Gftt: an open-source tool for evaluating remotely sensed glacier velocity products

Contents

Whyjay Zheng[1], Shashank Bhushan[2], Maximillian S Van Wyk de Vries[3], William Hardy Kochtitzky[4], and David E Shean[2]

[1]University of California, Berkeley, Statistics, Berkeley, CA, United States

[2]University of Washington, Civil & Environmental Engineering, Seattle, WA, United States

[3]University of Minnesota Twin Cities, Department of Earth and Environmental Sciences and Saint Anthony Falls Laboratory, Minneapolis, United States

[4]University of Ottawa, Department of Geography, Environment and Geomatics, Ottawa, Canada

Corresponding author and email: Whyjay Zheng (whyjz@berkeley.edu)

Abstract

Glacier velocity reflects the dynamics of ice flow, and its change over time serves a key role in predicting the future sea-level rise. Glacier feature tracking (also known as offset tracking or pixel tracking) is one of the most widely-used approaches for mapping glacier velocity using remote sensing data. However, running this workflow relies on multiple empirical parameter choices such as correlation kernel selection, image filter, and template size. As each target glacier area has different data availability, surface feature density, and ice flow width, there is no one-size-fits-all parameter set for glacier feature tracking.

Finding an ideal parameter set for a given glacier requires quantitative and objective metrics to determine the quality of resulting velocity maps. The objective of our Glacier feature tracking test (gftt) project is both to devise a set of widely applicable metrics and to build a Python-based tool for calculating them. These metrics can be thus used for comparing the performance of different tracking parameters. We use Kaskawulsh glacier, Canada, as a test case to compare the velocity mapping results using Landsat 8 and Sentinel-2 images, various software packages (including Auto-RIFT, CARST, GIV, and vmap), and a range of input parameters. To begin with, we calculate random error over stable terrain, a metric that has been used for evaluating the uncertainty of the velocity products. We develop two other workflows for exploring new metrics and validating existing metrics, including the test with synthetic pixel offsets and the comparison with GNSS records. These existing and new metrics, calculated through the gftt software, will help determine optimal parameter sets for feature tracking of Kaskawulsh glacier and any other glacier around the world.

This work is supported by the NSF Earth Cube Program under awards 1928406, 1928374.

Plain language summary

There are many ways to measure how quickly glaciers move including using GPS and satellites. Here, we compare a variety of methods to measure glacier motion to determine which method(s) perform the best in mapping glacier movement. We develop a software package called “gftt” to help execute our work and share our methods with the scientific community.

How to cite

doi:10.1002/essoar.10509355.1

By Whyjay Zheng, Shashank Bhushan, Max Van Wyk de Vries, Will Kochtitzky, David Shean

Metric #1: Off-ice errors

Contents

Assume that the off-ice area has a ground truth of zero horizontal motion ($V = V_x = V_y = 0$), then we can calculate the difference between the feature tracking results and the ground truth (i.e. zero) over the off-ice area on an image, which could serve as a metric to evaluate the overall uncertainty of velocity values.

Requirements and Limits

- There should be off-ice area in the source images. Works for mountain glaciers and outlet glaciers, but for ice sheet feature tracking it may be hard or impossible to find such the area.
- Even though there are off-ice areas, this method may still fail if the off-ice pixels are decorrelated over time. A common reason for that is surface melt in the spring and snow accumulation in the fall.

Procedure

1. Prepare the feature tracking results as Geotiff files, ideally V_x as one file and V_y as the other. The unit for pixel value is flexible, from pixels, pixels/day, pixels/year, to meters, meters/day, to meters/year. **In this notebook, all of the feature tracking data use meters/day as the unit of the pixel value.**
2. Locate off-ice area and make an ESRI shapefile (polygon format) for that. Future development includes the inverse selection of an RGI glacier outline as the off-ice area as well.
3. Use `gftt.off_ice_errors` and `gftt.plot_off_ice_errors` to calculate, analyze, and visualize the off-ice errors.

Analysis

1. Basic information, importing modules

- Test area: **Kaskawulsh Glacier**
- Test packages: **CARST, Vmap, GIV, autoRIFT**
- Test pair: **Landsat 8 20180304-20180405**

```
# Developer's setting
%load_ext autoreload
%autoreload 2
```

```
import gftt
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.patches as patches
from matplotlib import colors, cm
from matplotlib.colors import ListedColormap
import rasterio
from rasterio.plot import show
import numpy as np
import pandas as pd
import geopandas as gpd
```

2. Load the feature tracking data list and off-ice area shapefile

Each row is a feature tracking test using a specific parameter combination described in each field.

```
df = pd.read_csv('../manifest_agu21.csv', dtype=str)
df
```

	Date	Chip (px)	Resolution (px)	Prefilter	Subpixel	Software	
0	Sen2-20180304-20180314	48	12	Gau	16-node oversampling	CARST	/home/jovyan/Project
1	Sen2-20180304-20180314	48	12	NAOF	16-node oversampling	CARST	/home/jovyan/Project
2	Sen2-20180304-20180314	48	12	None	16-node oversampling	CARST	/home/jovyan/Project
3	Sen2-20180304-20180314	48	1	Gau	16-node oversampling	CARST	/home/jovyan/Project
4	Sen2-20180304-20180314	48	1	NAOF	16-node oversampling	CARST	/home/jovyan/Project
...	...	...	...	...	...	...	
167	Sen2-20180508-20180627	64	8	Gau	pyrUP	autoRIFT	/home/jovyan/Projec
168	Sen2-20180508-20180627	32	4	NAOF	pyrUP	autoRIFT	/home/jovyan/Projec
169	Sen2-20180508-20180627	32	8	NAOF	pyrUP	autoRIFT	/home/jovyan/Projec
170	Sen2-20180508-20180627	64	4	NAOF	pyrUP	autoRIFT	/home/jovyan/Projec
171	Sen2-20180508-20180627	64	8	NAOF	pyrUP	autoRIFT	/home/jovyan/Projec

172 rows × 8 columns

Load the off-ice area shapefile.

```
# off-ice area
in_shp = '../../data/shapefiles/bedrock_V1.shp'
off_ice = gpd.read_file(in_shp)
# in_shp2 =
'/home/jovyan/Projects/PX_comparison/Bedrock_shp/Kaskawulsh_RGI60_EPSG32607.shp'
# off_ice2 = gpd.read_file(in_shp2)
```

3. Demo for a single test

3.1 Demo 1

```
## Example results
demo = df.loc[30]
print(demo)
```

```
Date                                                LS8-20180304-20180405
Chip (px)                                                              64
Resolution (px)                                                         4
Prefilter                                                             Gau
Subpixel                                            16-node oversampling
Software                                                            CARST
Vx                 /home/jovyan/Projects/PX_comparison/PX/CARST/2...
Vy                 /home/jovyan/Projects/PX_comparison/PX/CARST/2...
Name: 30, dtype: object
```

Now make a special `viridis` colormap designed for Kaskawulsh:

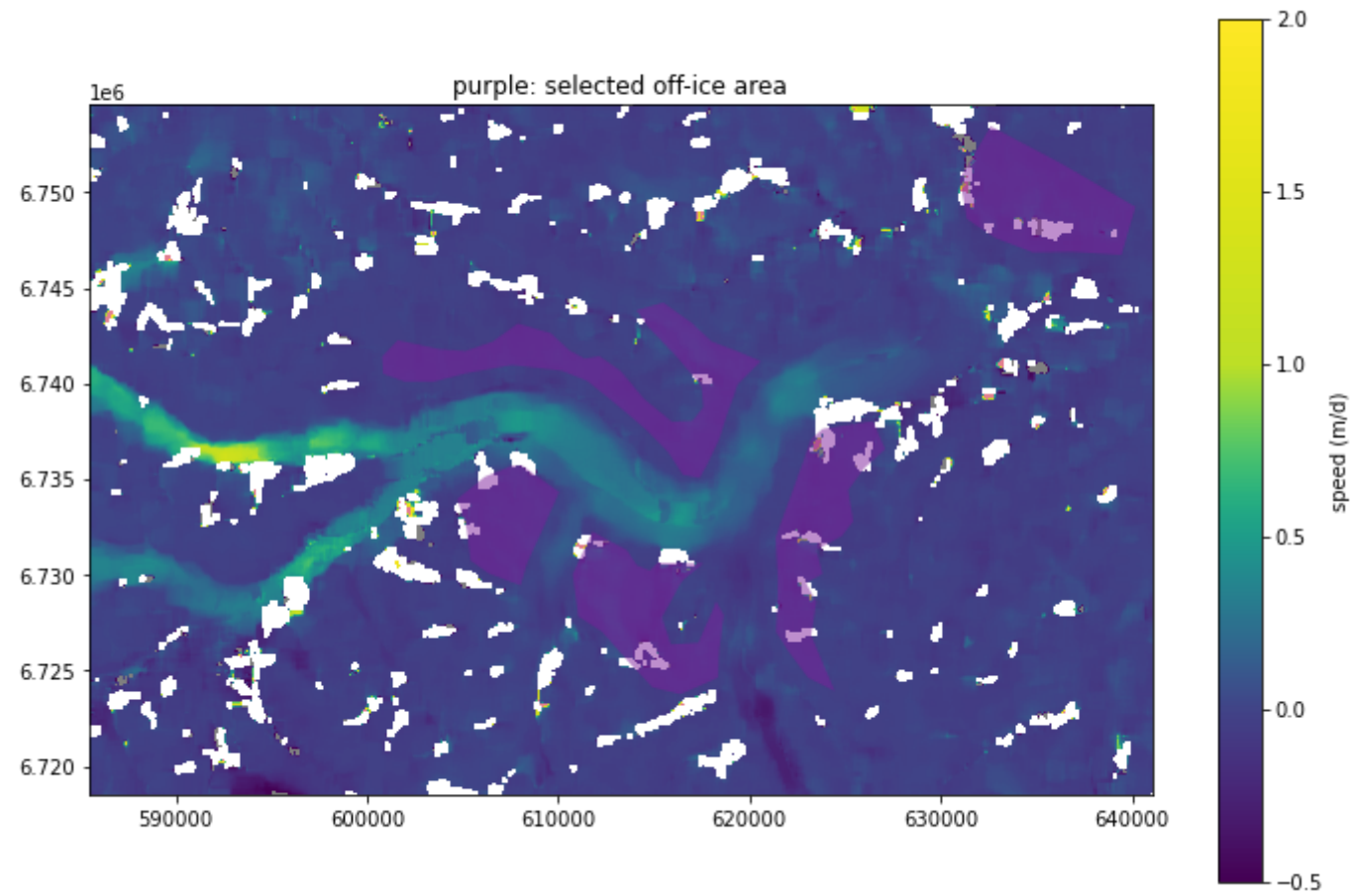
```
viridis = cm.get_cmap('viridis', 256)
newcolors1 = viridis(np.linspace(0, 0.2, 128))
newcolors2 = viridis(np.linspace(0.2, 0.9, 256))
newcolors3 = viridis(np.linspace(0.9, 1, 256))
newcolors = np.vstack((newcolors1, newcolors2, newcolors3))
cmap = ListedColormap(newcolors)
cmap.set_over(np.array([202,0,32,128]) / 255)
cmap.set_under(np.array([0,0,0,128]) / 255)
cmap
```

from_list

☐ under bad ☐ over ☐

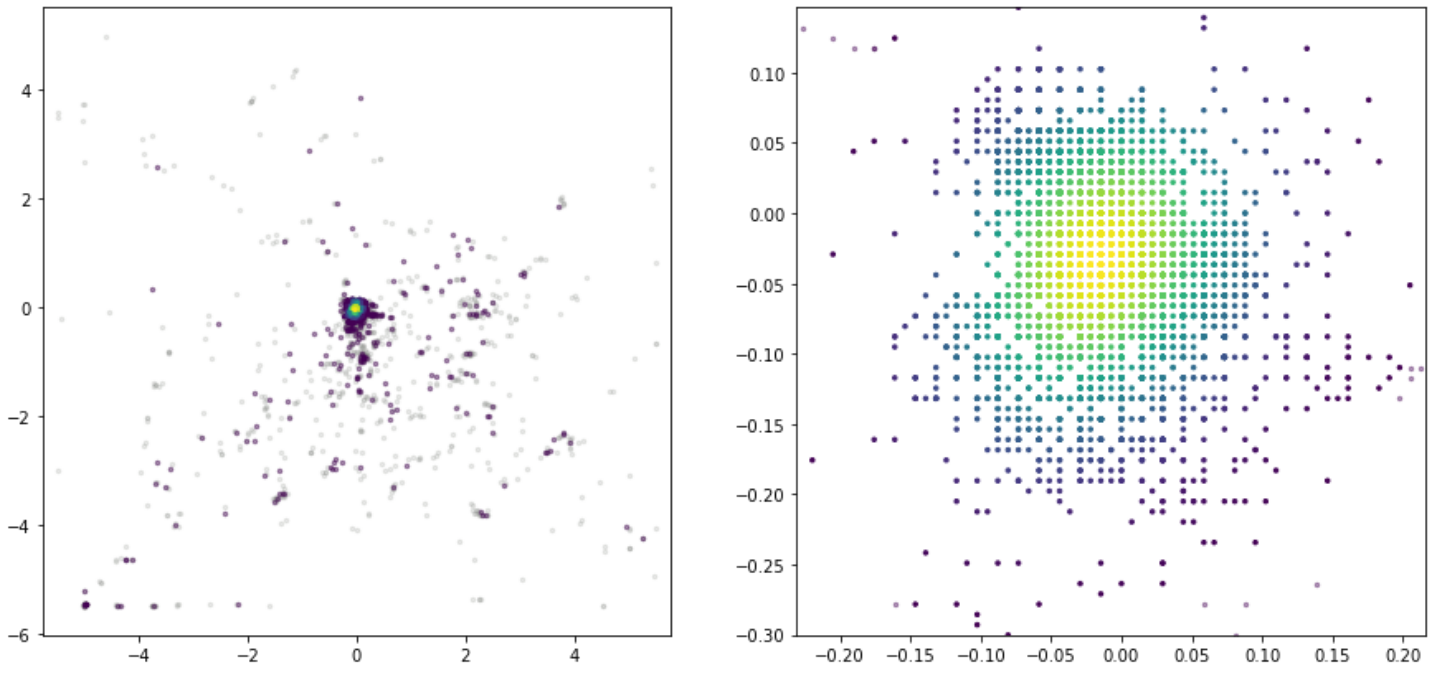
The velocity x component looks like this:

```
fig, ax0 = plt.subplots(1, 1, figsize=(12, 8))

# with rasterio.open(demo.Vx.replace('vx', 'mag')) as v_src:
#     v = v_src.read(1)
#     show(v_src, ax=ax0, vmin=0, vmax=2, cmap=cmap)

with rasterio.open(demo.Vx) as v_src:
    v = v_src.read(1)
    show(v_src, ax=ax0, vmin=-0.5, vmax=2, cmap=cmap)

cbar =  fig.colorbar(cm.ScalarMappable(norm=colors.Normalize(vmin=-0.5, vmax=2),
cmap=cmap), ax=ax0)
cbar.set_label('speed (m/d)')
off_ice.plot(ax=ax0, alpha=0.5, facecolor='xkcd:purple')
ax0.set_title('purple: selected off-ice area');
```



Now we can call the `off_ice_errors` function from GFTT and calculate off-ice errors:

```
fig, ax1 = plt.subplots(1, 2, figsize=(15, 7))

for ax in ax1:
    ax.axis('equal')
vx, vy, z, thres_idx = gftt.off_ice_errors(vxfile=demo.Vx, vyfile=demo.Vy,
off_ice_area=in_shp, ax=ax1[0])
gftt.plot_off_ice_errors(vx, vy, z, thres_idx, ax=ax1[1])
```



Here, x-axis is V_x (m/day), y-axis is V_y (m/day). We randomly select 10,000 pixels and color code them based on the Gaussian kernel density estimate, and the other pixels are plotting as gray points. Transparent color points have a kernel density below a chosen confidence level, which is 3.0 by default and corresponds to a confidence level of 99.7%.

