_Alaska.shp`, downloadable at https://doi.org/10.7265/4m1f-gd79).

```
import numpy as np
import matplotlib.pyplot as plt
import os,sys,glob
from pygeotools.lib import geolib,iolib,warplib,malib
from imview import pltlib
import rasterio
import geopandas as gpd
```

```
%matplotlib inline
```

```
%cd /nobackup/sbhusha1/feature_tracking_wg
```

```
img_fn = 'Cropped/LC08_L1TP_061018_20180304_20180319_01_T1_B8_s.TIF'
```

```
def create_synthetic_offset(imgfile, mode='subpixel', block_size=500):
    """
    imgfile: str, geotiff file path
    mode: 'subpixel' or 'multipixel'
    block_size: int, increment block size
    ----
    returns:
    shift_arx: np.ndarray, offset field (x), in pixels
    shift_ary: np.ndarray, offset field (y), in pixels
    ----
    """
    with rasterio.open(imgfile) as src:
        data_shape = (src.height, src.width)
    idxy, idxx = np.indices(data_shape)
    # for Numpy array, first is row element (-> geotiff's y direction, height)
    # and second is column element (-> geotiff's x direction, width)

    if mode == 'subpixel':
```

(continues on next page)

(continued from previous page)

```
        shift_arx = idxx // block_size
        shift_arx = 0.1 * shift_arx + 0.1
        shift_ary = idxy // block_size
        shift_ary = -0.1 * shift_ary - 0.1
    elif mode == 'multipixel':
        shift_arx = 1 + idxx // block_size
        shift_ary = -1 - idxy // block_size
    else:
        raise ValueError('Mode is not defined.')

    return shift_arx, shift_ary

def apply_synthetic_offset(imgfile, shift_arx, shift_ary, spline_order=1):
    """
    imgfile: str, geotiff file path
    shift_arx: np.ndarray, offset field (x) from gftt.create_synthetic_offset
    shift_ary: np.ndarray, offset field (y) from gftt.create_synthetic_offset
    ----
    returns:
    ----
    """
    import rasterio
    from scipy.ndimage import map_coordinates
    with rasterio.open(imgfile) as src:
        data_shape = (src.height, src.width)
        data = src.read(1)
    idxy, idxx = np.indices(data_shape)
    shifted_y = idxy + shift_ary
    shifted_x = idxx + shift_arx
    shifted_yx = np.vstack((shifted_y.flatten(), shifted_x.flatten()))

    shifted_val = map_coordinates(data, shifted_yx, order=spline_order, mode='nearest
↪')
    shifted_val = np.reshape(shifted_val, data_shape)

    return shifted_val
```

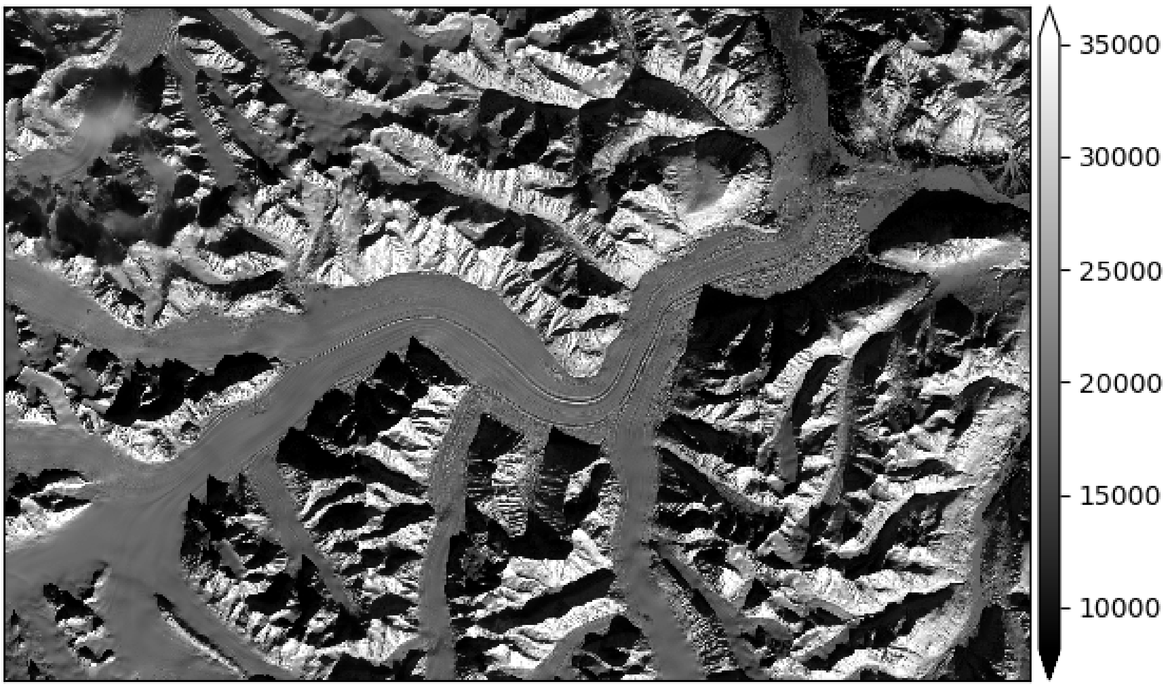
```
data_shape = img.shape
shift_arx, shift_ary = create_synthetic_offset(img_fn)
shift_arx
```

```
array([[0.1, 0.1, 0.1, ..., 0.8, 0.8, 0.8],
       [0.1, 0.1, 0.1, ..., 0.8, 0.8, 0.8],
       [0.1, 0.1, 0.1, ..., 0.8, 0.8, 0.8],
       ...,
       [0.1, 0.1, 0.1, ..., 0.8, 0.8, 0.8],
       [0.1, 0.1, 0.1, ..., 0.8, 0.8, 0.8],
       [0.1, 0.1, 0.1, ..., 0.8, 0.8, 0.8]])
```

```
shifted_image = apply_synthetic_offset(img_fn, shift_arx, shift_ary)
outfn = os.path.splitext(img_fn)[0]+'_subpixel_shift.tif'
iolib.writeGTiff(shifted_image,outfn,src_ds=ds_list[2],ndv=0)
```

```
f,ax = plt.subplots()
pltlib.iv(iolib.fn_getma(outfn),ax=ax,cmap='gray')
```

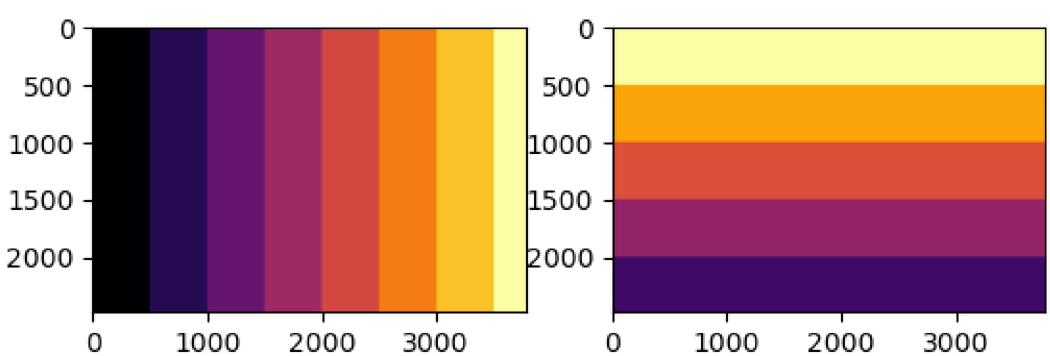
```
<Axes: >
```




```
!vmap.py $img_fn $outfn -dt none
```

```
out_disp_fn = 'LC08_L1TP_061018_20180304_20180319_01_T1_B8_s__LC08_L1TP_061018_
↪20180304_20180319_01_T1_B8_s_subpixel_shift_vmap_minm_35px_spm1/vmap-F.tif'
out_dx,out_dy = [iolib.fn_getma(out_disp_fn,b) for b in [1,2]]
```

```
f,ax = plt.subplots(1,2)
clim_x = (0.1,0.8)
clim_y = (-0.6,-0.1)
cb1 = ax[0].imshow(shift_arx,cmap='inferno',clim=clim_x)
cb2 = ax[1].imshow(shift_ary,cmap='inferno',clim=clim_y)
```



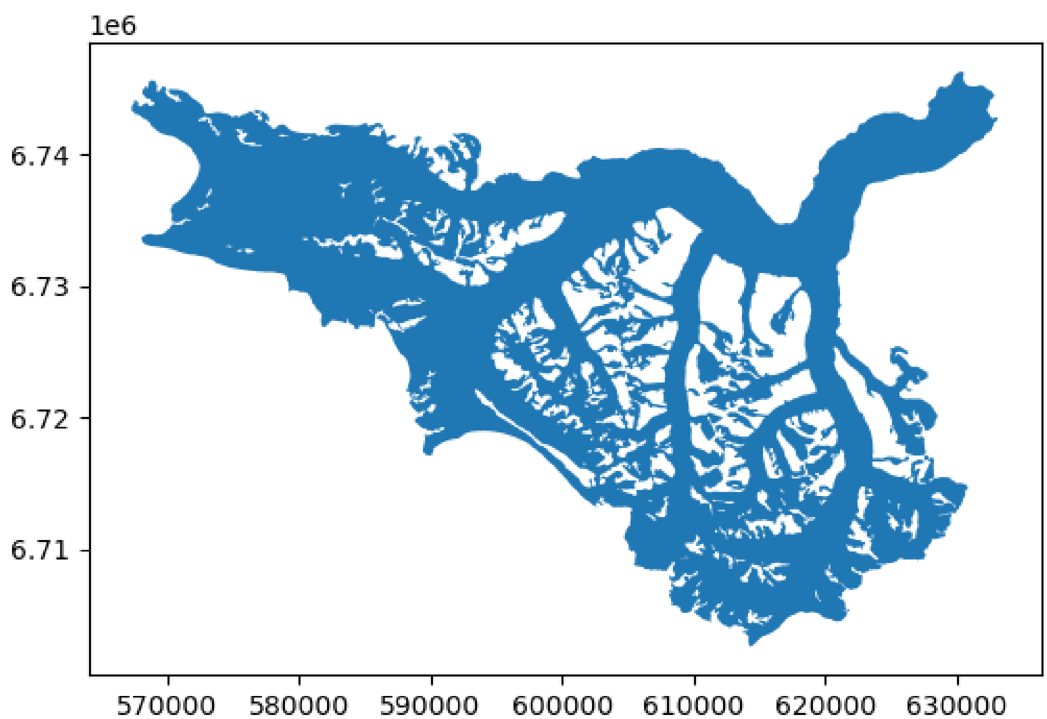
```
ds = iolib.fn_getds(out_disp_fn)
extent = geolib.ds_extent(ds)
fig_extent = [extent[0],extent[2],extent[1],extent[3]]
```

```
glac_shp = gpd.read_file('/nobackup/sbhusha1/reference_data/rgi60/regions/01_rgi60_
↪Alaska.shp')
```

```
kaskwulsh_mask = glac_shp['RGIId'] == 'RGI60-01.16201'
kaskwulsh_shp = (glac_shp[kaskwulsh_mask]).to_crs("EPSG:32607")
```

```
kaskwulsh_shp.plot()
```

```
<Axes: >
```

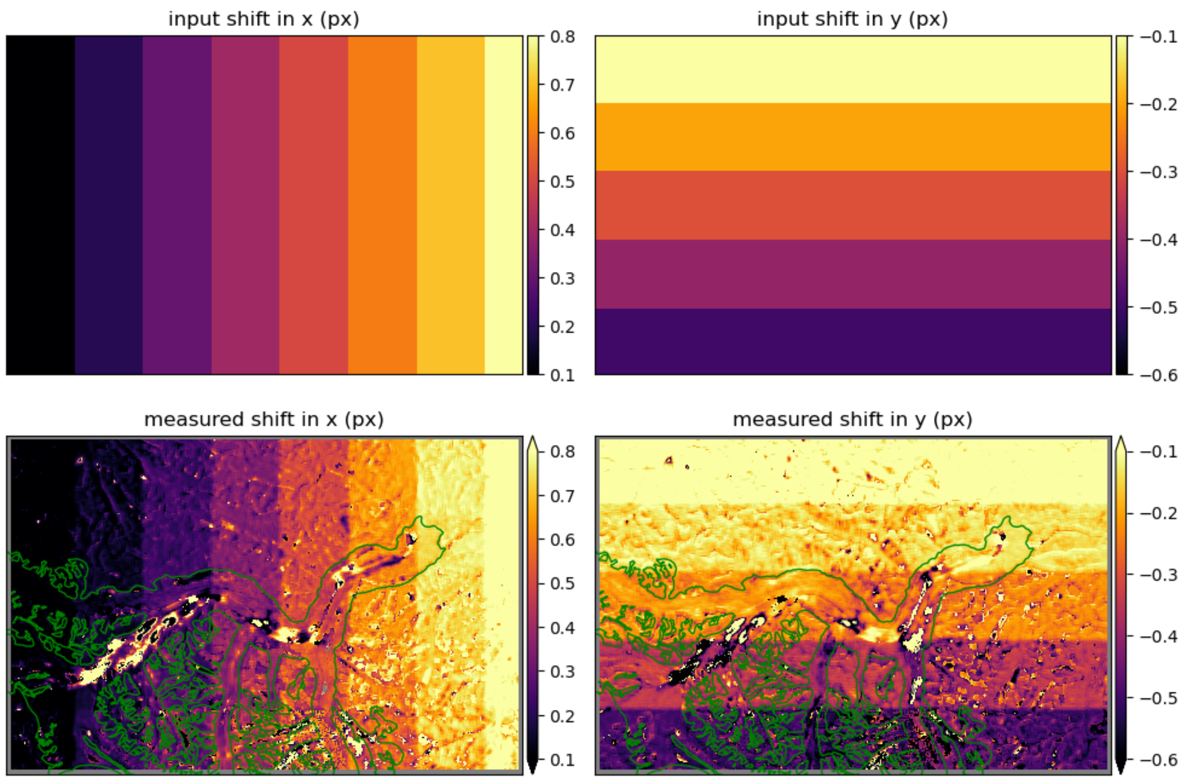



```
f,axa = plt.subplots(2,2,figsize=(10,8),sharex=True,sharey=True)
ax = axa.ravel()
clim_x = (0.1,0.8)
clim_y = (-0.6,-0.1)
pltlib.iv(shift_arx,ax=ax[0],cmap='inferno',clim=clim_x,title='input shift in x (px)',
↪extent=fig_extent)
pltlib.iv(shift_ary,ax=ax[1],cmap='inferno',clim=clim_y,title='input shift in y (px)',
↪extent=fig_extent)
kaskwulsh_shp.plot(ax=ax[2],facecolor="None",edgecolor='green',linewidth=0.95)
kaskwulsh_shp.plot(ax=ax[3],facecolor="None",edgecolor='green',linewidth=0.95)
pltlib.iv(-1*out_dx,ax=ax[2],cmap='inferno',clim=clim_x,cbar=False,title='measured␣
↪shift in x (px)',extent=fig_extent)
pltlib.add_cbar(ax[2],mappable=cb1)
```

(continues on next page)

(continued from previous page)

```
pltlib.iv(-1*out_dy,ax=ax[3],cmap='inferno',clim=clim_y,cbar=False,title='measured␣
↪shift in y (px)',extent=fig_extent)

pltlib.add_cbar(ax[3],cb2)
plt.tight_layout()
#f.savefig('figure6_manuscript.png',dpi=300,bbox_inches='tight', pad_inches=0.1)
f.savefig('/nobackup/sbhusha1/notebooks/velocity/figure6_manuscript_revised_
↪kaskawulsh_only.png',dpi=300,bbox_inches='tight', pad_inches=0.1)
```



CHAPTER

SIXTEEN

FIGURE 7 SCRIPT

```
import matplotlib.pyplot as plt
from matplotlib.colors import LinearSegmentedColormap
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib as mpl
```

```
# font and linewidth settings
font = {'size'   : 20}
mpl.rc('font', **font)
mpl.rc('legend', fontsize=16)
axes_settings = {'linewidth'   : 2}
mpl.rc('axes', **axes_settings)
```

Now read the data:

```
df = pd.read_csv('../results_ITSLIVE.csv', dtype=str)
for field in ['Assigned-x-error',
              'Assigned-y-error',
              'SAV-uncertainty-x',
              'SAV-uncertainty-y',
              'SAV-peak-x',
              'SAV-peak-y',
              'SAV-outlier-percent',
              'LSR-uncertainty-nm',
              'LSR-uncertainty-sh',
              ]:
    df[field] = df[field].astype(float)
```

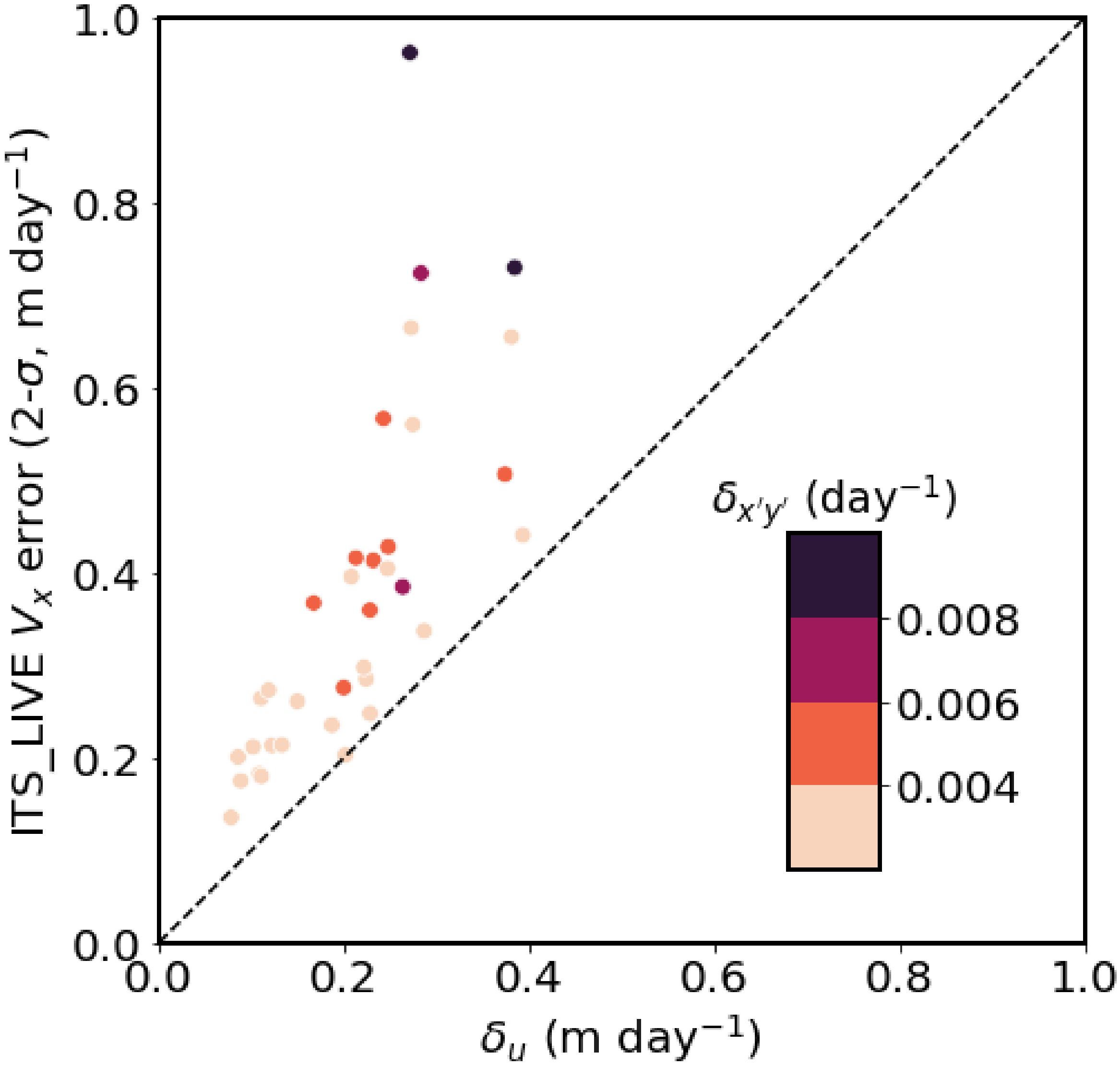
Create additional column fields for grouping data:

```
df['SAV-uncertainty-x-m/day'] = df['SAV-uncertainty-x'] / 365
df['SAV-uncertainty-y-m/day'] = df['SAV-uncertainty-y'] / 365
df['Assigned-x-error-m/day'] = df['Assigned-x-error'] / 365
df['Assigned-y-error-m/day'] = df['Assigned-y-error'] / 365
df['LSR-uncertainty-nm-1/day'] = df['LSR-uncertainty-nm'] / 365
df['LSR-uncertainty-sh-1/day'] = df['LSR-uncertainty-sh'] / 365
df['Assigned-x-error-m/day-95CI'] = df['Assigned-x-error-m/day'] * 2
df['Assigned-y-error-m/day-95CI'] = df['Assigned-y-error-m/day'] * 2
```

Make custom colormap:

```
cmap = sns.color_palette("rocket_r", as_cmap=True)
cmaplist = [cmap(i) for i in range(15, cmap.N, 70)]
discrete_cmap = LinearSegmentedColormap.from_list('Custom rocket_r', cmaplist, len(cmaplist))
discrete_cmap
```

```
fig, ax = plt.subplots(1, 1, figsize=(7, 7), constrained_layout=True)
sns.scatterplot(data=df, x='SAV-uncertainty-x-m/day', y='Assigned-x-error-m/day-95CI', hue='LSR-uncertainty-sh-1/day', hue_norm=(0.002,0.01), palette=discrete_cmap, s=60, ax=ax, legend=False)
ax.plot([0, 1], [0, 1], '--', color='k')
ax.set_aspect('equal', adjustable='box')
ax.set_xlim(0, 1)
ax.set_ylim(0, 1)
ax.set_xlabel('$\delta_u$ (m day$^{-1}$)')
ax.set_ylabel('ITS_LIVE $V_x$ error (2-$\sigma$, m day$^{-1}$)')


norm = mpl.colors.Normalize(vmin=0.002, vmax=0.01)
mappable = mpl.cm.ScalarMappable(norm=norm, cmap=discrete_cmap)

ax2 = fig.add_axes([0.7, 0.2, 0.08, 0.3])
fig.colorbar(mappable, cax=ax2, orientation='vertical', ticks=[0.004, 0.006, 0.008])

ax2.set_title("$\delta_{x'y'}$ (day$^{-1}$)", size=19)

## legend:
## < 0.004 lvl 1
## 0.004 - 0.006 lvl 2
## 0.006 - 0.008 lvl 3
## > 0.008 lvl 4

# save figure
fig.patch.set_facecolor('xkcd:white')
fig.savefig('Fig7.png', dpi=200)
```


ITS_LIVE V_x error (2-σ, m day^{-1})
δ_u (m day^{-1})
$\delta_{x'y'}$ (day^{-1})
0.008
0.006
0.004
0.0
0.2
0.4
0.6
0.8
1.0

Part IV

Intermediate processing steps

CHAPTER SEVENTEEN

GNSS DATA PROCESSING SCRIPT

Note: This notebook is not reproducible because the source data paths direct to a large database. It is meant to give readers a sense about how the provided GNSS records (`Kaskawulsh_v2_2018-mm-dd_to_2018-mm-dd_GPS.csv`) were made.