To see how close these points approximate a normal distribution errors are randomly distributed, we can quickly calculate how much the selected pixel account for. For a 3-sigma threshold, the non-transparent points should account for ~99.7% of all the points:

```
np.sum(thres_idx) / vx.shape
```

```
array([0.9749])
```

This gives us 97.5%, indicating more points falling outside of the chosen confidence interval than assumed. This is okay because the standard deviation of these points would be a conservative estimate, affected by many outliers and larger than the actual standard deviation.

The 3-sigma error can be thus derived by finding the boundary of the selected pixels. These numbers should resemble the overall uncertainty of both on-ice and off-ice pixels, and we use them as a metric for estimating the feature tracking results.

```
print('Vx error (3-sigma): {:5f}  m/day'.format(0.5 * (max(vx[thres_idx]) -
min(vx[thres_idx]))))
print('Vy error (3-sigma): {:5f}  m/day'.format(0.5 * (max(vy[thres_idx]) -
min(vy[thres_idx]))))
```

```
Vx error (3-sigma): 0.212402  m/day
Vy error (3-sigma): 0.223389  m/day
```

3.2 Demo 2

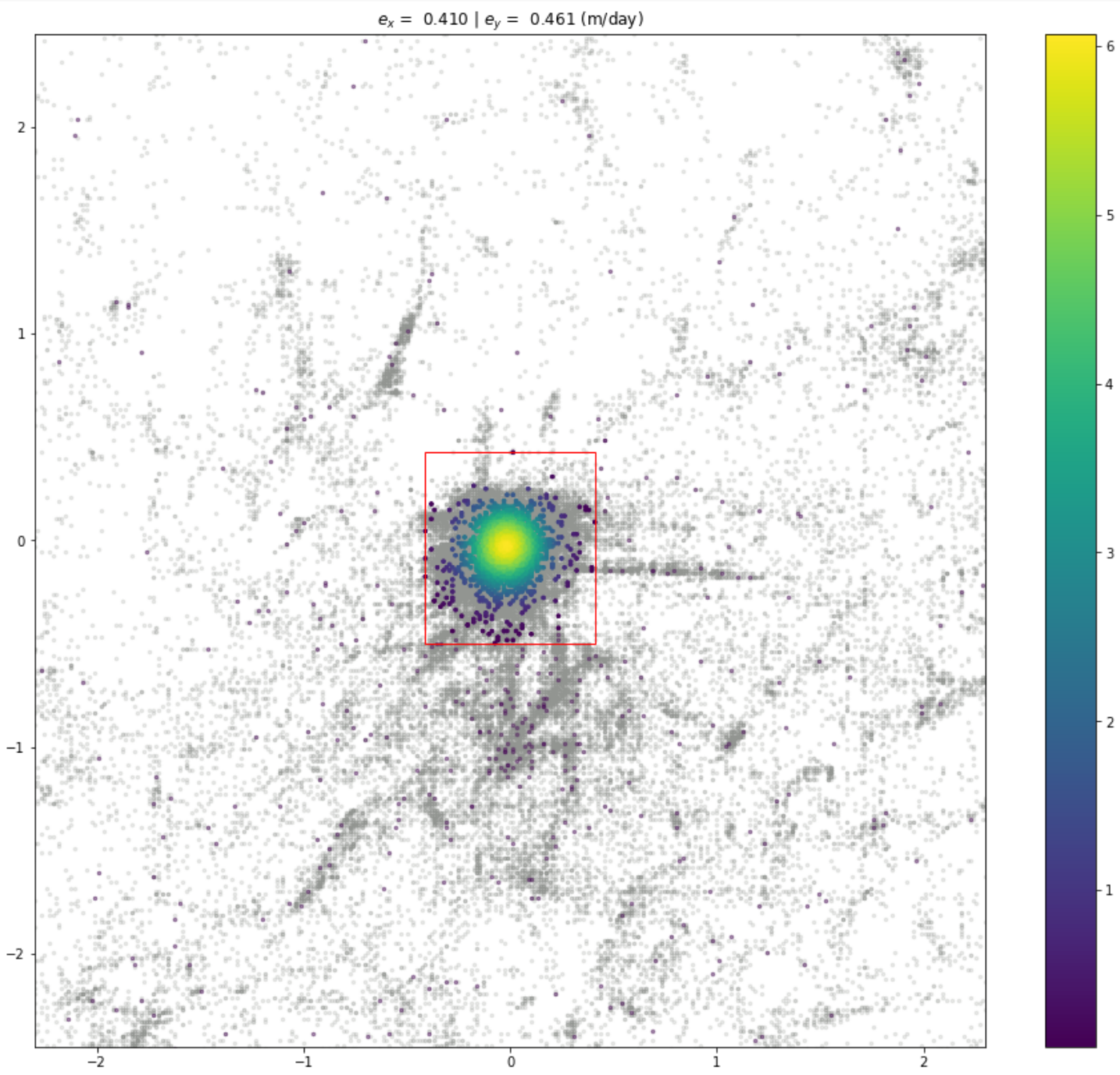
This demo provides a bigger output figure with the error bound shown as a red box.

```
demo = df.loc[18]
demo
```

```
Date                                                LS8-20180304-20180405
Chip (px)                                                              32
Resolution (px)                                                         1
Prefilter                                                             Gau
Subpixel                                          16-node oversampling
Software                                                           CARST
Vx                  /home/jovyan/Projects/PX_comparison/PX/CARST/2...
Vy                  /home/jovyan/Projects/PX_comparison/PX/CARST/2...
Name: 18, dtype: object
```

```
viridis = cm.get_cmap('viridis', 12)
pt_style = {'s': 6, 'edgecolor': None}

fig, axt = plt.subplots(1, 1, figsize=(16, 14))
vx, vy, z, thres_idx = gftt.off_ice_errors(vxfile=demo.Vx, vyfile=demo.Vy,
off_ice_area=in_shp, max_n=10000, plot=True, ax=axt)

idx = thres_idx    # alias
axt.axis('equal')
tmp = axt.scatter(vx[idx], vy[idx], c=z[idx], **pt_style)
# axt.scatter(vx[~idx], vy[~idx], color=viridis(0), alpha=0.4, **pt_style)
# # axt.set_xlim((min(vx[idx]), max(vx[idx])))
# # axt.set_ylim((min(vy[idx]), max(vy[idx])))
axt.set_xlim(-2.3, 2.3)
axt.set_ylim(-2.3, 2.3)

# (left, bottom), width, height
rect = patches.Rectangle((min(vx[thres_idx]), min(vy[thres_idx])), max(vx[thres_idx]) -
min(vx[thres_idx]), max(vy[thres_idx]) - min(vy[thres_idx]),
                         linewidth=1, edgecolor='r', facecolor='none')
axt.add_patch(rect)
axt.set_title('$e_x$ = {:6.3f} | $e_y$ = {:6.3f} (m/day)'.format(0.5 * (max(vx[thres_idx])
- min(vx[thres_idx])),
                                                                 0.5 * (max(vy[thres_idx])
- min(vy[thres_idx]))))

fig.colorbar(tmp)
```

```
<matplotlib.colorbar.Colorbar at 0x7f98d0bcbbe0>
```



4. Off-ice error for 42 tests using the same Landsat 8 pair

Here’s a complete list of all 42 tests with their parameter settings.

```
## Example results

# demo = df.loc[df['Date'] == 'LS8-20180304-20180405']
demo = df.loc[np.logical_and(df['Date'] == 'LS8-20180304-20180405', df['Subpixel'] !=
'affine')]
# demo = df.loc[df['Date'] == 'LS8-20180802-20180818']
# demo = df.loc[np.logical_and(df['Date'] == 'LS8-20180802-20180818', df['Subpixel'] !=
'affine')]
# demo = df.loc[df['Date'] == 'Sen2-20180304-20180314']
# demo = df.loc[np.logical_and(df['Date'] == 'Sen2-20180304-20180314', df['Subpixel'] !=
'affine')]
# demo = df.loc[df['Date'] == 'Sen2-20180508-20180627']
# demo = df.loc[np.logical_and(df['Date'] == 'Sen2-20180508-20180627', df['Subpixel'] !=
'affine')]
# There are 43-1 tests
demo
```

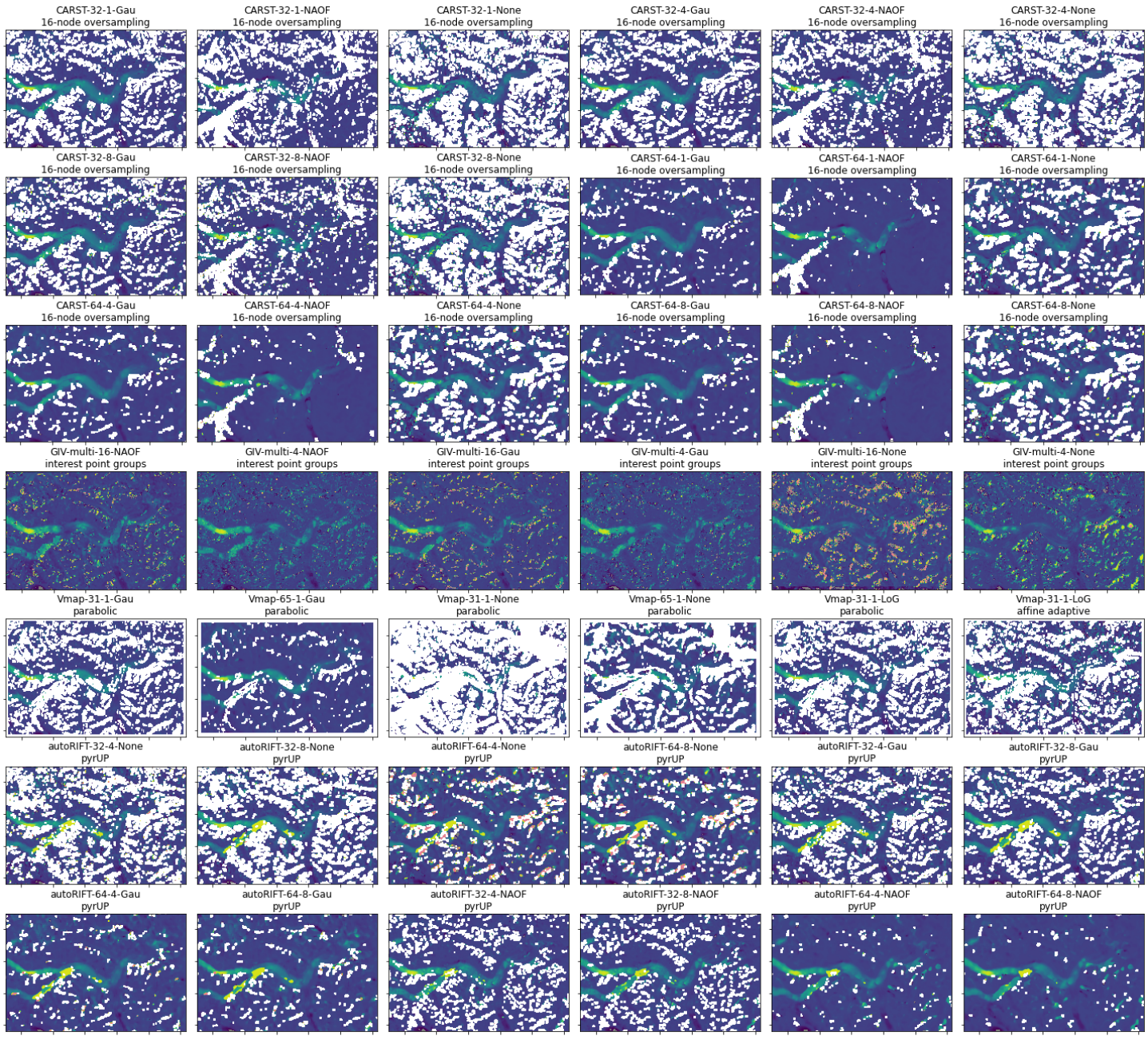
	Date	Chip (px)	Resolution (px)	Prefilter	Subpixel	Software	
18	LS8-20180304-20180405	32	1	Gau	16-node oversampling	CARST	/home/jovyan/Project
19	LS8-20180304-20180405	32	1	NAOF	16-node oversampling	CARST	/home/jovyan/Project
20	LS8-20180304-20180405	32	1	None	16-node oversampling	CARST	/home/jovyan/Project
21	LS8-20180304-20180405	32	4	Gau	16-node oversampling	CARST	/home/jovyan/Project
22	LS8-20180304-20180405	32	4	NAOF	16-node oversampling	CARST	/home/jovyan/Project
23	LS8-20180304-20180405	32	4	None	16-node oversampling	CARST	/home/jovyan/Project
24	LS8-20180304-20180405	32	8	Gau	16-node oversampling	CARST	/home/jovyan/Project
25	LS8-20180304-20180405	32	8	NAOF	16-node oversampling	CARST	/home/jovyan/Project
26	LS8-20180304-20180405	32	8	None	16-node oversampling	CARST	/home/jovyan/Project
27	LS8-20180304-20180405	64	1	Gau	16-node oversampling	CARST	/home/jovyan/Project
28	LS8-20180304-20180405	64	1	NAOF	16-node oversampling	CARST	/home/jovyan/Project
29	LS8-20180304-20180405	64	1	None	16-node oversampling	CARST	/home/jovyan/Project
30	LS8-20180304-20180405	64	4	Gau	16-node oversampling	CARST	/home/jovyan/Project
31	LS8-20180304-20180405	64	4	NAOF	16-node oversampling	CARST	/home/jovyan/Project
32	LS8-20180304-20180405	64	4	None	16-node oversampling	CARST	/home/jovyan/Project
33	LS8-20180304-20180405	64	8	Gau	16-node oversampling	CARST	/home/jovyan/Project
34	LS8-20180304-20180405	64	8	NAOF	16-node oversampling	CARST	/home/jovyan/Project
35	LS8-20180304-20180405	64	8	None	16-node oversampling	CARST	/home/jovyan/Project