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
import pandas as pd
import geopandas as gpd
import os
from datetime import datetime
from scipy import interpolate
import math
import glob
from pyproj import transform, CRS
from math import degrees, atan2
# from scipy.stats import linregress
import scipy.stats
from netCDF4 import Dataset, num2date
import calendar
from scipy.io import loadmat

pd.set_option('display.max_columns', None)

from IPython.core.display import display, HTML
display(HTML("<style>.container { width:100% !important; }</style>"))

def convert_time(seconds):
    seconds = seconds % (24 * 3600)
    hour = seconds // 3600
    seconds %= 3600
    minutes = seconds // 60
    seconds %= 60

    return "%d:%02d:%02d" % (hour, minutes, seconds)

def custom_mean(df):
    return df.mean(skipna=False)

def rsquared(x, y):
    """ Return R^2 where x and y are array-like."""
    slope, intercept, r_value, p_value, std_err = scipy.stats.linregress(x, y)
    return r_value**2
```

(continues on next page)

(continued from previous page)

```
def truncate(number, decimals=0):
    """
    Returns a value truncated to a specific number of decimal places.
    """
    if not isinstance(decimals, int):
        raise TypeError("decimal places must be an integer.")
    elif decimals < 0:
        raise ValueError("decimal places has to be 0 or more.")
    elif decimals == 0:
        return math.trunc(number)

    factor = 10.0 ** decimals
    return math.trunc(number * factor) / factor

horizontal_error = 0.12
vertical_error = 0.2
```

```
# #read Lower data
lower_path='/Users/willkochtitzky/Projects/Kaskawulsh/GPS_data/lower_cleaned.csv'
#read in PPP corrected data (from NR CAN)
#read lower data
lower_pos_data = pd.read_csv(lower_path)#,usecols=['DIR','HGT(m)','YEAR-MM-DD',
↪'HR:MN:SS.SS','UTM_EASTING','UTM_NORTHING','GDOP','RMSC(m)','RMSP(m)','SDLAT(95%)',
↪'SDLON(95%)','SDHGT(95%)','SDCLK(95%)','SDTZD(95%)'])

lower_pos_data['date'] = pd.to_datetime(lower_pos_data['YEAR-MM-DD']+'/'+lower_pos_
↪data['HR:MN:SS.SS'],format='%Y-%m-%d/%H:%M:%S.%f')
lower_pos_data['day_date'] = pd.to_datetime(lower_pos_data['YEAR-MM-DD'],format='%Y-
↪%m-%d')
lower_pos_data = lower_pos_data.drop(columns=['DIR','YEAR-MM-DD','HR:MN:SS.SS'])

# lower_pos_data[['HGT(m)','UTM_EASTING','UTM_NORTHING','GDOP','RMSC(m)','RMSP(m)',
↪'SDLAT(95%)','SDLON(95%)','SDHGT(95%)','SDCLK(95%)','SDTZD(95%)']] = lower_pos_
↪data[['HGT(m)','UTM_EASTING','UTM_NORTHING','GDOP','RMSC(m)','RMSP(m)','SDLAT(95%)',
↪'SDLON(95%)','SDHGT(95%)','SDCLK(95%)','SDTZD(95%)']].apply(pd.to_numeric)#change␣
↪the data types
lower_pos_data = lower_pos_data.sort_values(by=['date'])#sort the data

#filter out good data to inspect the bad data later
lower_pos_data_bad_data = lower_pos_data[(lower_pos_data['SDLAT(95%)']>horizontal_
↪error) & (lower_pos_data['SDLON(95%)']>horizontal_error) & (lower_pos_data['SDHGT(95
↪%)']>vertical_error)]

lower_pos_data = lower_pos_data[lower_pos_data['SDLAT(95%)']<horizontal_error]
lower_pos_data = lower_pos_data[lower_pos_data['SDLON(95%)']<horizontal_error]
lower_pos_data = lower_pos_data[lower_pos_data['SDHGT(95%)']<vertical_error]

lower_pos_data['year'] = lower_pos_data['date'].dt.year
lower_pos_data['day'] = lower_pos_data['date'].dt.dayofyear
lower_pos_data = lower_pos_data.reset_index() #reset the index for the calculations␣
↪ahead, this is crucial to make sure your counter is right!

#calculate how much time has occured since 1/1/00 so that you can figure out how much␣
↪time is between observations
```

(continues on next page)

(continued from previous page)

```
pos_sec_since = []
jan1_2007 = datetime(2007,1,1)
for i in range(0,len(lower_pos_data)):
    pos_sec_since.append((lower_pos_data['date'][i]-jan1_2007).total_seconds())
lower_pos_data['sec_since']=pos_sec_since

lower_pos_data_daily = lower_pos_data.set_index('date').groupby(pd.Grouper(freq='d')).
↪mean() #calculate daily data
lower_pos_data_daily = lower_pos_data_daily.reset_index() #reset the index after you␣
↪claculate daily data

distance=[float('nan')]
obs_duration_days=[float('nan')]
for i in range(1,len(lower_pos_data_daily)):
    distance.append(np.sqrt((lower_pos_data_daily['UTM_EASTING'][i]-lower_pos_data_
↪daily['UTM_EASTING'][i-1])**2+(lower_pos_data_daily['UTM_NORTHING'][i]-lower_pos_
↪data_daily['UTM_NORTHING'][i-1])**2))
    obs_duration_days.append((lower_pos_data_daily['sec_since'][i]-lower_pos_data_
↪daily['sec_since'][i-1])/3600/24)
lower_pos_data_daily['Distance']=distance
lower_pos_data_daily['obs_duration_days']=obs_duration_days
lower_pos_data_daily[['Distance','obs_duration_days']] = lower_pos_data_daily[[
↪'Distance','obs_duration_days']].apply(pd.to_numeric)#change the data types
lower_pos_data_daily['Velocity_m_per_d']=lower_pos_data_daily['Distance']/lower_pos_
↪data_daily['obs_duration_days']
```

```
#read middle data
#middle_path='/Users/willkochtitzky/Projects/Kaskawulsh/GPS_data/Middle_all_data.csv'
middle_path='/Users/willkochtitzky/Projects/Kaskawulsh/GPS_data/middle_cleaned.csv'
#read in PPP corrected data (from NR CAN)
middle_pos_data = pd.read_csv(middle_path)#,usecols=['DIR','HGT(m)','YEAR-MM-DD',
↪'HR:MN:SS.SS','UTM_EASTING','UTM_NORTHING','GDOP','RMSC(m)','RMSP(m)','SDLAT(95%)',
↪'SDLON(95%)','SDHGT(95%)','SDCLK(95%)','SDTZD(95%)'])

middle_pos_data['date'] = pd.to_datetime(middle_pos_data['YEAR-MM-DD']+'/'+middle_pos_
↪data['HR:MN:SS.SS'],format='%Y-%m-%d/%H:%M:%S.%f')
middle_pos_data['day_date'] = pd.to_datetime(middle_pos_data['YEAR-MM-DD'],format='%Y-
↪%m-%d')
middle_pos_data = middle_pos_data.drop(columns=['DIR','YEAR-MM-DD','HR:MN:SS.SS'])

# middle_pos_data[['HGT(m)','UTM_EASTING','UTM_NORTHING','GDOP','RMSC(m)','RMSP(m)',
↪'SDLAT(95%)','SDLON(95%)','SDHGT(95%)','SDCLK(95%)','SDTZD(95%)']] = middle_pos_
↪data[['HGT(m)','UTM_EASTING','UTM_NORTHING','GDOP','RMSC(m)','RMSP(m)','SDLAT(95%)',
↪'SDLON(95%)','SDHGT(95%)','SDCLK(95%)','SDTZD(95%)']].apply(pd.to_numeric)#change␣
↪the data types
middle_pos_data = middle_pos_data.sort_values(by=['date'])#sort the data

#filter out good data to inspect the bad data later
middle_pos_data_bad_data = middle_pos_data[(middle_pos_data['SDLAT(95%)']>horizontal_
↪error) & (middle_pos_data['SDLON(95%)']>horizontal_error) & (middle_pos_data[
↪'SDHGT(95%)']>vertical_error)]

middle_pos_data = middle_pos_data[middle_pos_data['SDLAT(95%)']<horizontal_error]
middle_pos_data = middle_pos_data[middle_pos_data['SDLON(95%)']<horizontal_error]
middle_pos_data = middle_pos_data[middle_pos_data['SDHGT(95%)']<vertical_error]
```

(continues on next page)

(continued from previous page)

```
middle_pos_data['year'] = middle_pos_data['date'].dt.year
middle_pos_data['day'] = middle_pos_data['date'].dt.dayofyear
middle_pos_data = middle_pos_data.reset_index() #reset the index for the calculations␣
↪ahead, this is crucial to make sure your counter is right!

#calculate how much time has occured since 1/1/00 so that you can figure out how much␣
↪time is between observations
pos_sec_since = []
jan1_2007 = datetime(2007,1,1)
for i in range(0,len(middle_pos_data)):
    pos_sec_since.append((middle_pos_data['date'][i]-jan1_2007).total_seconds())
middle_pos_data['sec_since']=pos_sec_since

middle_pos_data_daily = middle_pos_data.set_index('date').groupby(pd.Grouper(freq='d
↪')).mean() #calculate daily data
middle_pos_data_daily = middle_pos_data_daily.reset_index() #reset the index after␣
↪you claculate daily data

distance=[float('nan')]
obs_duration_days=[float('nan')]
for i in range(1,len(middle_pos_data_daily)):
    distance.append(np.sqrt((middle_pos_data_daily['UTM_EASTING'][i]-middle_pos_data_
↪daily['UTM_EASTING'][i-1])**2+(middle_pos_data_daily['UTM_NORTHING'][i]-middle_pos_
↪data_daily['UTM_NORTHING'][i-1])**2))
    obs_duration_days.append((middle_pos_data_daily['sec_since'][i]-middle_pos_data_
↪daily['sec_since'][i-1])/3600/24)
middle_pos_data_daily['Distance']=distance
middle_pos_data_daily['obs_duration_days']=obs_duration_days
middle_pos_data_daily[['Distance','obs_duration_days']] = middle_pos_data_daily[[
↪'Distance','obs_duration_days']].apply(pd.to_numeric)#change the data types
middle_pos_data_daily['Velocity_m_per_d']=middle_pos_data_daily['Distance']/middle_
↪pos_data_daily['obs_duration_days']
```

```
#read Upper data
upper_path='/Users/willkochtitzky/Projects/Kaskawulsh/GPS_data/upper_cleaned.csv'
#read in PPP corrected data (from NR CAN)
upper_pos_data = pd.read_csv(upper_path)#,usecols=['DIR','HGT(m)','YEAR-MM-DD',
↪'HR:MN:SS.SS','UTM_EASTING','UTM_NORTHING','GDOP','RMSC(m)','RMSP(m)','SDLAT(95%)',
↪'SDLON(95%)','SDHGT(95%)','SDCLK(95%)','SDTZD(95%)'])
upper_pos_data['date'] = pd.to_datetime(upper_pos_data['YEAR-MM-DD']+'/'+upper_pos_
↪data['HR:MN:SS.SS'],format='%Y-%m-%d/%H:%M:%S.%f')
upper_pos_data['day_date'] = pd.to_datetime(upper_pos_data['YEAR-MM-DD'],format='%Y-
↪%m-%d')
upper_pos_data = upper_pos_data.drop(columns=['DIR','YEAR-MM-DD','HR:MN:SS.SS'])

# upper_pos_data[['HGT(m)','UTM_EASTING','UTM_NORTHING','GDOP','RMSC(m)','RMSP(m)',
↪'SDLAT(95%)','SDLON(95%)','SDHGT(95%)','SDCLK(95%)','SDTZD(95%)']] = upper_pos_
↪data[['HGT(m)','UTM_EASTING','UTM_NORTHING','GDOP','RMSC(m)','RMSP(m)','SDLAT(95%)',
↪'SDLON(95%)','SDHGT(95%)','SDCLK(95%)','SDTZD(95%)']].apply(pd.to_numeric)#change␣
↪the data types
upper_pos_data = upper_pos_data.sort_values(by=['date'])#sort the data

#filter out good data to inspect the bad data later
upper_pos_data_bad_data = upper_pos_data[(upper_pos_data['SDLAT(95%)']>horizontal_
↪error) & (upper_pos_data['SDLON(95%)']>horizontal_error) & (upper_pos_data['SDHGT(95
↪%)']>vertical_error)]
```

(continues on next page)

(continued from previous page)

```
upper_pos_data = upper_pos_data[upper_pos_data['SDLAT(95%)']<horizontal_error]
upper_pos_data = upper_pos_data[upper_pos_data['SDLON(95%)']<horizontal_error]
upper_pos_data = upper_pos_data[upper_pos_data['SDHGT(95%)']<vertical_error]

upper_pos_data['year'] = upper_pos_data['date'].dt.year
upper_pos_data['day'] = upper_pos_data['date'].dt.dayofyear
upper_pos_data = upper_pos_data.reset_index() #reset the index for the calculations␣
↪ahead, this is crucial to make sure your counter is right!

#calculate how much time has occured since 1/1/00 so that you can figure out how much␣
↪time is between observations
pos_sec_since = []
jan1_2007 = datetime(2007,1,1)
for i in range(0,len(upper_pos_data)):
    pos_sec_since.append((upper_pos_data['date'][i]-jan1_2007).total_seconds())
upper_pos_data['sec_since']=pos_sec_since

upper_pos_data_daily = upper_pos_data.set_index('date').groupby(pd.Grouper(freq='d')).
↪mean() #calculate daily data
upper_pos_data_daily = upper_pos_data_daily.reset_index() #reset the index after you␣
↪claculate daily data

distance=[float('nan')]
obs_duration_days=[float('nan')]
for i in range(1,len(upper_pos_data_daily)):
    distance.append(np.sqrt((upper_pos_data_daily['UTM_EASTING'][i]-upper_pos_data_
↪daily['UTM_EASTING'][i-1])**2+(upper_pos_data_daily['UTM_NORTHING'][i]-upper_pos_
↪data_daily['UTM_NORTHING'][i-1])**2))
    obs_duration_days.append((upper_pos_data_daily['sec_since'][i]-upper_pos_data_
↪daily['sec_since'][i-1])/3600/24)
upper_pos_data_daily['Distance']=distance
upper_pos_data_daily['obs_duration_days']=obs_duration_days
upper_pos_data_daily[['Distance','obs_duration_days']] = upper_pos_data_daily[[
↪'Distance','obs_duration_days']].apply(pd.to_numeric)#change the data types
upper_pos_data_daily['Velocity_m_per_d']=upper_pos_data_daily['Distance']/upper_pos_
↪data_daily['obs_duration_days']
```

```
#read Arm data
arm_path='/Users/willkochtitzky/Projects/Kaskawulsh/GPS_data/arm_cleaned.csv'
#read in PPP corrected data (from NR CAN)
arm_pos_data = pd.read_csv(arm_path)#,usecols=['DIR','HGT(m)','YEAR-MM-DD','HR:MN:SS.
↪SS','UTM_EASTING','UTM_NORTHING','GDOP','RMSC(m)','RMSP(m)','SDLAT(95%)','SDLON(95%)
↪','SDHGT(95%)','SDCLK(95%)','SDTZD(95%)'])
arm_pos_data['date'] = pd.to_datetime(arm_pos_data['YEAR-MM-DD']+'/'+arm_pos_data[
↪'HR:MN:SS.SS'],format='%Y-%m-%d/%H:%M:%S.%f')
arm_pos_data['day_date'] = pd.to_datetime(arm_pos_data['YEAR-MM-DD'],format='%Y-%m-%d
↪')
arm_pos_data = arm_pos_data.drop(columns=['DIR','YEAR-MM-DD','HR:MN:SS.SS'])

# arm_pos_data[['HGT(m)','UTM_EASTING','UTM_NORTHING','GDOP','RMSC(m)','RMSP(m)',
↪'SDLAT(95%)','SDLON(95%)','SDHGT(95%)','SDCLK(95%)','SDTZD(95%)']] = arm_pos_data[[
↪'HGT(m)','UTM_EASTING','UTM_NORTHING','GDOP','RMSC(m)','RMSP(m)','SDLAT(95%)',
↪'SDLON(95%)','SDHGT(95%)','SDCLK(95%)','SDTZD(95%)']].apply(pd.to_numeric)#change␣
↪the data types
```

(continues on next page)

(continued from previous page)

```
arm_pos_data = arm_pos_data.sort_values(by=['date'])#sort the data

#filter out good data to inspect the bad data later
arm_pos_data_bad_data = arm_pos_data[(arm_pos_data['SDLAT(95%)']>horizontal_error) &␣
↪(arm_pos_data['SDLON(95%)']>horizontal_error) & (arm_pos_data['SDHGT(95%)']>
↪vertical_error)]

arm_pos_data = arm_pos_data[arm_pos_data['SDLAT(95%)']<horizontal_error]
arm_pos_data = arm_pos_data[arm_pos_data['SDLON(95%)']<horizontal_error]
arm_pos_data = arm_pos_data[arm_pos_data['SDHGT(95%)']<vertical_error]

arm_pos_data['year'] = arm_pos_data['date'].dt.year
arm_pos_data['day'] = arm_pos_data['date'].dt.dayofyear
arm_pos_data = arm_pos_data.reset_index() #reset the index for the calculations ahead,
↪ this is crucial to make sure your counter is right!

#calculate how much time has occured since 1/1/00 so that you can figure out how much␣
↪time is between observations
pos_sec_since = []
jan1_2007 = datetime(2007,1,1)
for i in range(0,len(arm_pos_data)):
    pos_sec_since.append((arm_pos_data['date'][i]-jan1_2007).total_seconds())
arm_pos_data['sec_since']=pos_sec_since

arm_pos_data_daily = arm_pos_data.set_index('date').groupby(pd.Grouper(freq='d')).
↪mean() #calculate daily data
arm_pos_data_daily = arm_pos_data_daily.reset_index() #reset the index after you␣
↪claculate daily data

distance=[float('nan')]
obs_duration_days=[float('nan')]
for i in range(1,len(arm_pos_data_daily)):
    distance.append(np.sqrt((arm_pos_data_daily['UTM_EASTING'][i]-arm_pos_data_daily[
↪'UTM_EASTING'][i-1])**2+(arm_pos_data_daily['UTM_NORTHING'][i]-arm_pos_data_daily[
↪'UTM_NORTHING'][i-1])**2))
    obs_duration_days.append((arm_pos_data_daily['sec_since'][i]-arm_pos_data_daily[
↪'sec_since'][i-1])/3600/24)
arm_pos_data_daily['Distance']=distance
arm_pos_data_daily['obs_duration_days']=obs_duration_days
arm_pos_data_daily[['Distance','obs_duration_days']] = arm_pos_data_daily[['Distance',
↪'obs_duration_days']].apply(pd.to_numeric)#change the data types
arm_pos_data_daily['Velocity_m_per_d']=arm_pos_data_daily['Distance']/arm_pos_data_
↪daily['obs_duration_days']
```

```
#calculate velocity between two dates:

# Whyjay project
start_dates = ['2018-05-08', '2018-09-03', '2018-08-18', '2018-08-02',
               '2018-04-28', '2018-04-12', '2018-05-23', '2018-04-21',
               '2018-04-05', '2018-03-04', '2018-09-30', '2018-09-20',
               '2018-09-10', '2018-08-31', '2018-08-11', '2018-08-01',
               '2018-07-27', '2018-07-22', '2018-06-27', '2018-06-12',
               '2018-05-23', '2018-05-18', '2018-05-08', '2018-03-29',
               '2018-03-14', '2018-03-04', '2018-09-17', '2018-08-18',
               '2018-07-29', '2018-07-24', '2018-07-04', '2018-06-19',
```

(continues on next page)

(continued from previous page)

```
               '2018-05-15', '2018-03-16', '2018-03-06']
end_dates = ['2018-06-27', '2018-10-05', '2018-09-03', '2018-08-18',
             '2018-08-02', '2018-04-28', '2018-06-08', '2018-05-23',
             '2018-04-21', '2018-04-05', '2018-10-05', '2018-09-30',
             '2018-09-20', '2018-09-10', '2018-08-31', '2018-08-11',
             '2018-08-01', '2018-07-27', '2018-07-22', '2018-06-27',
             '2018-06-12', '2018-05-23', '2018-05-18', '2018-05-08',
             '2018-03-29', '2018-03-14', '2018-10-02', '2018-09-17',
             '2018-08-18', '2018-07-29', '2018-07-24', '2018-07-04',
             '2018-06-19', '2018-05-15', '2018-03-16']


for i in range(0,len(start_dates)):
    vel_date_start=pd.to_datetime(start_dates[i],format='%Y-%m-%d')
    vel_date_end=pd.to_datetime(end_dates[i],format='%Y-%m-%d')

    #Lower
    for i in range(0,len(lower_pos_data_daily)):
        if lower_pos_data_daily['date'][i]==vel_date_start:
            start_pos= lower_pos_data_daily[['date','UTM_EASTING','UTM_NORTHING',
↪'HGT(m)','sec_since']].iloc[i]
        if lower_pos_data_daily['date'][i]==vel_date_end:
            end_pos=lower_pos_data_daily[['date','UTM_EASTING','UTM_NORTHING','HGT(m)
↪','sec_since']].iloc[i]

    distance_traveled = np.sqrt((start_pos['UTM_EASTING']-end_pos['UTM_EASTING
↪'])**2+(start_pos['UTM_NORTHING']-end_pos['UTM_NORTHING'])**2)
    time_between_obs = (end_pos['sec_since']-start_pos['sec_since'])/3600/24

    velocity_obs_lower=pd.DataFrame({'date1':[vel_date_start],'date2':[vel_date_end],
↪'start_easting':[start_pos['UTM_EASTING']],'start_northing':[start_pos['UTM_NORTHING
↪']],'end_easting':[end_pos['UTM_EASTING']],'end_northing':[end_pos['UTM_NORTHING']],
↪'distance_traveled (m)':[distance_traveled],'velocity (m/d)':[distance_traveled/
↪time_between_obs]},index=[0])
    velocity_obs_lower['label']='lower'

    #Middle
    for i in range(0,len(middle_pos_data_daily)):
        if middle_pos_data_daily['date'][i]==vel_date_start:
            start_pos= middle_pos_data_daily[['date','UTM_EASTING','UTM_NORTHING',
↪'HGT(m)','sec_since']].iloc[i]
        if middle_pos_data_daily['date'][i]==vel_date_end:
            end_pos=middle_pos_data_daily[['date','UTM_EASTING','UTM_NORTHING','HGT(m)
↪','sec_since']].iloc[i]

    distance_traveled = np.sqrt((start_pos['UTM_EASTING']-end_pos['UTM_EASTING
↪'])**2+(start_pos['UTM_NORTHING']-end_pos['UTM_NORTHING'])**2)
    time_between_obs = (end_pos['sec_since']-start_pos['sec_since'])/3600/24

    velocity_obs_middle=pd.DataFrame({'date1':[vel_date_start],'date2':[vel_date_end],
↪'start_easting':[start_pos['UTM_EASTING']],'start_northing':[start_pos['UTM_NORTHING
↪']],'end_easting':[end_pos['UTM_EASTING']],'end_northing':[end_pos['UTM_NORTHING']],
↪'distance_traveled (m)':[distance_traveled],'velocity (m/d)':[distance_traveled/
↪time_between_obs]},index=[1])
    velocity_obs_middle['label']='middle'
```

(continues on next page)

(continued from previous page)

```
    #Upper
    for i in range(0,len(upper_pos_data_daily)):
        if upper_pos_data_daily['date'][i]==vel_date_start:
            start_pos= upper_pos_data_daily[['date','UTM_EASTING','UTM_NORTHING',
↪'HGT(m)','sec_since']].iloc[i]
        if upper_pos_data_daily['date'][i]==vel_date_end:
            end_pos=upper_pos_data_daily[['date','UTM_EASTING','UTM_NORTHING','HGT(m)
↪','sec_since']].iloc[i]

    distance_traveled = np.sqrt((start_pos['UTM_EASTING']-end_pos['UTM_EASTING
↪'])**2+(start_pos['UTM_NORTHING']-end_pos['UTM_NORTHING'])**2)
    time_between_obs = (end_pos['sec_since']-start_pos['sec_since'])/3600/24

    velocity_obs_upper=pd.DataFrame({'date1':[vel_date_start],'date2':[vel_date_end],
↪'start_easting':[start_pos['UTM_EASTING']],'start_northing':[start_pos['UTM_NORTHING
↪']],'end_easting':[end_pos['UTM_EASTING']],'end_northing':[end_pos['UTM_NORTHING']],
↪'distance_traveled (m)':[distance_traveled],'velocity (m/d)':[distance_traveled/
↪time_between_obs]},index=[2])
    velocity_obs_upper['label']='upper'

    #Arm
    for i in range(0,len(arm_pos_data_daily)):
        if arm_pos_data_daily['date'][i]==vel_date_start:
            start_pos= arm_pos_data_daily[['date','UTM_EASTING','UTM_NORTHING','HGT(m)
↪','sec_since']].iloc[i]
        if arm_pos_data_daily['date'][i]==vel_date_end:
            end_pos=arm_pos_data_daily[['date','UTM_EASTING','UTM_NORTHING','HGT(m)',
↪'sec_since']].iloc[i]

    distance_traveled = np.sqrt((start_pos['UTM_EASTING']-end_pos['UTM_EASTING
↪'])**2+(start_pos['UTM_NORTHING']-end_pos['UTM_NORTHING'])**2)
    time_between_obs = (end_pos['sec_since']-start_pos['sec_since'])/3600/24

    velocity_obs_arm=pd.DataFrame({'date1':[vel_date_start],'date2':[vel_date_end],
↪'start_easting':[start_pos['UTM_EASTING']],'start_northing':[start_pos['UTM_NORTHING
↪']],'end_easting':[end_pos['UTM_EASTING']],'end_northing':[end_pos['UTM_NORTHING']],
↪'distance_traveled':[distance_traveled],'velocity':[distance_traveled/time_between_
↪obs]},index=[3])
    velocity_obs_arm['label']='arm'

    # print(velocity_obs_lower)
    # print(velocity_obs_middle)
    # print(velocity_obs_upper)
    # print(velocity_obs_arm)

#     velocity_data_between_two_dates = pd.concat([velocity_obs_lower,velocity_obs_
↪middle,velocity_obs_upper],axis=0)
    velocity_data_between_two_dates = pd.concat([velocity_obs_lower,velocity_obs_
↪middle,velocity_obs_upper,velocity_obs_arm],axis=0)

    csv_name='Kaskawulsh_v2_'+vel_date_start.strftime('%Y-%m-%d')+'_to_'+vel_date_end.
↪strftime('%Y-%m-%d')+'_GPS.csv'
    velocity_data_between_two_dates.to_csv(csv_name)
```

CHAPTER

EIGHTEEN

EXTRACT VELOCITY MAP DATA AT GNSS LOCATIONS

This script samples velocity at GNSS locations and updates all `pt*` fields in `notebooks/results_2022.csv`.

To reproduce this workflow, make sure you have downloaded all necessary input files (velocity maps and static terrain geometries) from https://doi.org/10.17605/OSF.IO/HE7YR and have updated the `Vx` and `Vy` columns in `notebooks/results_2022.csv` with the downloaded file paths before starting the analysis.