	Date	Chip (px)	Resolution (px)	Prefilter	Subpixel	Software	
72	LS8-20180304-20180405	multi	16	NAOF	interest point groups	GIV	/home/jovyan/Proje
73	LS8-20180304-20180405	multi	4	NAOF	interest point groups	GIV	/home/jovyan/Proje
74	LS8-20180304-20180405	multi	16	Gau	interest point groups	GIV	/home/jovyan/Proje
75	LS8-20180304-20180405	multi	4	Gau	interest point groups	GIV	/home/jovyan/Proje
76	LS8-20180304-20180405	multi	16	None	interest point groups	GIV	/home/jovyan/Proje
77	LS8-20180304-20180405	multi	4	None	interest point groups	GIV	/home/jovyan/Proje
96	LS8-20180304-20180405	31	1	Gau	parabolic	Vmap	/home/jovyan/Project
97	LS8-20180304-20180405	65	1	Gau	parabolic	Vmap	/home/jovyan/Project
104	LS8-20180304-20180405	31	1	None	parabolic	Vmap	/home/jovyan/Project
105	LS8-20180304-20180405	65	1	None	parabolic	Vmap	/home/jovyan/Project
112	LS8-20180304-20180405	31	1	LoG	parabolic	Vmap	/home/jovyan/Project
113	LS8-20180304-20180405	31	1	LoG	affine adaptive	Vmap	/home/jovyan/Project
124	LS8-20180304-20180405	32	4	None	pyrUP	autoRIFT	/home/jovyan/Proje
125	LS8-20180304-20180405	32	8	None	pyrUP	autoRIFT	/home/jovyan/Proje
126	LS8-20180304-20180405	64	4	None	pyrUP	autoRIFT	/home/jovyan/Proje
127	LS8-20180304-20180405	64	8	None	pyrUP	autoRIFT	/home/jovyan/Proje
128	LS8-20180304-20180405	32	4	Gau	pyrUP	autoRIFT	/home/jovyan/Proje
129	LS8-20180304-20180405	32	8	Gau	pyrUP	autoRIFT	/home/jovyan/Proje

	Date	Chip (px)	Resolution (px)	Prefilter	Subpixel	Software	
130	LS8-20180304-20180405	64	4	Gau	pyrUP	autoRIFT	/home/jovyan/Proje
131	LS8-20180304-20180405	64	8	Gau	pyrUP	autoRIFT	/home/jovyan/Proje
132	LS8-20180304-20180405	32	4	NAOF	pyrUP	autoRIFT	/home/jovyan/Proje
133	LS8-20180304-20180405	32	8	NAOF	pyrUP	autoRIFT	/home/jovyan/Proje
134	LS8-20180304-20180405	64	4	NAOF	pyrUP	autoRIFT	/home/jovyan/Proje
135	LS8-20180304-20180405	64	8	NAOF	pyrUP	autoRIFT	/home/jovyan/Proje

And here's what the data (V_x) look like:

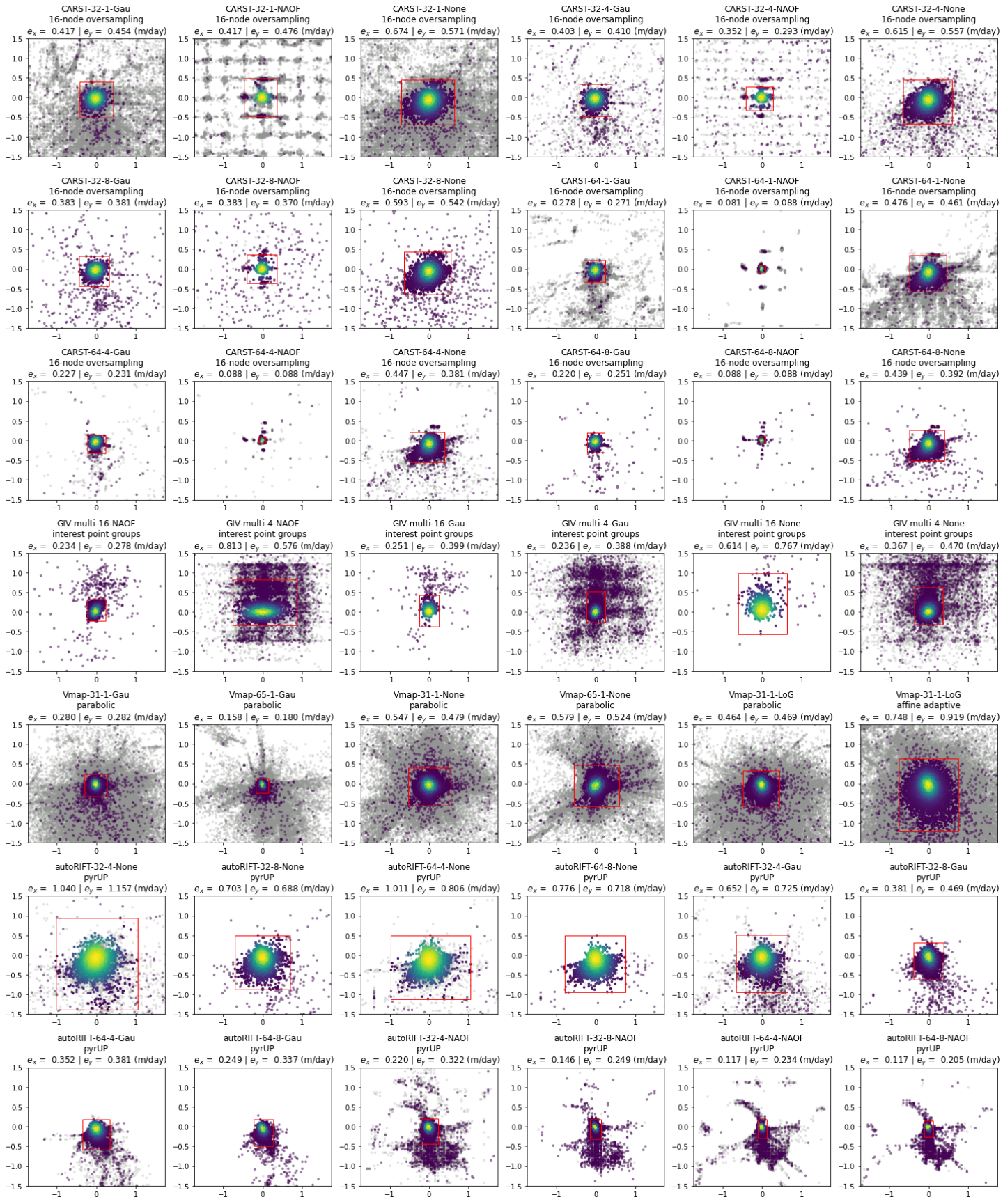
```
fig, ax3 = plt.subplots(7, 6, figsize=(20, 18))

n = 0

for idx, row in demo.iterrows():
    label = '-'.join((row.Software, row['Chip (px)'], row['Resolution (px)'],
row.Prefilter)) + '\n' + row.Subpixel
    ax_sel = ax3[n // 6, n % 6]

    with rasterio.open(row.Vx) as vx_src:
        show(vx_src, ax=ax_sel, vmin=-0.5, vmax=2, cmap=cmap)
    ax_sel.set_title(label)
    ax_sel.xaxis.set_ticklabels([])
    ax_sel.yaxis.set_ticklabels([])

    n += 1

plt.tight_layout()
fig.patch.set_facecolor('xkcd:white')
# fig.savefig('tmp.png')
```



Off-ice error plots for all 42 tests:

```
fig, ax4 = plt.subplots(7, 6, figsize=(20, 24))

n = 0

for idx, row in demo.iterrows():
    label = '-'.join((row.Software, row['Chip (px)'], row['Resolution (px)'],
row.Prefilter)) + '\n' + row.Subpixel
    ax_sel = ax4[n // 6, n % 6]
    ax_sel.axis('equal')

    if row.Software == 'GIV':
        vx, vy, z, thres_idx = gftt.off_ice_errors(vxfile=row.Vx, vyfile=row.Vy,
wfile=row.Vx.replace('u_', 'pkr_'), off_ice_area=in_shp, max_n=10000, plot=True,
ax=ax_sel)
    else:
        vx, vy, z, thres_idx = gftt.off_ice_errors(vxfile=row.Vx, vyfile=row.Vy,
off_ice_area=in_shp, max_n=10000, plot=True, ax=ax_sel)
    # gftt.plot_off_ice_errors(vx, vy, z, thres_idx, ax=ax_sel, zoom=True)
    ax_sel.set_xlim(-1.5, 1.5)
    ax_sel.set_ylim(-1.5, 1.5)

    rect = patches.Rectangle((min(vx[thres_idx]), min(vy[thres_idx])), max(vx[thres_idx])
- min(vx[thres_idx]), max(vy[thres_idx]) - min(vy[thres_idx]),
                             linewidth=1, edgecolor='r', facecolor='none')
    ax_sel.add_patch(rect)

    demo.loc[idx, 'OIE-x'] = 0.5 * (max(vx[thres_idx]) - min(vx[thres_idx]))
    demo.loc[idx, 'OIE-y'] = 0.5 * (max(vy[thres_idx]) - min(vy[thres_idx]))
    label += '\n $e_x$ = {:6.3f} | $e_y$ = {:6.3f} (m/day)'.format(demo.loc[idx, 'OIE-x'],
                                                                   demo.loc[idx, 'OIE-y'])
    ax_sel.set_title(label)
    # ax_sel.xaxis.set_ticklabels([])
    # ax_sel.yaxis.set_ticklabels([])



    n += 1


plt.tight_layout()
fig.patch.set_facecolor('xkcd:white')
# fig.savefig('tmp.png')
```

```
/home/jovyan/.local/lib/python3.9/site-packages/pandas/core/indexing.py:1599:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self.obj[key] = infer_fill_value(value)
/home/jovyan/.local/lib/python3.9/site-packages/pandas/core/indexing.py:1720:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(loc, value, pi)
```



Results are attached to the right of the metadata sheet. (`OIE-x` and `OIE-y`)

```
# demo.to_csv('../results_agu21.csv', index=False)
demo
```

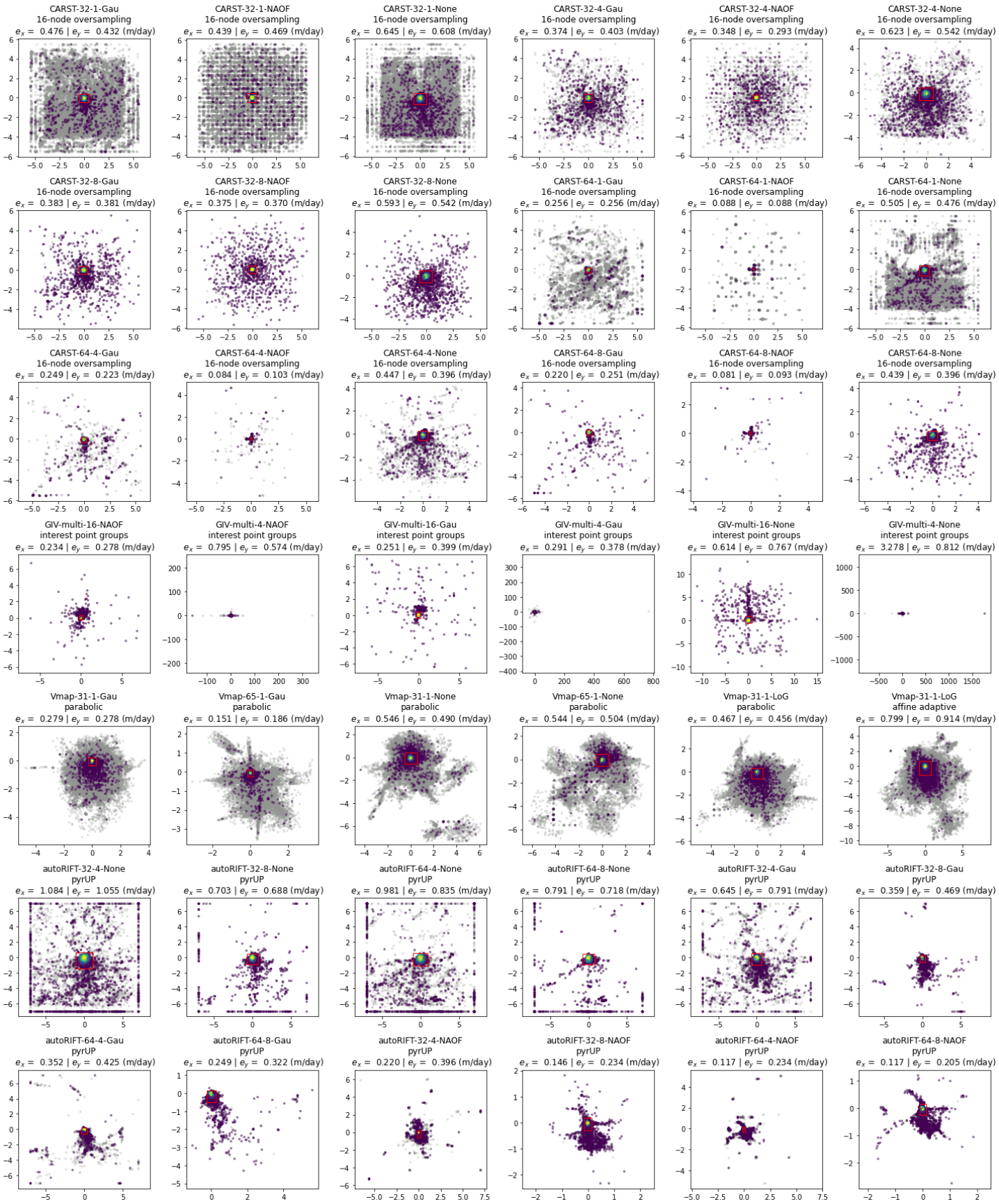
	Date	Chip (px)	Resolution (px)	Prefilter	Subpixel	Software	
18	LS8-20180304-20180405	32	1	Gau	16-node oversampling	CARST	/home/jovyan/Project
19	LS8-20180304-20180405	32	1	NAOF	16-node oversampling	CARST	/home/jovyan/Project
20	LS8-20180304-20180405	32	1	None	16-node oversampling	CARST	/home/jovyan/Project
21	LS8-20180304-20180405	32	4	Gau	16-node oversampling	CARST	/home/jovyan/Project
22	LS8-20180304-20180405	32	4	NAOF	16-node oversampling	CARST	/home/jovyan/Project
23	LS8-20180304-20180405	32	4	None	16-node oversampling	CARST	/home/jovyan/Project
24	LS8-20180304-20180405	32	8	Gau	16-node oversampling	CARST	/home/jovyan/Project
25	LS8-20180304-20180405	32	8	NAOF	16-node oversampling	CARST	/home/jovyan/Project
26	LS8-20180304-20180405	32	8	None	16-node oversampling	CARST	/home/jovyan/Project
27	LS8-20180304-20180405	64	1	Gau	16-node oversampling	CARST	/home/jovyan/Project
28	LS8-20180304-20180405	64	1	NAOF	16-node oversampling	CARST	/home/jovyan/Project
29	LS8-20180304-20180405	64	1	None	16-node oversampling	CARST	/home/jovyan/Project
30	LS8-20180304-20180405	64	4	Gau	16-node oversampling	CARST	/home/jovyan/Project
31	LS8-20180304-20180405	64	4	NAOF	16-node oversampling	CARST	/home/jovyan/Project
32	LS8-20180304-20180405	64	4	None	16-node oversampling	CARST	/home/jovyan/Project
33	LS8-20180304-20180405	64	8	Gau	16-node oversampling	CARST	/home/jovyan/Project
34	LS8-20180304-20180405	64	8	NAOF	16-node oversampling	CARST	/home/jovyan/Project
35	LS8-20180304-20180405	64	8	None	16-node oversampling	CARST	/home/jovyan/Project