```
from glaft.georaster import Raster
import rasterio
import pandas as pd
import geopandas as gpd
import numpy as np

df = pd.read_csv('../results_2022.csv', dtype=str)
# df
```

The cell below provides a sanity check for `glaft.georaster.Raster`'s `value_at_coords` method.

```
tmp = Raster(df.loc[0, 'Vx'])
a, b = tmp.value_at_coords(621306.41954208, 6738829.50233354, window=3, return_
↪window=True)
vx_grid = rasterio.open(df.loc[0, 'Vx'])
sample_gen_vx = vx_grid.sample([(621306.41954208, 6738829.50233354)])
vx_sampled = np.array([float(record) for record in sample_gen_vx])
```

Now let's start the analysis. If you download these files from https://doi.org/10.17605/OSF.IO/HE7YR, make sure to change the paths to the correct file locations on your local machine.

```
gps_files = ['/home/jovyan/Projects/PX_comparison/GPS/Kaskawulsh_2018-04-05_to_2018-
↪03-04_GPS',
             '/home/jovyan/Projects/PX_comparison/GPS/Kaskawulsh_2018-08-18_to_2018-
↪08-02_GPS',
             '/home/jovyan/Projects/PX_comparison/GPS/Kaskawulsh_2018-03-14_to_2018-
↪03-04_GPS',
             '/home/jovyan/Projects/PX_comparison/GPS/Kaskawulsh_2018-06-27_to_2018-
↪05-08_GPS']

datestrs = ['LS8-20180304-20180405', 'LS8-20180802-20180818', 'Sen2-20180304-20180314
↪', 'Sen2-20180508-20180627']
datenums = [32, 16, 10, 50]
```

Steps here:

1. Get and print the UTM coordinates of three GNSS statations for each scene pair.

2. Sample every velocity maps.
3. Create additional fields and calculate the difference between GNSS and feature tracked measurements.

```
for gps_file, datestr, datenum in zip(gps_files, datestrs, datenums):
    gps = pd.read_csv(gps_file)
    # Additional treatment for Sen2-20180508-20180627
    # many of the points here should be (nan, nan), (nan, nan) but nans does not work␣
↪with rio.sample
    if datestr == 'Sen2-20180508-20180627':
        gps.loc[1, 'end_easting'] = 610481.2868266493
        gps.loc[1, 'end_northing'] = 6737102.953712379
        gps.loc[2, 'end_easting'] = 601790.4387747
        gps.loc[2, 'end_northing'] = 6733753.77267354
        gps = gps.loc[0:2]
    gps = gpd.GeoDataFrame(gps, geometry=gpd.points_from_xy(gps['end_easting'], gps[
↪'end_northing']), crs='EPSG:32607')
    # This is beginning coordinates
    gps_xy = list(gps[['end_easting', 'end_northing']].to_records(index=False))
    print(datestr, gps_xy)

    gps['vx (m/d)']  = (gps['start_easting'] - gps['end_easting']) / datenum
    gps['vy (m/d)']  = (gps['start_northing'] - gps['end_northing']) / datenum

    df_s = df.loc[df['Date'] == datestr]
    for idx, row in df_s.iterrows():
        vx_grid = Raster(row.Vx)
        vy_grid = Raster(row.Vy)
        sampled = []
        for x, y in gps_xy:
            vx_avg, vx_3by3 = vx_grid.value_at_coords(x, y, window=3, return_
↪window=True)
            vy_avg, vy_3by3 = vy_grid.value_at_coords(x, y, window=3, return_
↪window=True)
            vx_3by3[vx_3by3 < -9998] = np.nan
            vy_3by3[vy_3by3 < -9998] = np.nan
            vx_3by3[vx_3by3 == 0.0] = np.nan    #Vmap
            vy_3by3[vy_3by3 == 0.0] = np.nan    #Vmap
            vx_nn = vx_3by3[0, 1, 1]      # nearest neighbor value
            vy_nn = vy_3by3[0, 1, 1]
            if np.any(~np.isnan(vx_3by3)):
                vx_avg = np.nanmean(vx_3by3)
            else:
                vx_avg = np.nan
            if np.any(~np.isnan(vy_3by3)):
                vy_avg = np.nanmean(vy_3by3)
            else:
                vy_avg = np.nan

            vx_nn -= float(df.loc[idx, 'SAV-peak-x'])
            vx_avg -= float(df.loc[idx, 'SAV-peak-x'])
            vy_nn -= float(df.loc[idx, 'SAV-peak-y'])
            vy_avg -= float(df.loc[idx, 'SAV-peak-y'])

            sampled.append([vx_nn, vx_avg, vy_nn, vy_avg])

        sampled = np.array(sampled)
        # print(row.Vx, float(df.loc[idx, 'SAV-peak-x']), float(df.loc[idx, 'SAV-peak-
↪y']), sampled)
```

(continues on next page)

(continued from previous page)

```
        df.loc[idx, 'pt0_vxdiff'] = sampled[0, 0] - gps.loc[0, 'vx (m/d)']
        df.loc[idx, 'pt0_vxavgdiff'] = sampled[0, 1] - gps.loc[0, 'vx (m/d)']
        df.loc[idx, 'pt0_vydiff'] = sampled[0, 2] - gps.loc[0, 'vy (m/d)']
        df.loc[idx, 'pt0_vyavgdiff'] = sampled[0, 3] - gps.loc[0, 'vy (m/d)']
        df.loc[idx, 'pt1_vxdiff'] = sampled[1, 0] - gps.loc[1, 'vx (m/d)']
        df.loc[idx, 'pt1_vxavgdiff'] = sampled[1, 1] - gps.loc[1, 'vx (m/d)']
        df.loc[idx, 'pt1_vydiff'] = sampled[1, 2] - gps.loc[1, 'vy (m/d)']
        df.loc[idx, 'pt1_vyavgdiff'] = sampled[1, 3] - gps.loc[1, 'vy (m/d)']
        df.loc[idx, 'pt2_vxdiff'] = sampled[2, 0] - gps.loc[2, 'vx (m/d)']
        df.loc[idx, 'pt2_vxavgdiff'] = sampled[2, 1] - gps.loc[2, 'vx (m/d)']
        df.loc[idx, 'pt2_vydiff'] = sampled[2, 2] - gps.loc[2, 'vy (m/d)']
        df.loc[idx, 'pt2_vyavgdiff'] = sampled[2, 3] - gps.loc[2, 'vy (m/d)']
```

```
LS8-20180304-20180405 [(621306.41954208, 6738829.50233354), (610435.5249175,␣
↪6737129.57698521), (601733.22946583, 6733710.66504834)]
LS8-20180802-20180818 [(621363.01607688, 6738895.12164604), (610506.52739125,␣
↪6737089.56006354), (601790.43877479, 6733753.77267354)]
Sen2-20180304-20180314 [(621306.41954208, 6738829.50233354), (610435.5249175,␣
↪6737129.57698521), (601733.22946583, 6733710.66504834)]
Sen2-20180508-20180627 [(621324.96198502, 6738852.60218059), (610481.28682665,␣
↪6737102.95371238), (601790.4387747, 6733753.77267354)]
```

You can comment/uncomment these lines to examine the data/results.

```
gps
# df
```

```
   Unnamed: 0       date1       date2  start_easting  start_northing  \
0           0  2018-06-27  2018-05-08  621348.115449    6.738880e+06
1           1  2018-06-27  2018-05-08  610481.286827    6.737103e+06
2           2  2018-06-27  2018-05-08            NaN             NaN

     end_easting  end_northing  distance_traveled (m)  velocity (m/d)  \
0  621324.961985  6.738853e+06              36.045525        0.720837
1  610481.286827  6.737103e+06                    NaN             NaN
2  601790.438775  6.733754e+06                    NaN             NaN

                       geometry  vx (m/d)  vy (m/d)
0  POINT (621324.962 6738852.602)  0.463069   0.55252
1  POINT (610481.287 6737102.954)  0.000000   0.00000
2  POINT (601790.439 6733753.773)       NaN       NaN
```

```
df.to_csv('../results_2022.csv', index=False)
```

CHAPTER NINETEEN

CALCULATE INVALID PIXEL PERCENTAGE

This script calculates invalid pixel percentage and updates the `Invalid-pixel-percent` field in `notebooks/results_2022.csv`.

To reproduce this workflow, make sure you have downloaded all necessary input files (velocity maps and static terrain geometries) from https://doi.org/10.17605/OSF.IO/HE7YR and have updated the `Vx` and `Vy` columns in `notebooks/results_2022.csv` with the downloaded file paths before starting the analysis.

```
import glaft
import pandas as pd
```

```
df = pd.read_csv('../results_2022.csv', dtype=str)
# df
```

```
for idx, row in df.iterrows():
    # print(row.Vx)
    if row.Software == 'Vmap':
        # print('yes')
        ## Vmap derived velocity maps have a NoData value of 0, which needs a special␣
↪attention.
        exp = glaft.Velocity(vxfile=row.Vx, vyfile=row.Vy, nodata=0)
    else:
        exp = glaft.Velocity(vxfile=row.Vx, vyfile=row.Vy)
    exp.cal_invalid_pixel_percent()
    df.loc[idx, 'Invalid-pixel-percent'] = exp.invalid_percent * 100
```

```
df.to_csv('../results_2022.csv', index=False)
# df
```

Part V

ITS_LIVE data processing scripts

CHAPTER

TWENTY

ITS_LIVE: CALCULATE METRIC 1

This notebook calculates δ_u and δ_v for all ITS_LIVE velocity maps.

To reproduce this workflow, make sure you have downloaded all necessary input files (velocity maps and static terrain geometries) from https://doi.org/10.17605/OSF.IO/HE7YR and have updated the `Vx` and `Vy` columns in `notebooks/results_ITSLIVE.csv` or `notebooks/manifest_ITSLIVE.csv` with the downloaded file paths before starting the analysis.