	Date	Chip (px)	Resolution (px)	Prefilter	Subpixel	Software	
72	LS8-20180304-20180405	multi	16	NAOF	interest point groups	GIV	/home/jovyan/Proje
73	LS8-20180304-20180405	multi	4	NAOF	interest point groups	GIV	/home/jovyan/Proje
74	LS8-20180304-20180405	multi	16	Gau	interest point groups	GIV	/home/jovyan/Proje
75	LS8-20180304-20180405	multi	4	Gau	interest point groups	GIV	/home/jovyan/Proje
76	LS8-20180304-20180405	multi	16	None	interest point groups	GIV	/home/jovyan/Proje
77	LS8-20180304-20180405	multi	4	None	interest point groups	GIV	/home/jovyan/Proje
96	LS8-20180304-20180405	31	1	Gau	parabolic	Vmap	/home/jovyan/Project
97	LS8-20180304-20180405	65	1	Gau	parabolic	Vmap	/home/jovyan/Project
104	LS8-20180304-20180405	31	1	None	parabolic	Vmap	/home/jovyan/Project
105	LS8-20180304-20180405	65	1	None	parabolic	Vmap	/home/jovyan/Project
112	LS8-20180304-20180405	31	1	LoG	parabolic	Vmap	/home/jovyan/Project
113	LS8-20180304-20180405	31	1	LoG	affine adaptive	Vmap	/home/jovyan/Project
124	LS8-20180304-20180405	32	4	None	pyrUP	autoRIFT	/home/jovyan/Proje
125	LS8-20180304-20180405	32	8	None	pyrUP	autoRIFT	/home/jovyan/Proje
126	LS8-20180304-20180405	64	4	None	pyrUP	autoRIFT	/home/jovyan/Proje
127	LS8-20180304-20180405	64	8	None	pyrUP	autoRIFT	/home/jovyan/Proje
128	LS8-20180304-20180405	32	4	Gau	pyrUP	autoRIFT	/home/jovyan/Proje
129	LS8-20180304-20180405	32	8	Gau	pyrUP	autoRIFT	/home/jovyan/Proje

	Date	Chip (px)	Resolution (px)	Prefilter	Subpixel	Software	
130	LS8-20180304-20180405	64	4	Gau	pyrUP	autoRIFT	/home/jovyan/Proje
131	LS8-20180304-20180405	64	8	Gau	pyrUP	autoRIFT	/home/jovyan/Proje
132	LS8-20180304-20180405	32	4	NAOF	pyrUP	autoRIFT	/home/jovyan/Proje
133	LS8-20180304-20180405	32	8	NAOF	pyrUP	autoRIFT	/home/jovyan/Proje
134	LS8-20180304-20180405	64	4	NAOF	pyrUP	autoRIFT	/home/jovyan/Proje
135	LS8-20180304-20180405	64	8	NAOF	pyrUP	autoRIFT	/home/jovyan/Proje

And here's another figure showing the full extent of the off-ice area velocity plot.

```
fig, ax5 = plt.subplots(7, 6, figsize=(20, 24))

n = 0

for idx, row in demo.iterrows():
    label = '-'.join((row.Software, row['Chip (px)'], row['Resolution (px)'],
row.Prefilter)) + '\n' + row.Subpixel
    ax_sel = ax5[n // 6, n % 6]
    ax_sel.axis('equal')

    if row.Software == 'GIV':
        vx, vy, z, thres_idx = gftt.off_ice_errors(vxfile=row.Vx, vyfile=row.Vy,
wfile=row.Vx.replace('u_', 'pkr_'), off_ice_area=in_shp, max_n=10000, plot=True,
ax=ax_sel)
    else:
        vx, vy, z, thres_idx = gftt.off_ice_errors(vxfile=row.Vx, vyfile=row.Vy,
off_ice_area=in_shp, max_n=10000, plot=True, ax=ax_sel)
    # gftt.plot_off_ice_errors(vx, vy, z, thres_idx, ax=ax_sel, zoom=True)
    # ax_sel.set_xlim(-0.5, 0.5)
    # ax_sel.set_ylim(-0.5, 0.5)

    rect = patches.Rectangle((min(vx[thres_idx]), min(vy[thres_idx])), max(vx[thres_idx])
- min(vx[thres_idx]), max(vy[thres_idx]) - min(vy[thres_idx]),
                             linewidth=1, edgecolor='r', facecolor='none')
    ax_sel.add_patch(rect)
    label += '\n $e_x$ = {:6.3f} | $e_y$ = {:6.3f} (m/day)'.format(0.5 *
(max(vx[thres_idx]) - min(vx[thres_idx])),
                                                                   0.5 *
(max(vy[thres_idx]) - min(vy[thres_idx])))
    ax_sel.set_title(label)
    # ax_sel.xaxis.set_ticklabels([])
    # ax_sel.yaxis.set_ticklabels([])

    n += 1

plt.tight_layout()
fig.patch.set_facecolor('xkcd:white')
```



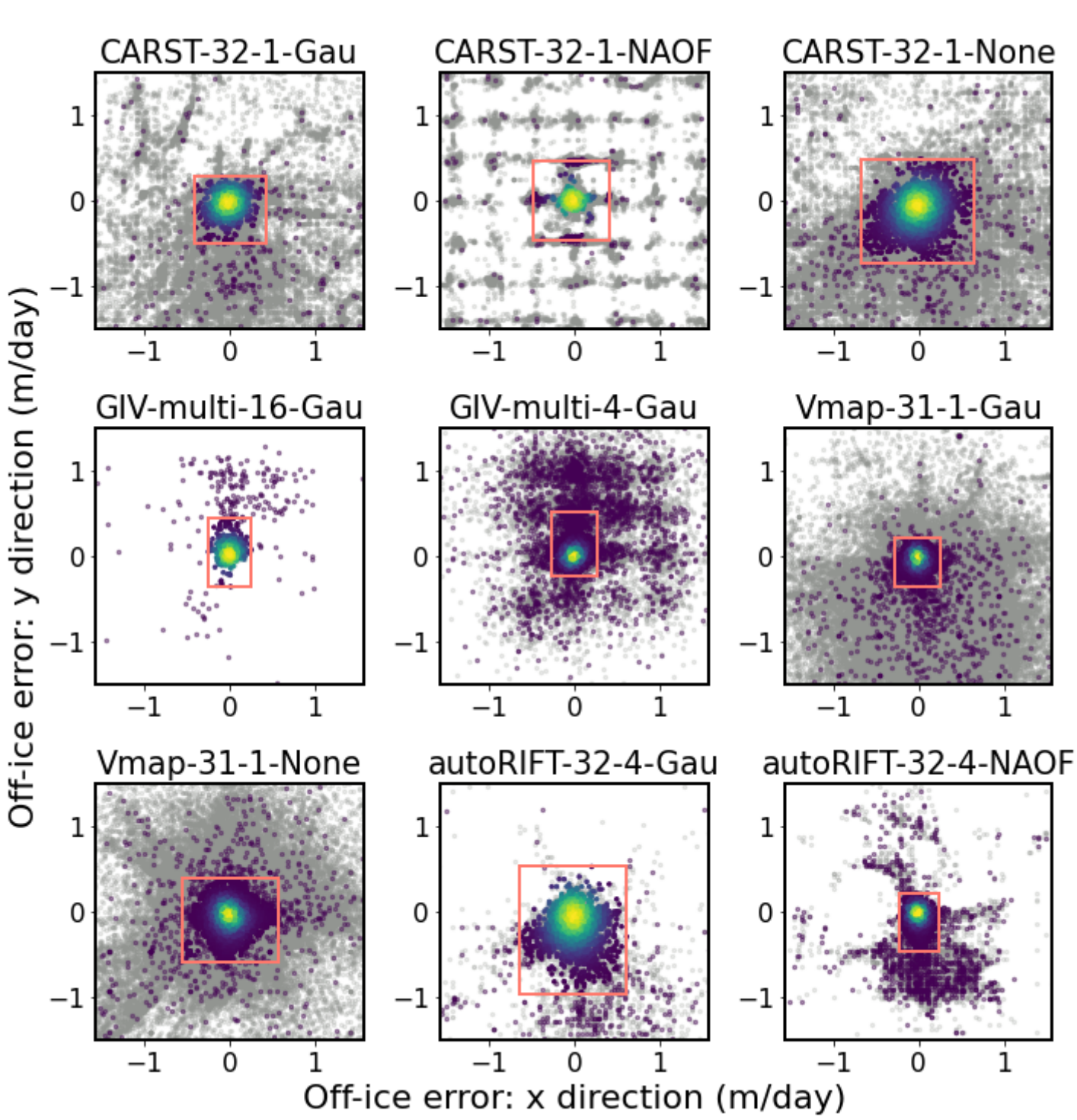
5. AGU 2021 poster, Figure 2

Here shows the script to make Figure in the AGU 2021 poster. We use only 9 tests for readability.

```
demo = df.loc[[18, 19, 20, 74,75, 96, 104, 128, 132]]
demo
```

	Date	Chip (px)	Resolution (px)	Prefilter	Subpixel	Software	
18	LS8-20180304-20180405	32	1	Gau	16-node oversampling	CARST	/home/jovyan/Project
19	LS8-20180304-20180405	32	1	NAOF	16-node oversampling	CARST	/home/jovyan/Project
20	LS8-20180304-20180405	32	1	None	16-node oversampling	CARST	/home/jovyan/Project
74	LS8-20180304-20180405	multi	16	Gau	interest point groups	GIV	/home/jovyan/Proje
75	LS8-20180304-20180405	multi	4	Gau	interest point groups	GIV	/home/jovyan/Proje
96	LS8-20180304-20180405	31	1	Gau	parabolic	Vmap	/home/jovyan/Project
104	LS8-20180304-20180405	31	1	None	parabolic	Vmap	/home/jovyan/Project
128	LS8-20180304-20180405	32	4	Gau	pyrUP	autoRIFT	/home/jovyan/Proje
132	LS8-20180304-20180405	32	4	NAOF	pyrUP	autoRIFT	/home/jovyan/Proje

```
font = {'size'   : 17}
matplotlib.rc('font', **font)
axes_settings = {'linewidth'   : 2}
# axes_settings = {'linewidth'   : 2, 'labelsize': 14}
matplotlib.rc('axes', **axes_settings)

fig, ax_agu = plt.subplots(3, 3, figsize=(10, 10.3))

n = 0

for idx, row in demo.iterrows():
    # label = '-'.join((row.Software, row['Chip (px)'], row['Resolution (px)'],
row.Prefilter)) + '\n' + row.Subpixel
    label = '-'.join((row.Software, row['Chip (px)'], row['Resolution (px)'],
row.Prefilter))
    ax_sel = ax_agu[n // 3, n % 3]
    ax_sel.axis('equal')

    if row.Software == 'GIV':
        vx, vy, z, thres_idx = gftt.off_ice_errors(vxfile=row.Vx, vyfile=row.Vy,
wfile=row.Vx.replace('u_', 'pkr_'), off_ice_area=in_shp, max_n=10000, plot=True,
ax=ax_sel)
    else:
        vx, vy, z, thres_idx = gftt.off_ice_errors(vxfile=row.Vx, vyfile=row.Vy,
off_ice_area=in_shp, max_n=10000, plot=True, ax=ax_sel)
    # gftt.plot_off_ice_errors(vx, vy, z, thres_idx, ax=ax_sel, zoom=True)
    ax_sel.set_xlim(-1.5, 1.5)
    ax_sel.set_ylim(-1.5, 1.5)

    rect = patches.Rectangle((min(vx[thres_idx]), min(vy[thres_idx])), max(vx[thres_idx])
- min(vx[thres_idx]), max(vy[thres_idx]) - min(vy[thres_idx]),
                             linewidth=2, edgecolor='xkcd:salmon', facecolor='none')
    ax_sel.add_patch(rect)

    demo.loc[idx, 'OIE-x'] = 0.5 * (max(vx[thres_idx]) - min(vx[thres_idx]))
    demo.loc[idx, 'OIE-y'] = 0.5 * (max(vy[thres_idx]) - min(vy[thres_idx]))
    # label += '\n $e_x$ = {:6.3f} | $e_y$ = {:6.3f} '.format(demo.loc[idx, 'OIE-x'],
    #                                                         demo.loc[idx, 'OIE-y'])
    ax_sel.set_title(label)
    # ax_sel.xaxis.set_ticklabels([])
    # ax_sel.yaxis.set_ticklabels([])



    n += 1


ax_agu[1, 0].set_ylabel('Off-ice error: y direction (m/day)', fontsize=22)
ax_agu[2, 1].set_xlabel('Off-ice error: x direction (m/day)', fontsize=22)

plt.tight_layout()
fig.patch.set_facecolor('xkcd:white')
# fig.savefig('OIE-9panels-AGU21_updated.png', dpi=450)
```



By Whyjay Zheng, Shashank Bhushan, Max Van Wyk de Vries, Will Kochtitzky, David Shean

Metric #2: Flow statistics on Sobel-filtered images

Contents

Sobel filter can highlight high-frequency variations on an image. Thus, we can apply sobel filter on the feature tracking results and calculate statistics of the on-ice pixels, which could serve as a metric to evaluate detail richness.

Requirements and Limits

- Have to prepare polygon(s) to select on-ice pixels.

Procedure

1. Prepare the feature tracking results as Geotiff files, ideally V_x as one file and V_y as the other. The unit for pixel value is flexible, from pixels, pixels/day, pixels/year, to meters, meters/day, to meters/year. **In this notebook, all of the feature tracking data use meters/day as the unit of the pixel value.**
2. Locate on-ice area and make an ESRI shapefile (polygon format) for that. Future development includes the RGI glacier outline as the on-ice area as well.
3. Use `gftt.sobel_scattering` and `gftt.plot_off_ice_errors` to calculate, analyze, and visualize the flow statistics.

Analysis

1. Basic information, importing modules

- Test area: **Kaskawulsh Glacier**
- Test packages: **CARST, Vmap, GIV, autoRIFT**
- Test pair: **Landsat 8 20180304-20180405**

```
# Developer's setting
%load_ext autoreload
%autoreload 2
```

```
import gftt
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.patches as patches
from matplotlib import colors, cm
from matplotlib.colors import ListedColormap
import rasterio
from rasterio.plot import show
import numpy as np
import pandas as pd
import geopandas as gpd
# import seaborn as sns
```

2. Load the feature tracking data list and on-ice area shapefile

Each row is a feature tracking test using a specific parameter combination described in each field.

```
df = pd.read_csv('../results_agu21.csv', dtype=str)
df
```

	Date	Chip (px)	Resolution (px)	Prefilter	Subpixel	Software	
0	LS8-20180304-20180405	32	1	Gau	16-node oversampling	CARST	/home/jovyan/Projects
1	LS8-20180304-20180405	32	1	NAOF	16-node oversampling	CARST	/home/jovyan/Projects
2	LS8-20180304-20180405	32	1	None	16-node oversampling	CARST	/home/jovyan/Projects
3	LS8-20180304-20180405	32	4	Gau	16-node oversampling	CARST	/home/jovyan/Projects
4	LS8-20180304-20180405	32	4	NAOF	16-node oversampling	CARST	/home/jovyan/Projects
5	LS8-20180304-20180405	32	4	None	16-node oversampling	CARST	/home/jovyan/Projects
6	LS8-20180304-20180405	32	8	Gau	16-node oversampling	CARST	/home/jovyan/Projects
7	LS8-20180304-20180405	32	8	NAOF	16-node oversampling	CARST	/home/jovyan/Projects
8	LS8-20180304-20180405	32	8	None	16-node oversampling	CARST	/home/jovyan/Projects
9	LS8-20180304-20180405	64	1	Gau	16-node oversampling	CARST	/home/jovyan/Projects
10	LS8-20180304-20180405	64	1	NAOF	16-node oversampling	CARST	/home/jovyan/Projects
11	LS8-20180304-20180405	64	1	None	16-node oversampling	CARST	/home/jovyan/Projects
12	LS8-20180304-20180405	64	4	Gau	16-node oversampling	CARST	/home/jovyan/Projects
13	LS8-20180304-20180405	64	4	NAOF	16-node oversampling	CARST	/home/jovyan/Projects
14	LS8-20180304-20180405	64	4	None	16-node oversampling	CARST	/home/jovyan/Projects
15	LS8-20180304-20180405	64	8	Gau	16-node oversampling	CARST	/home/jovyan/Projects
16	LS8-20180304-20180405	64	8	NAOF	16-node oversampling	CARST	/home/jovyan/Projects
17	LS8-20180304-20180405	64	8	None	16-node oversampling	CARST	/home/jovyan/Projects

	Date	Chip (px)	Resolution (px)	Prefilter	Subpixel	Software	
18	LS8-20180304-20180405	multi	16	NAOF	interest point groups	GIV	/home/jovyan/Projec
19	LS8-20180304-20180405	multi	4	NAOF	interest point groups	GIV	/home/jovyan/Projec
20	LS8-20180304-20180405	multi	16	Gau	interest point groups	GIV	/home/jovyan/Projec
21	LS8-20180304-20180405	multi	4	Gau	interest point groups	GIV	/home/jovyan/Projec
22	LS8-20180304-20180405	multi	16	None	interest point groups	GIV	/home/jovyan/Projec
23	LS8-20180304-20180405	multi	4	None	interest point groups	GIV	/home/jovyan/Projec
24	LS8-20180304-20180405	31	1	Gau	parabolic	Vmap	/home/jovyan/Projects,
25	LS8-20180304-20180405	65	1	Gau	parabolic	Vmap	/home/jovyan/Projects,
26	LS8-20180304-20180405	31	1	None	parabolic	Vmap	/home/jovyan/Projects,
27	LS8-20180304-20180405	65	1	None	parabolic	Vmap	/home/jovyan/Projects,
28	LS8-20180304-20180405	31	1	LoG	parabolic	Vmap	/home/jovyan/Projects
29	LS8-20180304-20180405	31	1	LoG	affine adaptive	Vmap	/home/jovyan/Projects
30	LS8-20180304-20180405	32	4	None	pyrUP	autoRIFT	/home/jovyan/Project
31	LS8-20180304-20180405	32	8	None	pyrUP	autoRIFT	/home/jovyan/Project
32	LS8-20180304-20180405	64	4	None	pyrUP	autoRIFT	/home/jovyan/Project
33	LS8-20180304-20180405	64	8	None	pyrUP	autoRIFT	/home/jovyan/Project
34	LS8-20180304-20180405	32	4	Gau	pyrUP	autoRIFT	/home/jovyan/Project
35	LS8-20180304-20180405	32	8	Gau	pyrUP	autoRIFT	/home/jovyan/Project

	Date	Chip (px)	Resolution (px)	Prefilter	Subpixel	Software	
36	LS8-20180304-20180405	64	4	Gau	pyrUP	autoRIFT	/home/jovyan/Project
37	LS8-20180304-20180405	64	8	Gau	pyrUP	autoRIFT	/home/jovyan/Project
38	LS8-20180304-20180405	32	4	NAOF	pyrUP	autoRIFT	/home/jovyan/Project
39	LS8-20180304-20180405	32	8	NAOF	pyrUP	autoRIFT	/home/jovyan/Project
40	LS8-20180304-20180405	64	4	NAOF	pyrUP	autoRIFT	/home/jovyan/Project
41	LS8-20180304-20180405	64	8	NAOF	pyrUP	autoRIFT	/home/jovyan/Project

Load the off-ice area shapefile.

```
# on-ice area
in_shp = '/home/jovyan/Projects/PX_comparison/Bedrock_shp/glacier_V1_Kaskawulsh_s.shp'
on_ice = gpd.read_file(in_shp)
```

3. Demo for a single test

```
## Example results
demo = df.loc[22]
print(demo)
```

```
Date                                          LS8-20180304-20180405
Chip (px)                                                     multi
Resolution (px)                                                  16
Prefilter                                                      None
Subpixel                                          interest point groups
Software                                                        GIV
Vx                 /home/jovyan/Projects/PX_comparison/PX/GIV/u_l...
Vy                 /home/jovyan/Projects/PX_comparison/PX/GIV/v_l...
OIE-x                                              0.6140104136459059
OIE-y                                              0.7672143993258354
SS-x                                               4.8425141531327425
SS-y                                                6.934244151078591
Name: 22, dtype: object
```

Now make a special `viridis` colormap designed for Kaskawulsh:

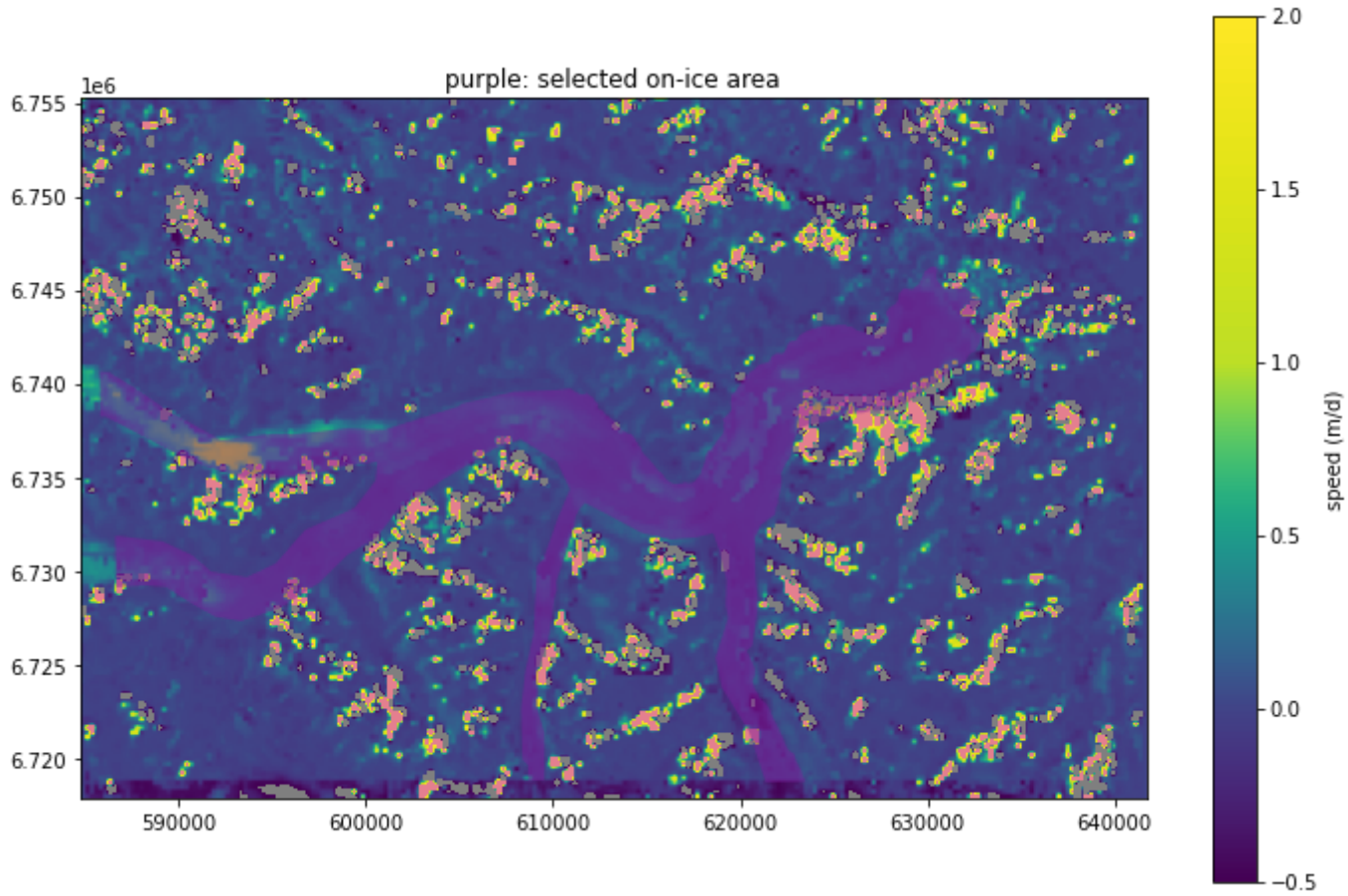
```
viridis = cm.get_cmap('viridis', 256)
newcolors1 = viridis(np.linspace(0, 0.2, 128))
newcolors2 = viridis(np.linspace(0.2, 0.9, 256))
newcolors3 = viridis(np.linspace(0.9, 1, 256))
newcolors = np.vstack((newcolors1, newcolors2, newcolors3))
cmap = ListedColormap(newcolors)
cmap.set_over(np.array([202,0,32,128]) / 255)
cmap.set_under(np.array([0,0,0,128]) / 255)
cmap
```

from_list

under bad over

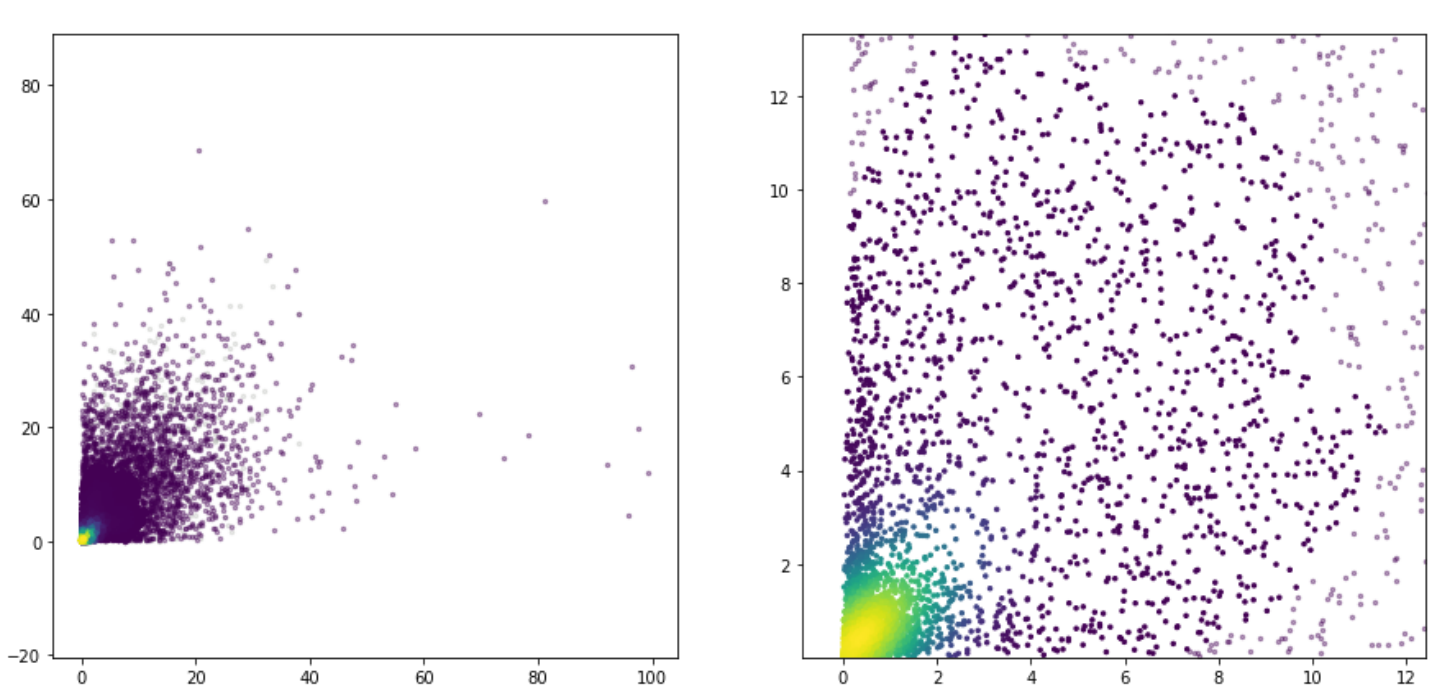
The velocity x component looks like this:

```
fig, ax0 = plt.subplots(1, 1, figsize=(12, 8))

with rasterio.open(demo.Vx) as v_src:
    v = v_src.read(1)
    show(v_src, ax=ax0, vmin=-0.5, vmax=2, cmap=cmap)

cbar =  fig.colorbar(cm.ScalarMappable(norm=colors.Normalize(vmin=-0.5, vmax=2),
cmap=cmap), ax=ax0)
cbar.set_label('speed (m/d)')
on_ice.plot(ax=ax0, alpha=0.5, facecolor='xkcd:purple')
ax0.set_title('purple: selected on-ice area');
```



Now we can call the `sobel_scattering` function from GFTT and calculate flow statistics:

```
fig, ax1 = plt.subplots(1, 2, figsize=(15, 7))

for ax in ax1:
    ax.axis('equal')
sx, sy, z, thres_idx = gftt.sobel_scattering(vxfile=demo.Vx, vyfile=demo.Vy,
on_ice_area=in_shp, ax=ax1[0])
gftt.plot_off_ice_errors(sx, sy, z, thres_idx, ax=ax1[1])

# limrange = (0, 3)
# ax1[0].set_xlim(limrange)
# ax1[0].set_ylim(limrange)
# ax1[1].set_xlim(limrange)
# ax1[1].set_ylim(limrange)
```

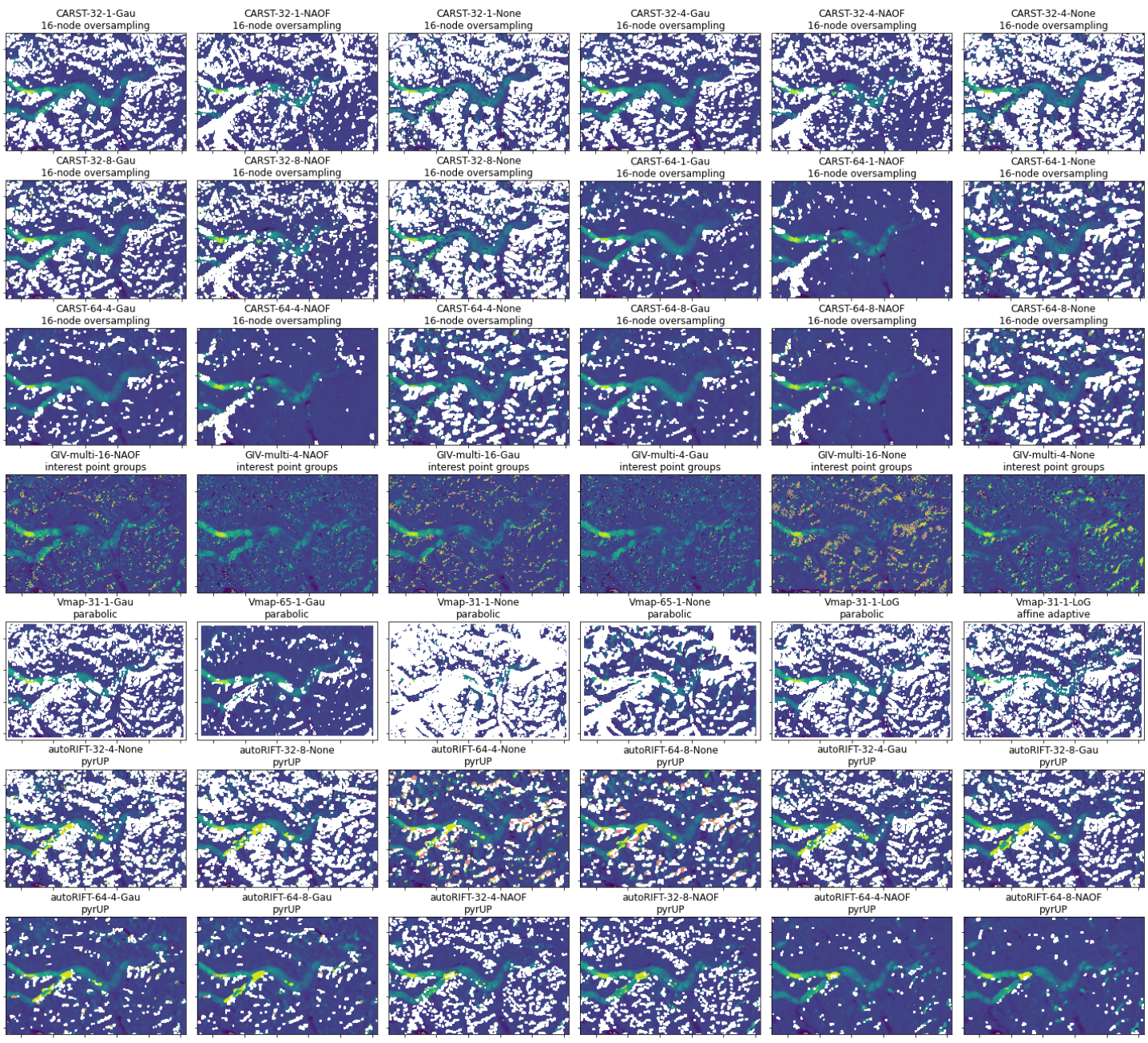


Here, x-axis is the (Sobel-flavored) gradient of flow speed along the x direction (m/day, per pixel), y-axis is the (Sobel-flavored) gradient of flow speed along the y direction (m/day, per pixel). Again, we randomly select 10,000 pixels and color code them based on the Gaussian kernel density estimate, and the other pixels are plotting as gray points. Transparent color points have a kernel density below a chosen confidence level, which is 3.0 by default and corresponds to a confidence level of 99.7%.