```
import glaft
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import netCDF4 as nc
```

```
# df = pd.read_csv('../manifest_ITSLIVE.csv', dtype=str)
df = pd.read_csv('../results_ITSLIVE.csv', dtype=str)
# df
```

```
# static area
in_shp = '/home/jovyan/Projects/PX_comparison/shapefiles/bedrock_V2_EPSG3413.shp'
```

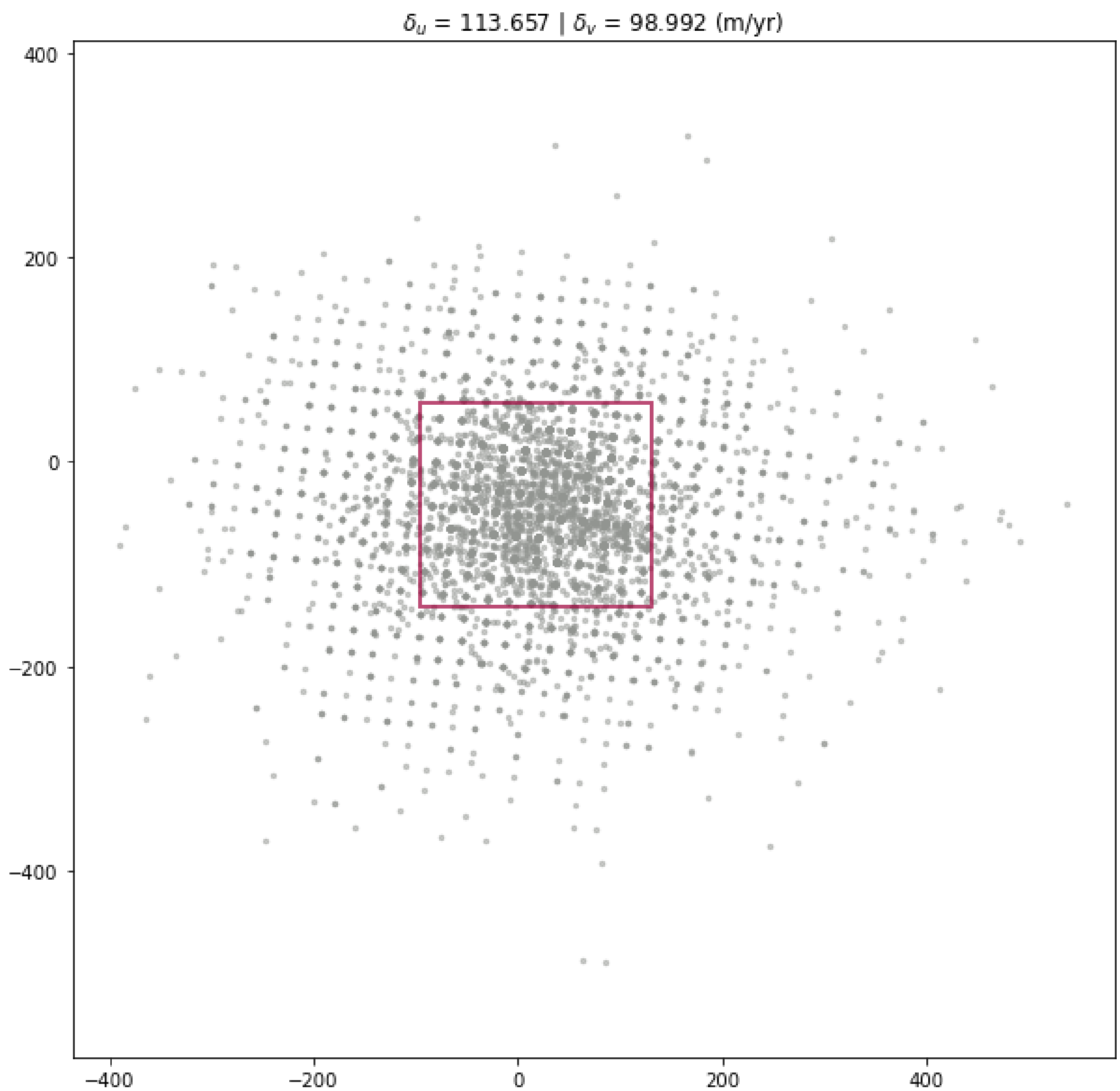
Locate the assigned error value in the metadata of the original NetCDF files:

```
for idx, row in df.iterrows():
    ncfname_processed = row.Vx[:-7]
    ncfname_list = ncfname_processed.split('/')
    ncfname_list.append(ncfname_list[-1])
    ncfname_list[-2] = 'raw_nc'
    ncfname = '/'.join(ncfname_list)
    with nc.Dataset(ncfname) as ds:
        vxd = ds['vx']
        xe = vxd.__dict__['error']
        # print(xe)
        vyd = ds['vy']
        ye = vyd.__dict__['error']
        # print(ye)
        df.loc[idx, 'Assigned-x-error'] = xe
        df.loc[idx, 'Assigned-y-error'] = ye

# df
```

Here's a demo for calculating Metric 1 for an ITS_LIVE velocity map:

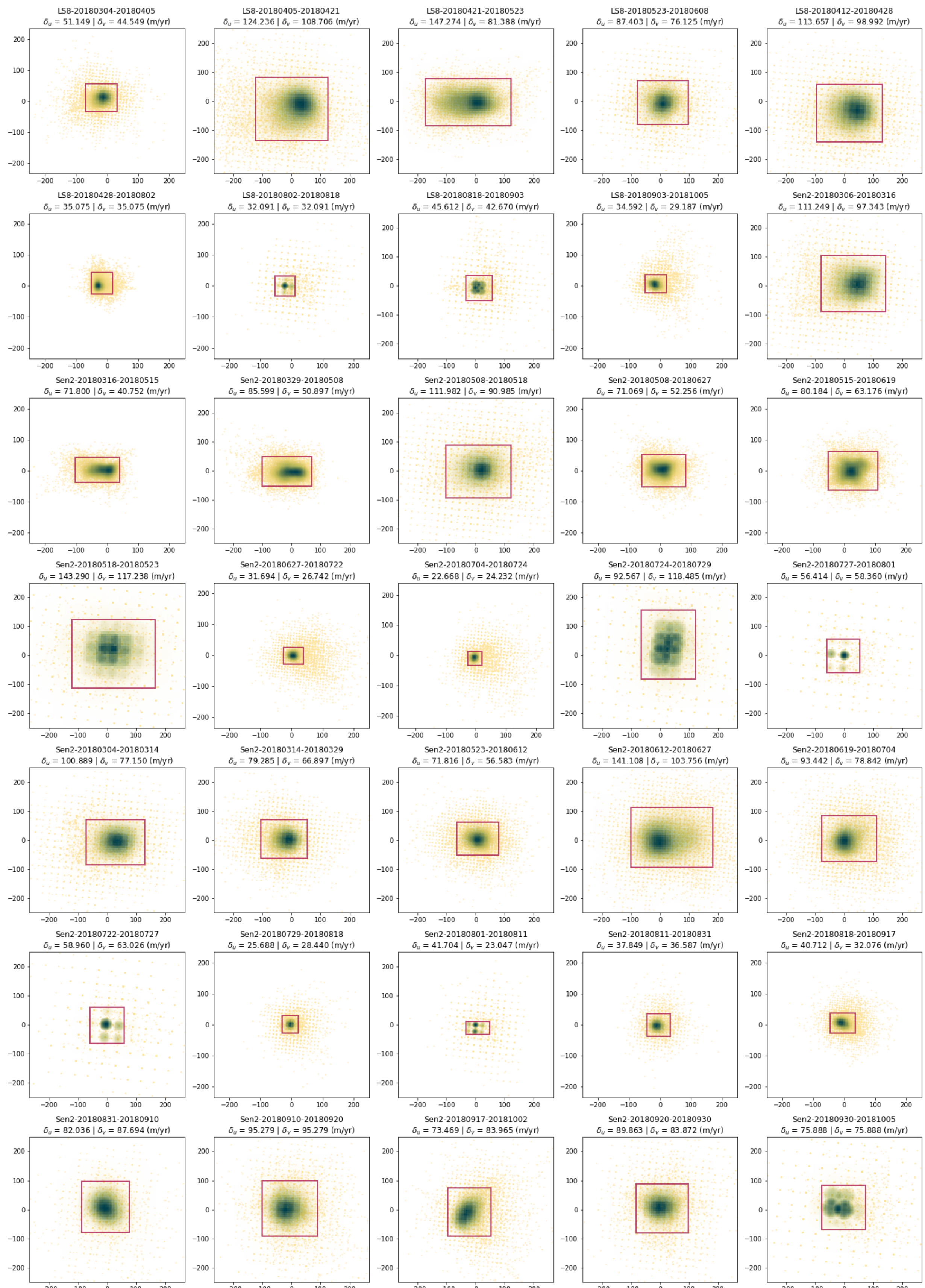
```
exp = glaft.Velocity(vxfile=df.loc[4, 'Vx'], vyfile=df.loc[4, 'Vy'],
                     static_area=in_shp, kde_gridsize=60, thres_sigma=2.0,
                     velocity_unit='m/yr')
exp.static_terrain_analysis(plot='full')
```

```
Running clip_static_area
Running calculate_xystd
Running calculate_bandwidth
Running calculate_kde
Running construct_crude_mesh
Running eval_crude_mesh
Running construct_fine_mesh
Running eval_fine_mesh
Running thresholding_fine_mesh
Running thresholding_metric
Running cal_outlier_percent
```



Now let's batch process all the maps:

```
fig, ax2 = plt.subplots(7, 5, figsize=(20, 28))
n = 0

for idx, row in df.iterrows():
    label = row.Label
    ax_sel = ax2[n // 5, n % 5]
    ax_sel.axis('equal')
    exp = glaft.Velocity(vxfile=row.Vx, vyfile=row.Vy, static_area=in_shp, kde_
↪gridsize=60, thres_sigma=2.0, velocity_unit='m/yr')
    exp.static_terrain_analysis(plot='zoomed', ax=ax_sel)
    ax_sel.set_xlim(-250, 250)
    ax_sel.set_ylim(-250, 250)
    titletext = ax_sel.get_title()
    titletext = label + '\n' + titletext
    ax_sel.set_title(titletext)

    df.loc[idx, 'SAV-uncertainty-x'] = exp.metric_static_terrain_x
    df.loc[idx, 'SAV-uncertainty-y'] = exp.metric_static_terrain_y
    df.loc[idx, 'SAV-peak-x'] = exp.kdepeak_x
    df.loc[idx, 'SAV-peak-y'] = exp.kdepeak_y
    df.loc[idx, 'SAV-outlier-percent'] = exp.outlier_percent * 100

    print('xstd: {}; xSAVuncer: {}'.format(np.std(exp.xy[0, :]), exp.metric_static_
↪terrain_x))
    print('ystd: {}; ySAVuncer: {}'.format(np.std(exp.xy[1, :]), exp.metric_static_
↪terrain_y))

    n += 1

plt.tight_layout()
fig.patch.set_facecolor('xkcd:white')
fig.savefig('figs/ITSLVE-SAV.png')
```


LS8-20180304-20180405
δu = 51.149 | δv = 44.549 (m/yr)
LS8-20180405-20180421
δu = 124.236 | δv = 108.706 (m/yr)
LS8-20180421-20180523
δu = 147.274 | δv = 81.388 (m/yr)
LS8-20180523-20180608
δu = 87.403 | δv = 76.125 (m/yr)
LS8-20180412-20180428
δu = 113.657 | δv = 98.992 (m/yr)
LS8-20180428-20180802
δu = 35.075 | δv = 35.075 (m/yr)
LS8-20180802-20180818
δu = 32.091 | δv = 32.091 (m/yr)
LS8-20180818-20180903
δu = 45.612 | δv = 42.670 (m/yr)
LS8-20180903-20181005
δu = 34.592 | δv = 29.187 (m/yr)
Sen2-20180306-20180316
δu = 111.249 | δv = 97.343 (m/yr)
Sen2-20180316-20180515
δu = 71.800 | δv = 40.752 (m/yr)
Sen2-20180329-20180508
δu = 85.599 | δv = 50.897 (m/yr)
Sen2-20180508-20180518
δu = 111.982 | δv = 90.985 (m/yr)
Sen2-20180508-20180627
δu = 71.069 | δv = 52.256 (m/yr)
Sen2-20180515-20180619
δu = 80.184 | δv = 63.176 (m/yr)
Sen2-20180518-20180523
δu = 143.290 | δv = 117.238 (m/yr)
Sen2-20180627-20180722
δu = 31.694 | δv = 26.742 (m/yr)
Sen2-20180704-20180724
δu = 22.668 | δv = 24.232 (m/yr)
Sen2-20180724-20180729
δu = 92.567 | δv = 118.485 (m/yr)
Sen2-20180727-20180801
δu = 56.414 | δv = 58.360 (m/yr)
Sen2-20180304-20180314
δu = 100.889 | δv = 77.150 (m/yr)
Sen2-20180314-20180329
δu = 79.285 | δv = 66.897 (m/yr)
Sen2-20180523-20180612
δu = 71.816 | δv = 56.583 (m/yr)
Sen2-20180612-20180627
δu = 141.108 | δv = 103.756 (m/yr)
Sen2-20180619-20180704
δu = 93.442 | δv = 78.842 (m/yr)
Sen2-20180722-20180727
δu = 58.960 | δv = 63.026 (m/yr)
Sen2-20180729-20180818
δu = 25.688 | δv = 28.440 (m/yr)
Sen2-20180801-20180811
δu = 41.704 | δv = 23.047 (m/yr)
Sen2-20180811-20180831
δu = 37.849 | δv = 36.587 (m/yr)
Sen2-20180818-20180917
δu = 40.712 | δv = 32.076 (m/yr)
Sen2-20180831-20180910
δu = 82.036 | δv = 87.694 (m/yr)
Sen2-20180910-20180920
δu = 95.279 | δv = 95.279 (m/yr)
Sen2-20180917-20181002
δu = 73.469 | δv = 83.965 (m/yr)
Sen2-20180920-20180930
δu = 89.863 | δv = 83.872 (m/yr)
Sen2-20180930-20181005
δu = 75.888 | δv = 75.888 (m/yr)

```
df.to_csv('../results_ITSLIVE.csv', index=False)
# df
```

CHAPTER

TWENTYONE

ITS_LIVE: CALCULATE METRIC 2

This notebook calculates $\delta_{x'y'}$ for all ITS_LIVE velocity maps.

To reproduce this workflow, make sure you have downloaded all necessary input files (velocity maps and static terrain geometries) from https://doi.org/10.17605/OSF.IO/HE7YR and have updated the `Vx` and `Vy` columns in `notebooks/results_ITSLIVE.csv` or `notebooks/manifest_ITSLIVE.csv` with the downloaded file paths before starting the analysis.