4. Off-ice error for 42 tests using the same Landsat 8 pair

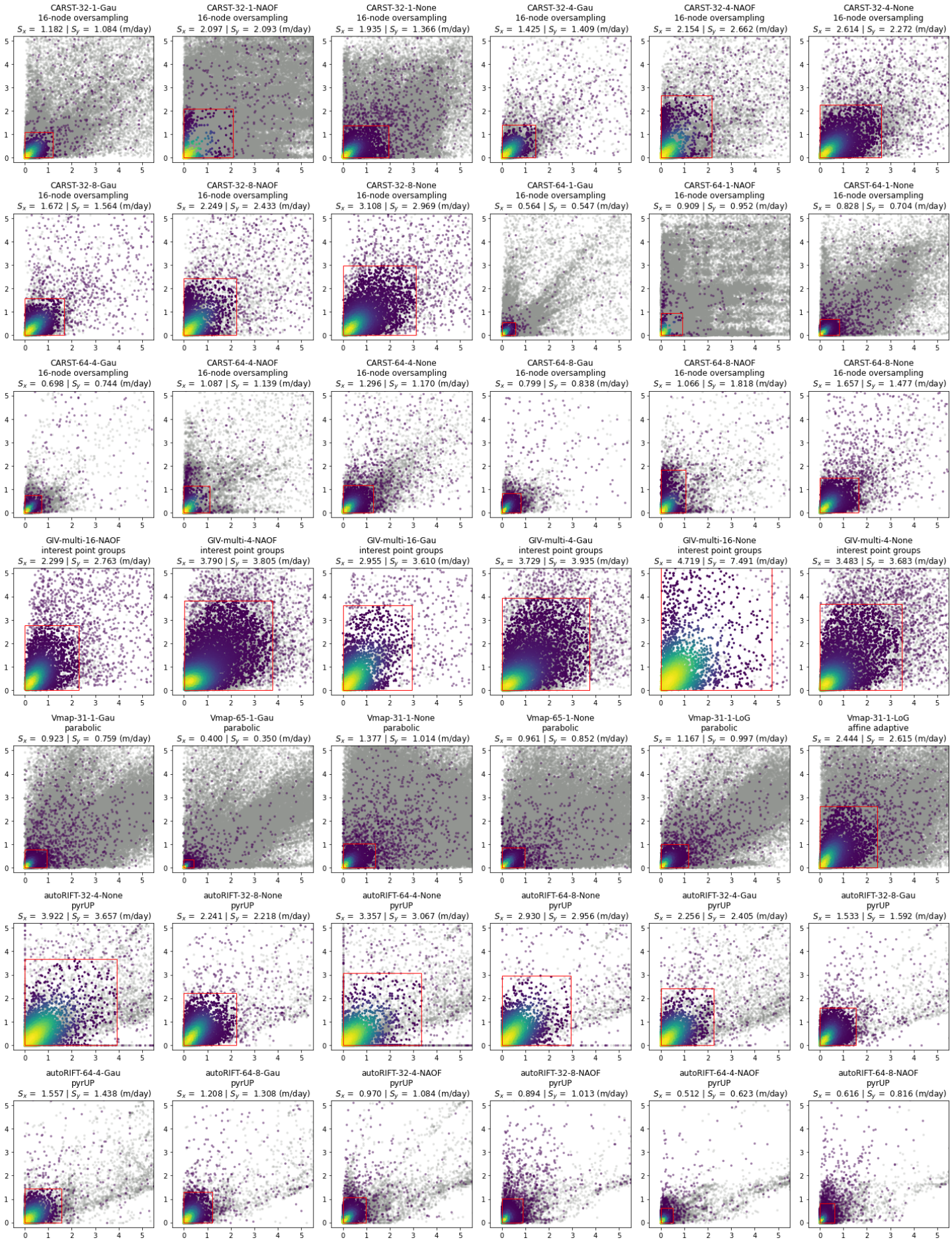
Here's what the data (V_x of all 42 tests) look like:

```
fig, ax2 = plt.subplots(7, 6, figsize=(20, 18))

demo = df
n = 0

for idx, row in demo.iterrows():
    label = '-'.join((row.Software, row['Chip (px)'], row['Resolution (px)'],
row.Prefilter)) + '\n' + row.Subpixel
    ax_sel = ax2[n // 6, n % 6]

    with rasterio.open(row.Vx) as vx_src:
        show(vx_src, ax=ax_sel, vmin=-0.5, vmax=2, cmap=cmap)
    ax_sel.set_title(label)
    ax_sel.xaxis.set_ticklabels([])
    ax_sel.yaxis.set_ticklabels([])

    n += 1

plt.tight_layout()
fig.patch.set_facecolor('xkcd:white')
# fig.savefig('tmp.png')
```



flow speed gradient plots for all 42 tests:

```
fig, ax3 = plt.subplots(7, 6, figsize=(20, 26))

demo = df
n = 0

for idx, row in demo.iterrows():
    label = '-'.join((row.Software, row['Chip (px)'], row['Resolution (px)'],
row.Prefilter)) + '\n' + row.Subpixel
    ax_sel = ax3[n // 6, n % 6]
    ax_sel.axis('equal')

    if row.Software == 'GIV':
        sx, sy, z, thres_idx = gftt.sobel_scattering(vxfile=row.Vx, vyfile=row.Vy,
wfile=row.Vx.replace('u_', 'pkr_'), on_ice_area=in_shp, max_n=10000, plot=True, ax=ax_sel)
    else:
        sx, sy, z, thres_idx = gftt.sobel_scattering(vxfile=row.Vx, vyfile=row.Vy,
on_ice_area=in_shp, max_n=10000, plot=True, ax=ax_sel)
    # gftt.plot_off_ice_errors(vx, vy, z, thres_idx, ax=ax_sel, zoom=True)
    ax_sel.set_xlim(0, 5)
    ax_sel.set_ylim(0, 5)

    rect = patches.Rectangle((min(sx[thres_idx]), min(sy[thres_idx])), max(sx[thres_idx])
- min(sx[thres_idx]), max(sy[thres_idx]) - min(sy[thres_idx]),
                             linewidth=1, edgecolor='r', facecolor='none')
    ax_sel.add_patch(rect)

    demo.loc[idx, 'SS-x'] = max(sx[thres_idx])
    demo.loc[idx, 'SS-y'] = max(sy[thres_idx])
    label += '\n $S_x$ = {:6.3f} | $S_y$ = {:6.3f} (m/day)'.format(demo.loc[idx, 'SS-x'],
                                                                   demo.loc[idx, 'SS-y'])
    ax_sel.set_title(label)
    # ax_sel.xaxis.set_ticklabels([])
    # ax_sel.yaxis.set_ticklabels([])



    n += 1


plt.tight_layout()
fig.patch.set_facecolor('xkcd:white')
# fig.savefig('tmp.png')
```



Results are attached to the right of the metadata sheet. (`SS-x` and `SS-y`)

```
# demo.to_csv('../results_agu21.csv', index=False)
demo
```

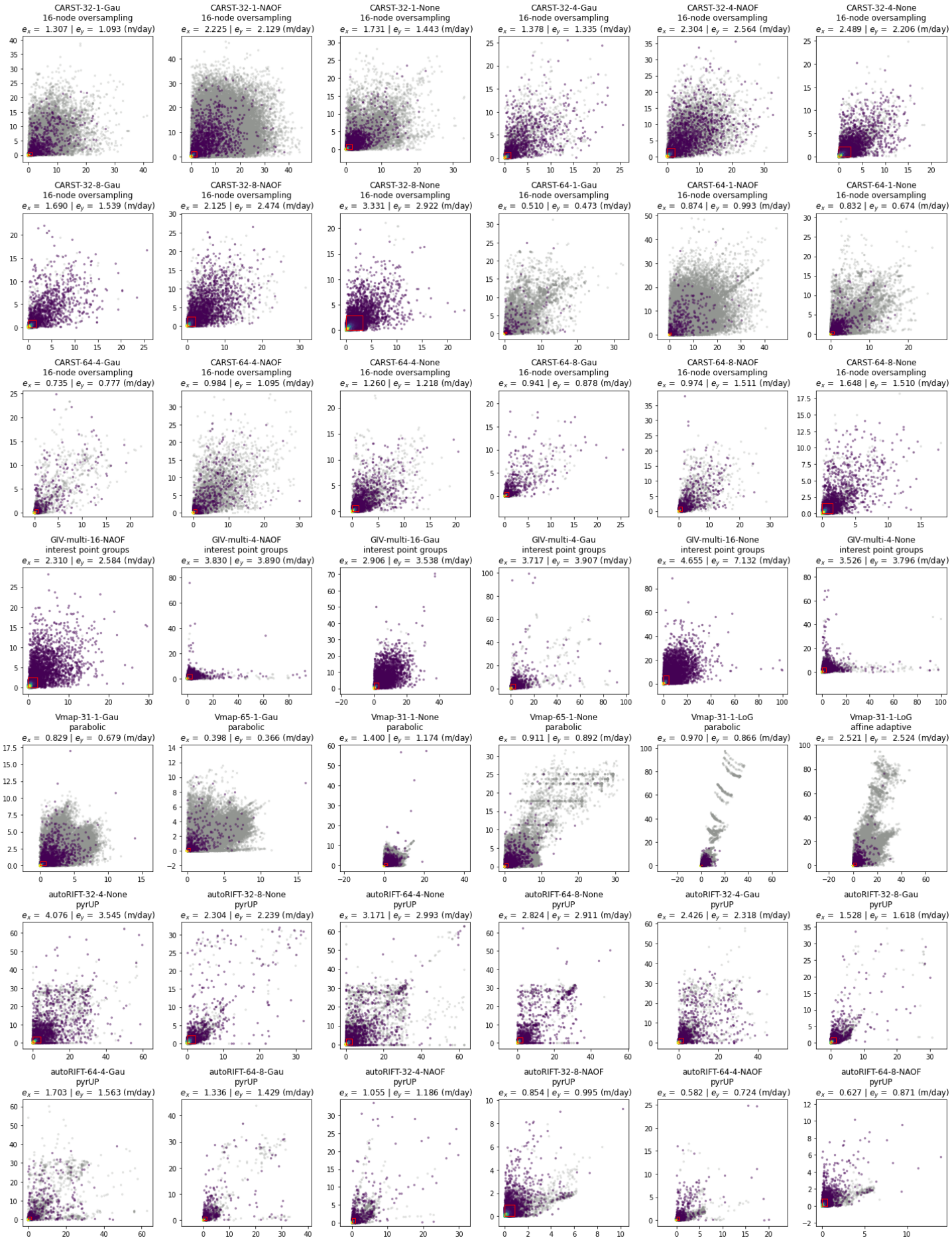
	Date	Chip (px)	Resolution (px)	Prefilter	Subpixel	Software	
0	LS8-20180304-20180405	32	1	Gau	16-node oversampling	CARST	/home/jovyan/Projects
1	LS8-20180304-20180405	32	1	NAOF	16-node oversampling	CARST	/home/jovyan/Projects
2	LS8-20180304-20180405	32	1	None	16-node oversampling	CARST	/home/jovyan/Projects
3	LS8-20180304-20180405	32	4	Gau	16-node oversampling	CARST	/home/jovyan/Projects
4	LS8-20180304-20180405	32	4	NAOF	16-node oversampling	CARST	/home/jovyan/Projects
5	LS8-20180304-20180405	32	4	None	16-node oversampling	CARST	/home/jovyan/Projects
6	LS8-20180304-20180405	32	8	Gau	16-node oversampling	CARST	/home/jovyan/Projects
7	LS8-20180304-20180405	32	8	NAOF	16-node oversampling	CARST	/home/jovyan/Projects
8	LS8-20180304-20180405	32	8	None	16-node oversampling	CARST	/home/jovyan/Projects
9	LS8-20180304-20180405	64	1	Gau	16-node oversampling	CARST	/home/jovyan/Projects
10	LS8-20180304-20180405	64	1	NAOF	16-node oversampling	CARST	/home/jovyan/Projects
11	LS8-20180304-20180405	64	1	None	16-node oversampling	CARST	/home/jovyan/Projects
12	LS8-20180304-20180405	64	4	Gau	16-node oversampling	CARST	/home/jovyan/Projects
13	LS8-20180304-20180405	64	4	NAOF	16-node oversampling	CARST	/home/jovyan/Projects
14	LS8-20180304-20180405	64	4	None	16-node oversampling	CARST	/home/jovyan/Projects
15	LS8-20180304-20180405	64	8	Gau	16-node oversampling	CARST	/home/jovyan/Projects
16	LS8-20180304-20180405	64	8	NAOF	16-node oversampling	CARST	/home/jovyan/Projects
17	LS8-20180304-20180405	64	8	None	16-node oversampling	CARST	/home/jovyan/Projects

	Date	Chip (px)	Resolution (px)	Prefilter	Subpixel	Software	
18	LS8-20180304-20180405	multi	16	NAOF	interest point groups	GIV	/home/jovyan/Projec
19	LS8-20180304-20180405	multi	4	NAOF	interest point groups	GIV	/home/jovyan/Projec
20	LS8-20180304-20180405	multi	16	Gau	interest point groups	GIV	/home/jovyan/Projec
21	LS8-20180304-20180405	multi	4	Gau	interest point groups	GIV	/home/jovyan/Projec
22	LS8-20180304-20180405	multi	16	None	interest point groups	GIV	/home/jovyan/Projec
23	LS8-20180304-20180405	multi	4	None	interest point groups	GIV	/home/jovyan/Projec
24	LS8-20180304-20180405	31	1	Gau	parabolic	Vmap	/home/jovyan/Projects,
25	LS8-20180304-20180405	65	1	Gau	parabolic	Vmap	/home/jovyan/Projects,
26	LS8-20180304-20180405	31	1	None	parabolic	Vmap	/home/jovyan/Projects,
27	LS8-20180304-20180405	65	1	None	parabolic	Vmap	/home/jovyan/Projects,
28	LS8-20180304-20180405	31	1	LoG	parabolic	Vmap	/home/jovyan/Projects
29	LS8-20180304-20180405	31	1	LoG	affine adaptive	Vmap	/home/jovyan/Projects
30	LS8-20180304-20180405	32	4	None	pyrUP	autoRIFT	/home/jovyan/Project
31	LS8-20180304-20180405	32	8	None	pyrUP	autoRIFT	/home/jovyan/Project
32	LS8-20180304-20180405	64	4	None	pyrUP	autoRIFT	/home/jovyan/Project
33	LS8-20180304-20180405	64	8	None	pyrUP	autoRIFT	/home/jovyan/Project
34	LS8-20180304-20180405	32	4	Gau	pyrUP	autoRIFT	/home/jovyan/Project
35	LS8-20180304-20180405	32	8	Gau	pyrUP	autoRIFT	/home/jovyan/Project

	Date	Chip (px)	Resolution (px)	Prefilter	Subpixel	Software	
36	LS8-20180304-20180405	64	4	Gau	pyrUP	autoRIFT	/home/jovyan/Project
37	LS8-20180304-20180405	64	8	Gau	pyrUP	autoRIFT	/home/jovyan/Project
38	LS8-20180304-20180405	32	4	NAOF	pyrUP	autoRIFT	/home/jovyan/Project
39	LS8-20180304-20180405	32	8	NAOF	pyrUP	autoRIFT	/home/jovyan/Project
40	LS8-20180304-20180405	64	4	NAOF	pyrUP	autoRIFT	/home/jovyan/Project
41	LS8-20180304-20180405	64	8	NAOF	pyrUP	autoRIFT	/home/jovyan/Project

And here's another figure showing the full extent of the flow velocity gradient plot.

```
fig, ax4 = plt.subplots(7, 6, figsize=(20, 26))

n = 0
demo = df

for idx, row in demo.iterrows():
    label = '-'.join((row.Software, row['Chip (px)'], row['Resolution (px)'],
row.Prefilter)) + '\n' + row.Subpixel
    ax_sel = ax4[n // 6, n % 6]
    ax_sel.axis('equal')

    if row.Software == 'GIV':
        sx, sy, z, thres_idx = gftt.sobel_scattering(vxfile=row.Vx, vyfile=row.Vy,
wfile=row.Vx.replace('u_', 'pkr_'), on_ice_area=in_shp, max_n=10000, plot=True, ax=ax_sel)
    else:
        sx, sy, z, thres_idx = gftt.sobel_scattering(vxfile=row.Vx, vyfile=row.Vy,
on_ice_area=in_shp, max_n=10000, plot=True, ax=ax_sel)
    # gftt.plot_off_ice_errors(vx, vy, z, thres_idx, ax=ax_sel, zoom=True)
    # ax_sel.set_xlim(-0.5, 0.5)
    # ax_sel.set_ylim(-0.5, 0.5)

    rect = patches.Rectangle((min(sx[thres_idx]), min(sy[thres_idx])), max(sx[thres_idx])
- min(sx[thres_idx]), max(sy[thres_idx]) - min(sy[thres_idx]),
                             linewidth=1, edgecolor='r', facecolor='none')
    ax_sel.add_patch(rect)
    label += '\n $e_x$ = {:6.3f} | $e_y$ = {:6.3f} (m/day)'.format(max(sx[thres_idx]),
                                                                   max(sy[thres_idx]))
    ax_sel.set_title(label)
    # ax_sel.xaxis.set_ticklabels([])
    # ax_sel.yaxis.set_ticklabels([])

    n += 1

plt.tight_layout()
fig.patch.set_facecolor('xkcd:white')
```



By Whyjay Zheng, Shashank Bhushan, Max Van Wyk de Vries, Will Kochtitzky, David Shean

Comparing Metric #1 (off-ice error) and #2 (Sobel flow statistics)

Contents

Analysis

1. Basic information, importing modules

- Test area: **Kaskawulsh Glacier**
- Test packages: **CARST, Vmap, GIV, autoRIFT**
- Test pair: **Landsat 8 20180304-20180405**

```
# Developer's setting
%load_ext autoreload
%autoreload 2
```

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

2. Load the feature tracking data list and the metrics for each test

Each row is a feature tracking test using a specific parameter combination described in each field.

```
df = pd.read_csv('../results_agu21.csv', dtype=str)
df['OIE-x'] = df['OIE-x'].astype(float)
df['OIE-y'] = df['OIE-y'].astype(float)
df['SS-x'] = df['SS-x'].astype(float)
df['SS-y'] = df['SS-y'].astype(float)
demo = df
demo
```

	Date	Chip (px)	Resolution (px)	Prefilter	Subpixel	Software	
0	LS8-20180304-20180405	32	1	Gau	16-node oversampling	CARST	/home/jovyan/Projects
1	LS8-20180304-20180405	32	1	NAOF	16-node oversampling	CARST	/home/jovyan/Projects
2	LS8-20180304-20180405	32	1	None	16-node oversampling	CARST	/home/jovyan/Projects
3	LS8-20180304-20180405	32	4	Gau	16-node oversampling	CARST	/home/jovyan/Projects
4	LS8-20180304-20180405	32	4	NAOF	16-node oversampling	CARST	/home/jovyan/Projects
5	LS8-20180304-20180405	32	4	None	16-node oversampling	CARST	/home/jovyan/Projects
6	LS8-20180304-20180405	32	8	Gau	16-node oversampling	CARST	/home/jovyan/Projects
7	LS8-20180304-20180405	32	8	NAOF	16-node oversampling	CARST	/home/jovyan/Projects
8	LS8-20180304-20180405	32	8	None	16-node oversampling	CARST	/home/jovyan/Projects
9	LS8-20180304-20180405	64	1	Gau	16-node oversampling	CARST	/home/jovyan/Projects
10	LS8-20180304-20180405	64	1	NAOF	16-node oversampling	CARST	/home/jovyan/Projects
11	LS8-20180304-20180405	64	1	None	16-node oversampling	CARST	/home/jovyan/Projects
12	LS8-20180304-20180405	64	4	Gau	16-node oversampling	CARST	/home/jovyan/Projects
13	LS8-20180304-20180405	64	4	NAOF	16-node oversampling	CARST	/home/jovyan/Projects
14	LS8-20180304-20180405	64	4	None	16-node oversampling	CARST	/home/jovyan/Projects
15	LS8-20180304-20180405	64	8	Gau	16-node oversampling	CARST	/home/jovyan/Projects
16	LS8-20180304-20180405	64	8	NAOF	16-node oversampling	CARST	/home/jovyan/Projects
17	LS8-20180304-20180405	64	8	None	16-node oversampling	CARST	/home/jovyan/Projects

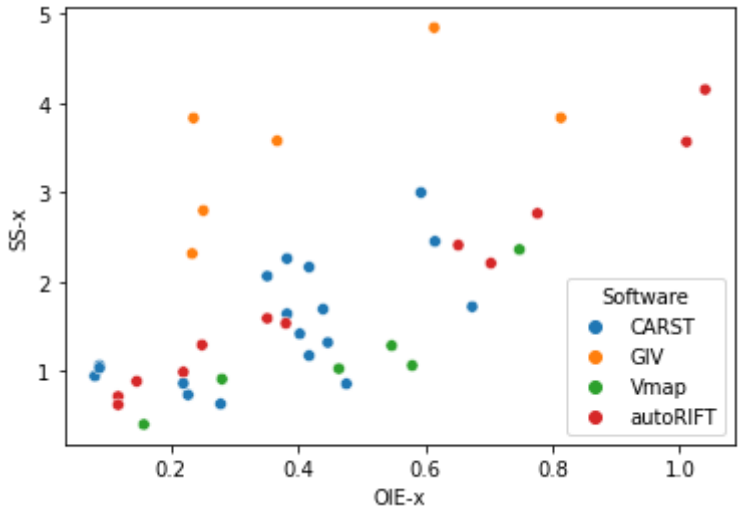
	Date	Chip (px)	Resolution (px)	Prefilter	Subpixel	Software	
18	LS8-20180304-20180405	multi	16	NAOF	interest point groups	GIV	/home/jovyan/Projec
19	LS8-20180304-20180405	multi	4	NAOF	interest point groups	GIV	/home/jovyan/Projec
20	LS8-20180304-20180405	multi	16	Gau	interest point groups	GIV	/home/jovyan/Projec
21	LS8-20180304-20180405	multi	4	Gau	interest point groups	GIV	/home/jovyan/Projec
22	LS8-20180304-20180405	multi	16	None	interest point groups	GIV	/home/jovyan/Projec
23	LS8-20180304-20180405	multi	4	None	interest point groups	GIV	/home/jovyan/Projec
24	LS8-20180304-20180405	31	1	Gau	parabolic	Vmap	/home/jovyan/Projects,
25	LS8-20180304-20180405	65	1	Gau	parabolic	Vmap	/home/jovyan/Projects,
26	LS8-20180304-20180405	31	1	None	parabolic	Vmap	/home/jovyan/Projects,
27	LS8-20180304-20180405	65	1	None	parabolic	Vmap	/home/jovyan/Projects,
28	LS8-20180304-20180405	31	1	LoG	parabolic	Vmap	/home/jovyan/Projects
29	LS8-20180304-20180405	31	1	LoG	affine adaptive	Vmap	/home/jovyan/Projects
30	LS8-20180304-20180405	32	4	None	pyrUP	autoRIFT	/home/jovyan/Project
31	LS8-20180304-20180405	32	8	None	pyrUP	autoRIFT	/home/jovyan/Project
32	LS8-20180304-20180405	64	4	None	pyrUP	autoRIFT	/home/jovyan/Project
33	LS8-20180304-20180405	64	8	None	pyrUP	autoRIFT	/home/jovyan/Project
34	LS8-20180304-20180405	32	4	Gau	pyrUP	autoRIFT	/home/jovyan/Project
35	LS8-20180304-20180405	32	8	Gau	pyrUP	autoRIFT	/home/jovyan/Project

	Date	Chip (px)	Resolution (px)	Prefilter	Subpixel	Software	
36	LS8-20180304-20180405	64	4	Gau	pyrUP	autoRIFT	/home/jovyan/Project
37	LS8-20180304-20180405	64	8	Gau	pyrUP	autoRIFT	/home/jovyan/Project
38	LS8-20180304-20180405	32	4	NAOF	pyrUP	autoRIFT	/home/jovyan/Project
39	LS8-20180304-20180405	32	8	NAOF	pyrUP	autoRIFT	/home/jovyan/Project
40	LS8-20180304-20180405	64	4	NAOF	pyrUP	autoRIFT	/home/jovyan/Project
41	LS8-20180304-20180405	64	8	NAOF	pyrUP	autoRIFT	/home/jovyan/Project

3. Sobel flow statistics VS Off-ice error

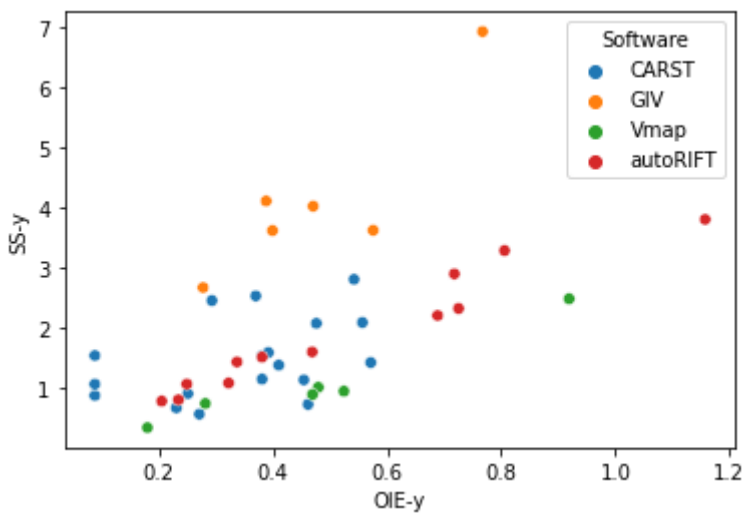
```
sns.scatterplot(data=demo, x='OIE-x', y='SS-x', hue='Software')
```

```
<AxesSubplot:xlabel='OIE-x', ylabel='SS-x'>
```



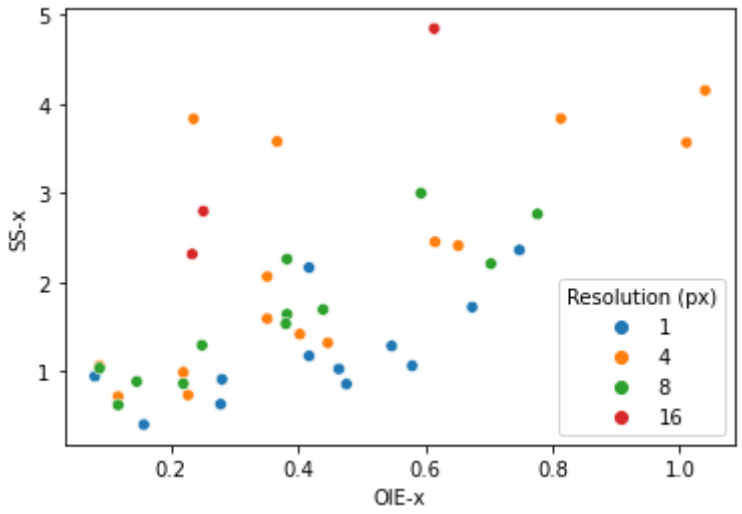

```
sns.scatterplot(data=demo, x='OIE-y', y='SS-y', hue='Software')
```

```
<AxesSubplot:xlabel='OIE-y', ylabel='SS-y'>
```



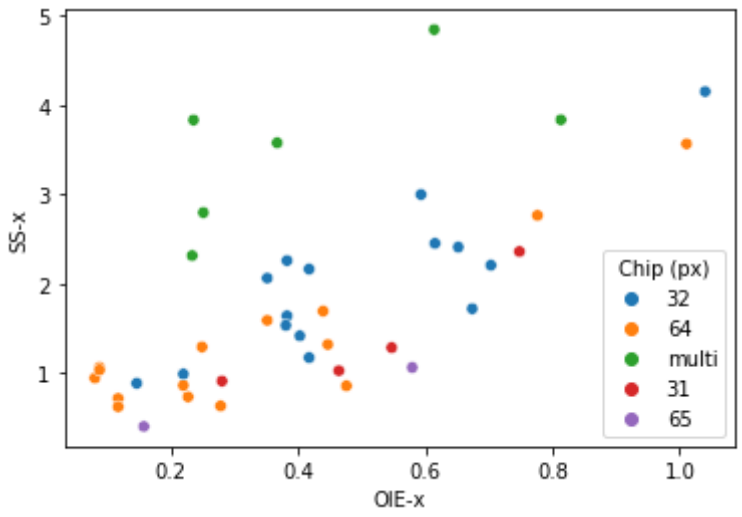

```
sns.scatterplot(data=demo, x='OIE-x', y='SS-x', hue='Resolution (px)')
```

```
<AxesSubplot:xlabel='OIE-x', ylabel='SS-x'>
```