```
import glaft
import matplotlib.pyplot as plt
import pandas as pd
```

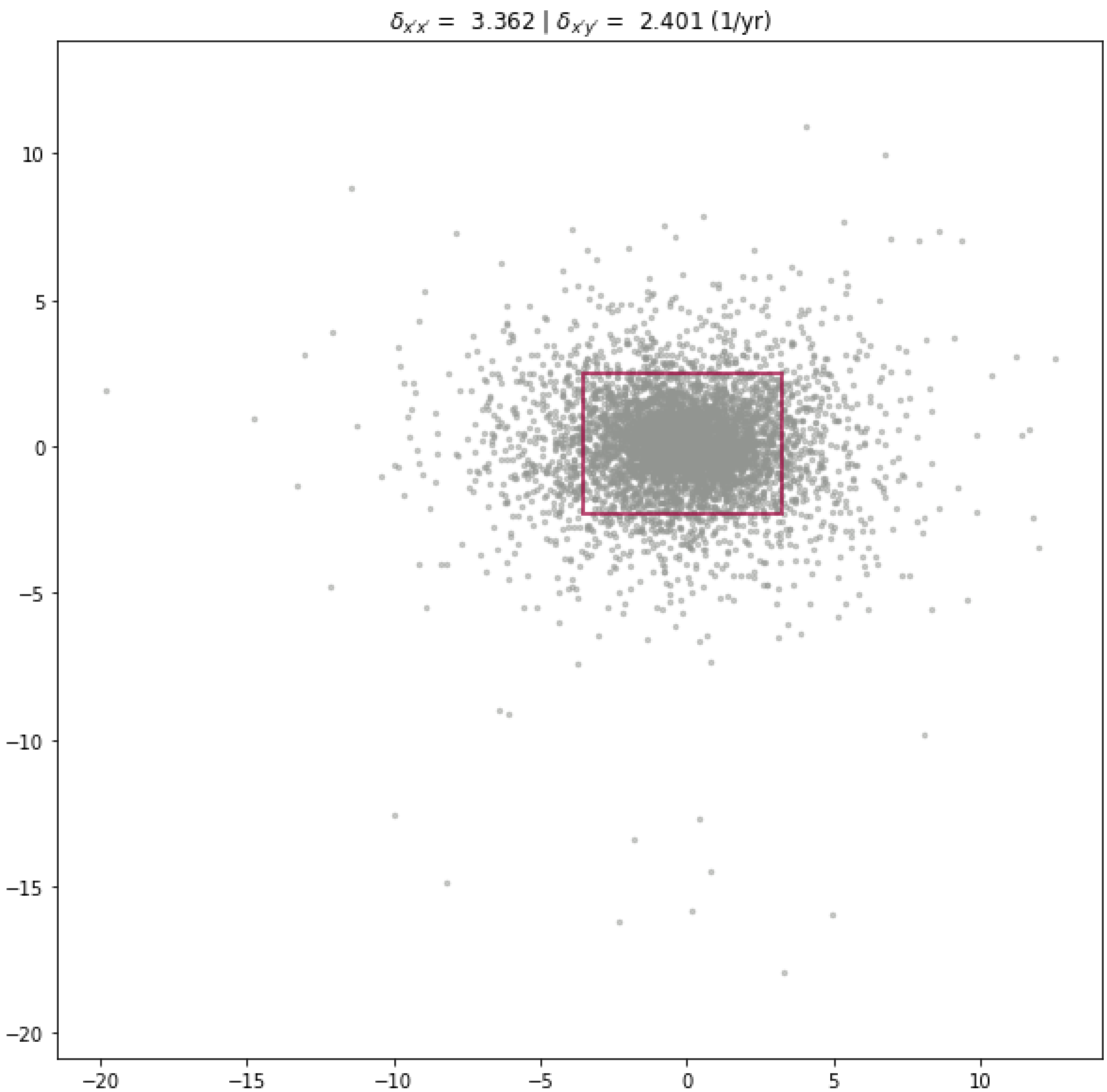
```
# df = pd.read_csv('../manifest.csv', dtype=str)
df = pd.read_csv('../results_ITSLIVE.csv', dtype=str)
# df
```

```
# flow area
in_shp = '/home/jovyan/Projects/PX_comparison/Bedrock_shp/glacier_V1_Kaskawulsh_s_
↪inwardBuffer600m_EPSG3413.shp'
```

Here's a demo for calculating Metric 2 for an ITS_LIVE velocity map:

```
exp = glaft.Velocity(vxfile=df.loc[4, 'Vx'], vyfile=df.loc[4, 'Vy'],
                     on_ice_area=in_shp, kde_gridsize=60, thres_sigma=2.0,
                     velocity_unit='m/yr')
exp.longitudinal_shear_analysis(plot='full')
```

```
Running clip_on_ice_area
Running get_grid_spacing
Running calculate_flow_theta
Running calculate_strain_rate
Running prep_strain_rate_kde
Running calculate_xystd
Running calculate_bandwidth
Running calculate_kde
Running construct_crude_mesh
Running eval_crude_mesh
Running construct_fine_mesh
Running eval_fine_mesh
Running thresholding_fine_mesh
Running thresholding_metric
Running cal_outlier_percent
```



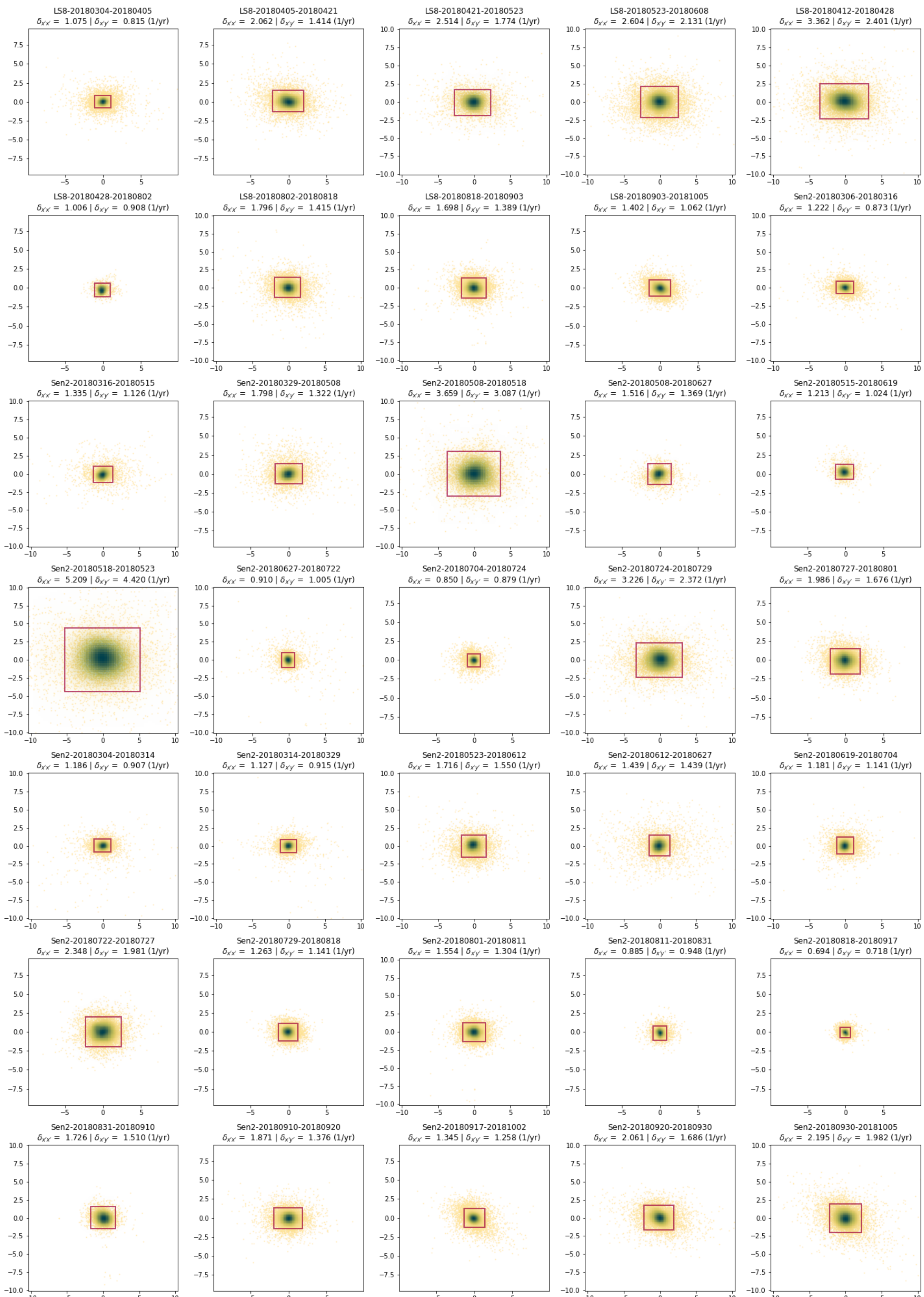
Now let's batch process all the maps:

```
fig, ax2 = plt.subplots(7, 5, figsize=(20, 28))
n = 0

for idx, row in df.iterrows():
    label = row.Label
    ax_sel = ax2[n // 5, n % 5]
    ax_sel.axis('equal')
    exp = glaft.Velocity(vxfile=row.Vx, vyfile=row.Vy, on_ice_area=in_shp, kde_
↪gridsize=60, thres_sigma=2.0, velocity_unit='m/yr')
    exp.longitudinal_shear_analysis(plot='zoomed', ax=ax_sel)
    ax_sel.set_xlim(-10, 10)
    ax_sel.set_ylim(-10, 10)
    titletext = ax_sel.get_title()
    titletext = label + '\n' + titletext
    ax_sel.set_title(titletext)
```

(continues on next page)

(continued from previous page)

```
    df.loc[idx, 'LSR-uncertainty-nm'] = exp.metric_alongflow_normal
    df.loc[idx, 'LSR-uncertainty-sh'] = exp.metric_alongflow_shear


    n += 1

plt.tight_layout()
fig.patch.set_facecolor('xkcd:white')
fig.savefig('figs/ITSLVE-LSR.png')
```


LS8-20180304-20180405
δxx' = 1.075 | δxy' = 0.815 (1/yr)
LS8-20180405-20180421
δxx' = 2.062 | δxy' = 1.414 (1/yr)
LS8-20180421-20180523
δxx' = 2.514 | δxy' = 1.774 (1/yr)
LS8-20180523-20180608
δxx' = 2.604 | δxy' = 2.131 (1/yr)
LS8-20180412-20180428
δxx' = 3.362 | δxy' = 2.401 (1/yr)
LS8-20180428-20180802
δxx' = 1.006 | δxy' = 0.908 (1/yr)
LS8-20180802-20180818
δxx' = 1.796 | δxy' = 1.415 (1/yr)
LS8-20180818-20180903
δxx' = 1.698 | δxy' = 1.389 (1/yr)
LS8-20180903-20181005
δxx' = 1.402 | δxy' = 1.062 (1/yr)
Sen2-20180306-20180316
δxx' = 1.222 | δxy' = 0.873 (1/yr)
Sen2-20180316-20180515
δxx' = 1.335 | δxy' = 1.126 (1/yr)
Sen2-20180329-20180508
δxx' = 1.798 | δxy' = 1.322 (1/yr)
Sen2-20180508-20180518
δxx' = 3.659 | δxy' = 3.087 (1/yr)
Sen2-20180508-20180627
δxx' = 1.516 | δxy' = 1.369 (1/yr)
Sen2-20180515-20180619
δxx' = 1.213 | δxy' = 1.024 (1/yr)
Sen2-20180518-20180523
δxx' = 5.209 | δxy' = 4.420 (1/yr)
Sen2-20180627-20180722
δxx' = 0.910 | δxy' = 1.005 (1/yr)
Sen2-20180704-20180724
δxx' = 0.850 | δxy' = 0.879 (1/yr)
Sen2-20180724-20180729
δxx' = 3.226 | δxy' = 2.372 (1/yr)
Sen2-20180727-20180801
δxx' = 1.986 | δxy' = 1.676 (1/yr)
Sen2-20180304-20180314
δxx' = 1.186 | δxy' = 0.907 (1/yr)
Sen2-20180314-20180329
δxx' = 1.127 | δxy' = 0.915 (1/yr)
Sen2-20180523-20180612
δxx' = 1.716 | δxy' = 1.550 (1/yr)
Sen2-20180612-20180627
δxx' = 1.439 | δxy' = 1.439 (1/yr)
Sen2-20180619-20180704
δxx' = 1.181 | δxy' = 1.141 (1/yr)
Sen2-20180722-20180727
δxx' = 2.348 | δxy' = 1.981 (1/yr)
Sen2-20180729-20180818
δxx' = 1.263 | δxy' = 1.141 (1/yr)
Sen2-20180801-20180811
δxx' = 1.554 | δxy' = 1.304 (1/yr)
Sen2-20180811-20180831
δxx' = 0.885 | δxy' = 0.948 (1/yr)
Sen2-20180818-20180917
δxx' = 0.694 | δxy' = 0.718 (1/yr)
Sen2-20180831-20180910
δxx' = 1.726 | δxy' = 1.510 (1/yr)
Sen2-20180910-20180920
δxx' = 1.871 | δxy' = 1.376 (1/yr)
Sen2-20180917-20181002
δxx' = 1.345 | δxy' = 1.258 (1/yr)
Sen2-20180920-20180930
δxx' = 2.061 | δxy' = 1.686 (1/yr)
Sen2-20180930-20181005
δxx' = 2.195 | δxy' = 1.982 (1/yr)

```
df.to_csv('../results_ITSLIVE.csv', index=False)
# df
```

CHAPTER

TWENTYTWO

ITS_LIVE: EXTRACT VELOCITY MAP DATA AT GNSS LOCATIONS

This script samples velocity at GNSS locations and updates all `pt*` fields in `notebooks/results_ITSLIVE.csv`.

To reproduce this workflow, make sure you have downloaded all necessary input files (velocity maps and static terrain geometries) from https://doi.org/10.17605/OSF.IO/HE7YR and have updated the `Vx` and `Vy` columns in `notebooks/results_ITSLIVE.csv` with the downloaded file paths before starting the analysis.

```
from glaft.georaster import Raster
import rasterio
import pandas as pd
import geopandas as gpd
import numpy as np
from datetime import datetime
from pyproj import Transformer

df = pd.read_csv('../results_ITSLIVE.csv', dtype=str)
# df
```

```
# All GNSS coordinates are in EPSG 32607 and have to be reprojected to EPSG3413␣
↪before sampling the geotiffs.
transformer = Transformer.from_crs("epsg:32607", "epsg:3413")
```

The cell below is the main procedure.

```
GPS_root = '/home/jovyan/Projects/PX_comparison/GPS/'

for idx, row in df.iterrows():
    startdate = datetime.strptime(row['Start date'], '%Y%m%d')
    enddate = datetime.strptime(row['End date'], '%Y%m%d')
    timedel = enddate - startdate
    duration = timedel.days / 365      # in yrs
    startdate_gpsstr = startdate.strftime('%Y-%m-%d')
    enddate_gpsstr = enddate.strftime('%Y-%m-%d')
    gps_file = GPS_root + 'Kaskawulsh_{}_to_{}_GPS'.format(enddate_gpsstr, startdate_
↪gpsstr)

    gps = pd.read_csv(gps_file)
    ######## Prep corrdinates in EPSG3413
    for idx2, row2 in gps.iterrows():
        x32607 = row2.start_easting
        y32607 = row2.start_northing
        x3413, y3413 = transformer.transform(x32607, y32607)
        gps.loc[idx2, 'start_easting_3413'] = x3413
        gps.loc[idx2, 'start_northing_3413'] = y3413
```

(continues on next page)

(continued from previous page)

```
        x32607 = row2.end_easting
        y32607 = row2.end_northing
        x3413, y3413 = transformer.transform(x32607, y32607)
        gps.loc[idx2, 'end_easting_3413'] = x3413
        gps.loc[idx2, 'end_northing_3413'] = y3413
    #########
    # This is beginning coordinates
    gps = gpd.GeoDataFrame(gps, geometry=gpd.points_from_xy(gps['end_easting_3413'],␣
↪gps['end_northing_3413']), crs='EPSG:3413')
    gps_xy = list(gps[['end_easting_3413', 'end_northing_3413']].to_
↪records(index=False))

    gps['vx (m/yr)']  = (gps['start_easting_3413'] - gps['end_easting_3413']) /␣
↪duration
    gps['vy (m/yr)']  = (gps['start_northing_3413'] - gps['end_northing_3413']) /␣
↪duration
    gps['v (m/yr)']  = np.abs(gps['velocity (m/d)'] * 365)

    vx_grid = Raster(row.Vx)
    vy_grid = Raster(row.Vy)
    v_grid = Raster(row.Vx.replace('vx', 'v'))
    verr_grid = Raster(row.Vx.replace('vx', 'v_error'))
    sampled = []
    sampled2 = []
    for x, y in gps_xy:
        # print(gps_file, x, y)
        if np.isnan(x) or np.isnan(y) or np.isinf(x) or np.isinf(y):
            sampled.append([np.nan, np.nan, np.nan, np.nan])
            sampled2.append([np.nan, np.nan])
        else:
            vx_avg, vx_3by3 = vx_grid.value_at_coords(x, y, window=3, return_
↪window=True)
            vy_avg, vy_3by3 = vy_grid.value_at_coords(x, y, window=3, return_
↪window=True)
            vx_3by3[vx_3by3 < -9998] = np.nan   # ITSLIVE nodata = -32767
            vy_3by3[vy_3by3 < -9998] = np.nan   # ITSLIVE nodata = -32767
            vx_nn = vx_3by3[0, 1, 1]      # nearest neighbor value
            vy_nn = vy_3by3[0, 1, 1]
            if np.any(~np.isnan(vx_3by3)):
                vx_avg = np.nanmean(vx_3by3)
            else:
                vx_avg = np.nan
            if np.any(~np.isnan(vy_3by3)):
                vy_avg = np.nanmean(vy_3by3)
            else:
                vy_avg = np.nan

            sampled.append([vx_nn, vx_avg, vy_nn, vy_avg])

            v_avg, v_3by3 = v_grid.value_at_coords(x, y, window=3, return_window=True)
            v_3by3[v_3by3 < -9998] = np.nan   # ITSLIVE nodata = -32767
            if np.any(~np.isnan(v_3by3)):
                v_avg = np.nanmean(v_3by3)
            else:
                v_avg = np.nan
```

(continues on next page)

(continued from previous page)

```
            verr_avg, verr_3by3 = verr_grid.value_at_coords(x, y, window=3, return_
↪window=True)
            verr_3by3[verr_3by3 < -9998] = np.nan    # ITSLIVE nodata = -32767
            if np.any(~np.isnan(verr_3by3)):
                verr_avg = np.nanmean(verr_3by3)
            else:
                verr_avg = np.nan

            sampled2.append([v_avg, verr_avg])


    sampled = np.array(sampled)
    sampled2 = np.array(sampled2)
    # print(sampled)
    # print(row.Vx, float(df.loc[idx, 'SAV-peak-x']), float(df.loc[idx, 'SAV-peak-y
↪']), sampled)

    df.loc[idx, 'pt0_vxavg'] = sampled[0, 1]
    df.loc[idx, 'pt0_vxgps'] = np.abs(gps.loc[0, 'vx (m/yr)'])
    df.loc[idx, 'pt0_vyavg'] = sampled[0, 3]
    df.loc[idx, 'pt0_vygps'] = np.abs(gps.loc[0, 'vy (m/yr)'])
    df.loc[idx, 'pt1_vxavg'] = sampled[1, 1]
    df.loc[idx, 'pt1_vxgps'] = np.abs(gps.loc[1, 'vx (m/yr)'])
    df.loc[idx, 'pt1_vyavg'] = sampled[1, 3]
    df.loc[idx, 'pt1_vygps'] = np.abs(gps.loc[1, 'vy (m/yr)'])
    df.loc[idx, 'pt2_vxavg'] = sampled[2, 1]
    df.loc[idx, 'pt2_vxgps'] = np.abs(gps.loc[2, 'vx (m/yr)'])
    df.loc[idx, 'pt2_vyavg'] = sampled[2, 3]
    df.loc[idx, 'pt2_vygps'] = np.abs(gps.loc[2, 'vy (m/yr)'])

    df.loc[idx, 'pt0_vxdiff'] = sampled[0, 0] - gps.loc[0, 'vx (m/yr)']
    df.loc[idx, 'pt0_vxavgdiff'] = sampled[0, 1] - gps.loc[0, 'vx (m/yr)']
    df.loc[idx, 'pt0_vydiff'] = sampled[0, 2] - gps.loc[0, 'vy (m/yr)']
    df.loc[idx, 'pt0_vyavgdiff'] = sampled[0, 3] - gps.loc[0, 'vy (m/yr)']
    df.loc[idx, 'pt1_vxdiff'] = sampled[1, 0] - gps.loc[1, 'vx (m/yr)']
    df.loc[idx, 'pt1_vxavgdiff'] = sampled[1, 1] - gps.loc[1, 'vx (m/yr)']
    df.loc[idx, 'pt1_vydiff'] = sampled[1, 2] - gps.loc[1, 'vy (m/yr)']
    df.loc[idx, 'pt1_vyavgdiff'] = sampled[1, 3] - gps.loc[1, 'vy (m/yr)']
    df.loc[idx, 'pt2_vxdiff'] = sampled[2, 0] - gps.loc[2, 'vx (m/yr)']
    df.loc[idx, 'pt2_vxavgdiff'] = sampled[2, 1] - gps.loc[2, 'vx (m/yr)']
    df.loc[idx, 'pt2_vydiff'] = sampled[2, 2] - gps.loc[2, 'vy (m/yr)']
    df.loc[idx, 'pt2_vyavgdiff'] = sampled[2, 3] - gps.loc[2, 'vy (m/yr)']

    df.loc[idx, 'pt0_vavg'] = sampled2[0, 0]
    df.loc[idx, 'pt0_vgps'] = gps.loc[0, 'v (m/yr)']
    df.loc[idx, 'pt0_vdiff'] = sampled2[0, 0] - gps.loc[0, 'v (m/yr)']
    df.loc[idx, 'pt0_verr'] = sampled2[0, 1]
    df.loc[idx, 'pt1_vavg'] = sampled2[1, 0]
    df.loc[idx, 'pt1_vgps'] = gps.loc[1, 'v (m/yr)']
    df.loc[idx, 'pt1_vdiff'] = sampled2[1, 0] - gps.loc[1, 'v (m/yr)']
    df.loc[idx, 'pt1_verr'] = sampled2[1, 1]
    df.loc[idx, 'pt2_vavg'] = sampled2[2, 0]
    df.loc[idx, 'pt2_vgps'] = gps.loc[2, 'v (m/yr)']
    df.loc[idx, 'pt2_vdiff'] = sampled2[2, 0] - gps.loc[2, 'v (m/yr)']
    df.loc[idx, 'pt2_verr'] = sampled2[2, 1]
```

You can comment/uncomment these lines to examine the data/results.

```
gps
# gps_xy
# df
```

```
   Unnamed: 0       date1       date2  start_easting  start_northing  \
0           0  2018-10-05  2018-09-30  621383.084841    6.738920e+06
1           1  2018-10-05  2018-09-30  610531.630118    6.737073e+06
2           2  2018-10-05  2018-09-30  601810.429736    6.733774e+06

     end_easting  end_northing  distance_traveled (m)  velocity (m/d)  \
0  621381.738315  6.738918e+06               2.015470       -0.403094
1  610530.195434  6.737074e+06               1.703303       -0.340661
2  601809.124834  6.733773e+06               1.881326       -0.376265

   start_easting_3413  start_northing_3413  end_easting_3413  \
0       -3.227459e+06        212767.842312     -3.227460e+06
1       -3.228240e+06        224145.144740     -3.228239e+06
2       -3.230736e+06        233478.821992     -3.230737e+06

   end_northing_3413                          geometry   vx (m/yr)   vy (m/yr)  \
0      212769.385111  POINT (-3227460.401 212769.385)  102.707969 -112.624285
1      224146.528571  POINT (-3228239.026 224146.529)  -79.947760 -101.019600
2      233480.308331  POINT (-3230737.265 233480.308)   92.088135 -108.502758

     v (m/yr)
0  147.129311
1  124.341145
2  137.336804
```

```
df = df.replace(-np.inf, np.nan)
df = df.replace(np.inf, np.nan)
df.to_csv('../results_ITSLIVE.csv', index=False)
```