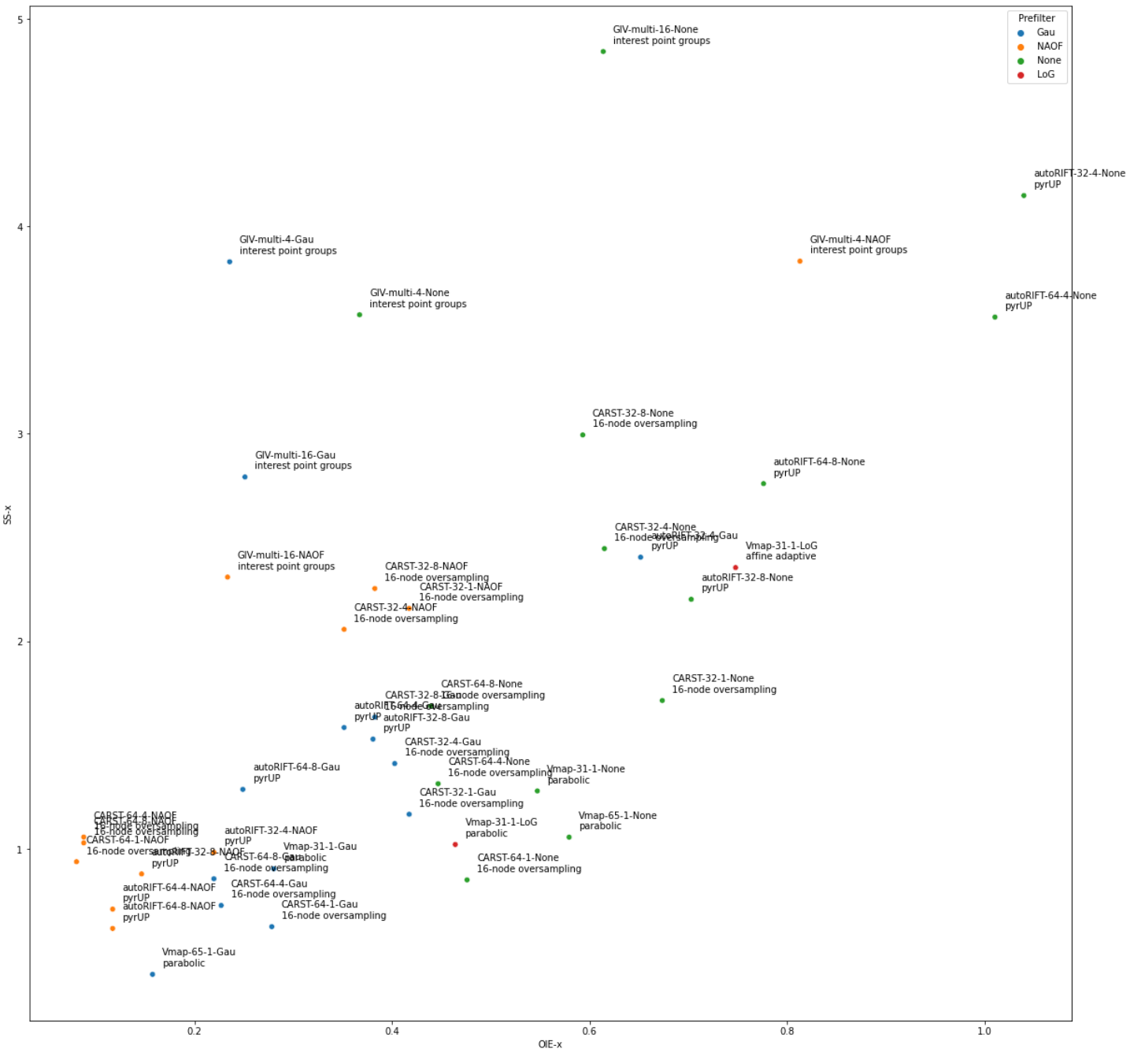



```
sns.scatterplot(data=demo, x='OIE-x', y='SS-x', hue='Chip (px)')
```

```
<AxesSubplot:xlabel='OIE-x', ylabel='SS-x'>
```

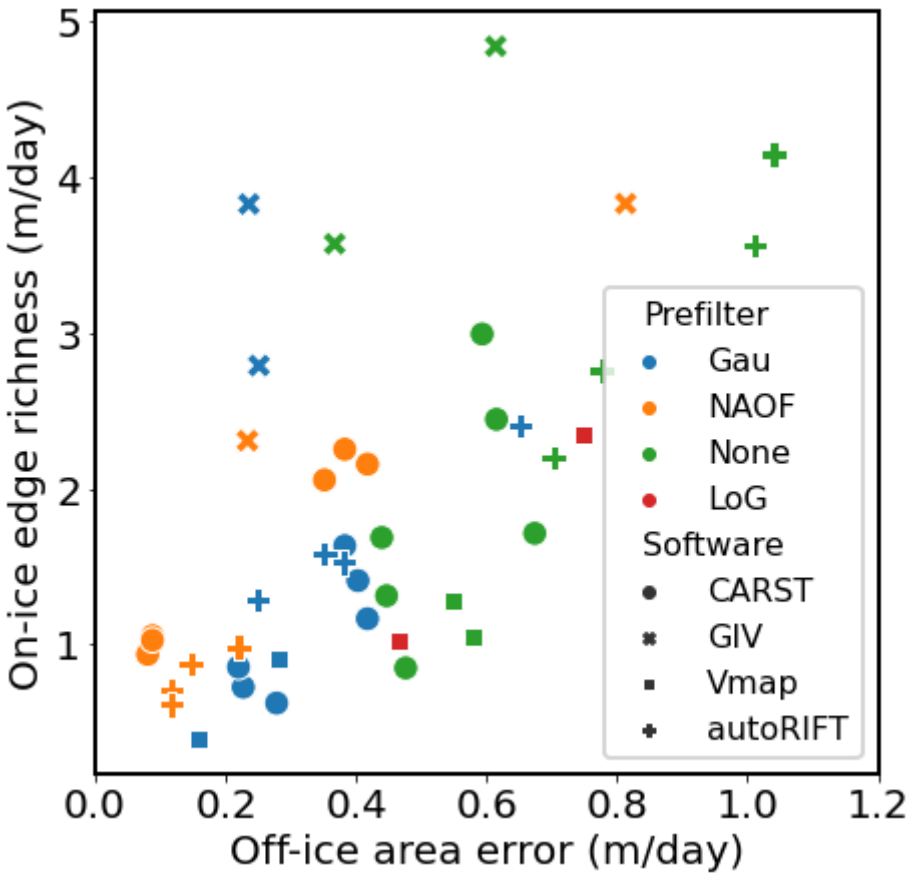



```
fig, ax5 = plt.subplots(1, 1, figsize=(20, 20))

sns.scatterplot(data=demo, x='OIE-x', y='SS-x', hue='Prefilter', ax=ax5)

for idx, row in demo.iterrows():
    label = '-'.join((row.Software, row['Chip (px)'], row['Resolution (px)'],
row.Prefilter)) + '\n' + row.Subpixel
    ax5.text(row['OIE-x'] + 0.01, row['SS-x'] + 0.04, label)
```



4. AGU 2021 poster, Figure 3

```
# font = {'family' : 'normal',
#         'weight' : 'bold',
#         'size'   : 22}
font = {'size'   : 20}
matplotlib.rc('font', **font)
matplotlib.rc('legend', fontsize=16)
axes_settings = {'linewidth'   : 2}
matplotlib.rc('axes', **axes_settings)
fig, ax6 = plt.subplots(1, 1, figsize=(7, 7))
sns.scatterplot(data=demo, x='OIE-x', y='SS-x', hue='Prefilter', style='Software', s=160,
ax=ax6, legend='full')
ax6.set_xlabel('Off-ice area error (m/day)')
ax6.set_ylabel('On-ice edge richness (m/day)')
ax6.set_xlim(0, 1.2)
legend = ax6.get_legend()
# legend.set_size(10)
legend.get_frame().set_linewidth(2)
fig.patch.set_facecolor('xkcd:white')
# fig.savefig('OIE-vs-SS_updated.svg')
```



By Whyjay Zheng, Shashank Bhushan, Max Van Wyk de Vries, Will Kochtitzky, David Shean

Validation #1: Compare GNSS data with metrics

Contents

– To be updated in the future work –

For each test:

1. We calculate the aforementioned metrics
2. We calculate the difference between GPS velocity (assumed as ground truth) and FT-derived velocity.
3. We compare the ground truth deviation and the metric and determine whether there is a correlation in between. Ideally, a good metric should indicate a certain type of correlation.

Requirements and Limits

- Availability of GNSS stations on a glacier

Procedure

Analysis

To be updated. Now you can only see 3 tests here without any metric values plotted on the figures.

```
%load_ext autoreload
%autoreload 2
```

```
import rasterio
import matplotlib.pyplot as plt
from rasterio.plot import show as rio_show
import matplotlib as mpl
import pandas as pd
import geopandas as gpd
import numpy as np
```

```
/data/whyj/Software/anaconda3/envs/carst/lib/python3.5/importlib/_bootstrap.py:222:
RuntimeWarning: numpy.dtype size changed, may indicate binary incompatibility. Expected
96, got 88
  return f(*args, **kwds)
```

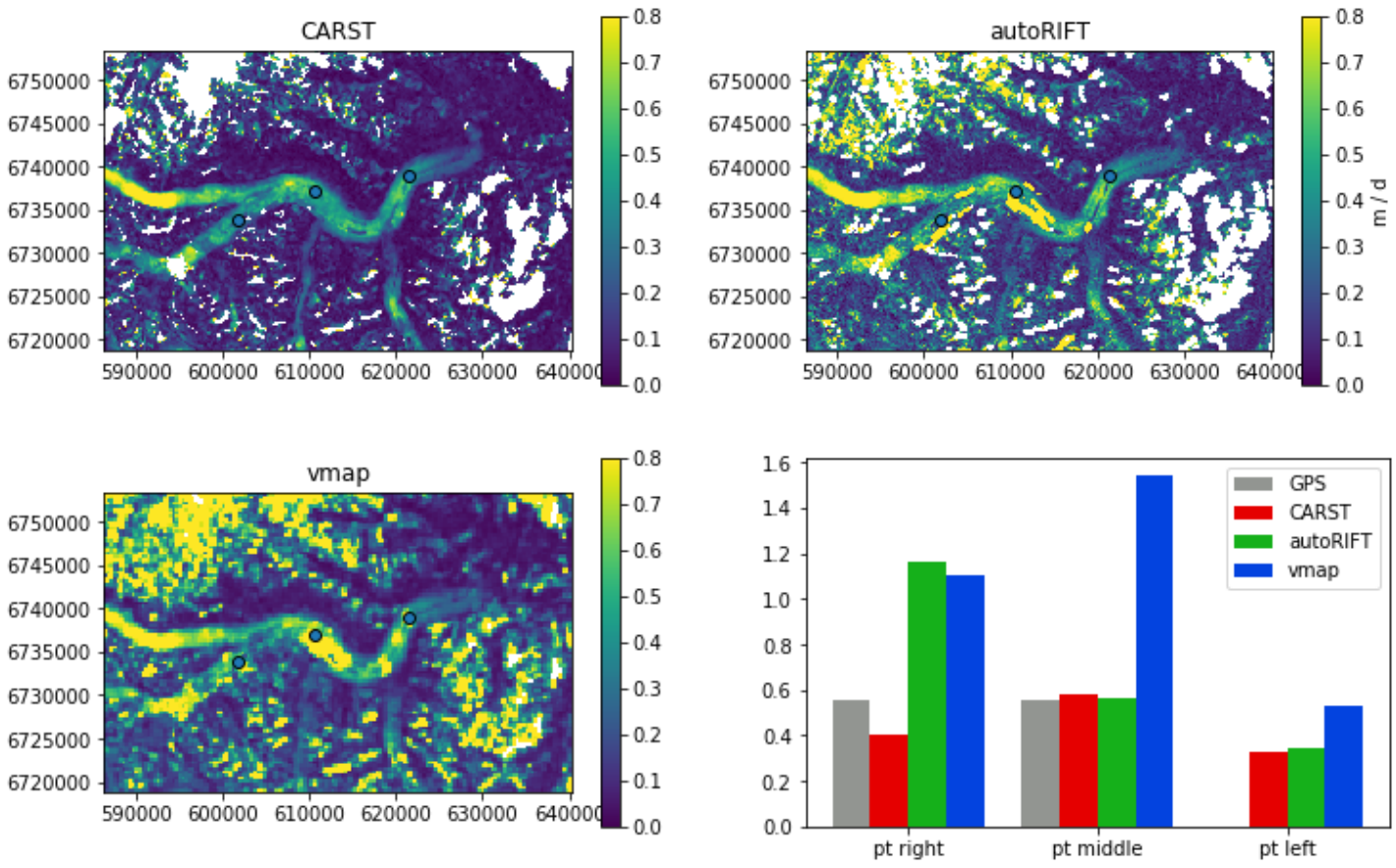
```
folder = '/data/whyj/Projects/Cryosphere/PX_comparison/Yukon/PX/autorift-
CARST_comparison/'

carst_f = folder + 'carst_120m.tif'
autorift_f = folder + 'autorift_120m.tif'
vmap_f = folder + 'vmap.tif'

gps_f = '/data/whyj/Projects/Cryosphere/PX_comparison/Yukon/PX/vmap_Will/Kaskawulsh_2018-
08-18_to_2018-09-03'
```

```
ca = rasterio.open(carst_f)
au = rasterio.open(autorift_f)
vm = rasterio.open(vmap_f)

gps = pd.read_csv(gps_f)
gps = gpd.GeoDataFrame(gps, geometry=gpd.points_from_xy(gps['end_easting'],
gps['end_northing']), crs='EPSG:32607')

gps_xy = list(gps[['end_easting', 'end_northing']].to_records(index=False))
sample_gen = ca.sample(gps_xy)
ca_velo = [float(record) for record in sample_gen]
sample_gen = au.sample(gps_xy)
au_velo = [float(record) for record in sample_gen]
sample_gen = vm.sample(gps_xy)
vm_velo = [float(record) for record in sample_gen]

gps['CARST velo (m/d)'] = ca_velo
gps['autoRIFT velo (m/d)'] = au_velo
gps['vmap velo (m/d)'] = vm_velo

gps
# CHECK THE LAST 5 COLUMNS
```

	Unnamed: 0	date1	date2	start_easting	start_northing	end_easting	end_northing
0	0	2018-08-18	2018-09-03	621368.621431	6.738902e+06	621374.193187	6.738909e+06
1	1	2018-08-18	2018-09-03	610513.759702	6.737085e+06	610521.198198	6.737080e+06
2	2	2018-08-18	2018-09-03	NaN	NaN	601801.697365	6.733765e+06

```
fig=plt.figure(figsize=(12, 12))
ax1=fig.add_subplot(321)
ax2=fig.add_subplot(322, sharex=ax1, sharey=ax1)
ax3=fig.add_subplot(323, sharex=ax1, sharey=ax1)

ax1.set_title('CARST')
ax2.set_title('autoRIFT')
ax3.set_title('vmap')

rio_show(ca, ax=ax1, vmin=0, vmax=0.8)
rio_show(au, ax=ax2, vmin=0, vmax=0.8)
rio_show(vm, ax=ax3, vmin=0, vmax=0.8)
# clim=im.properties()['clim']

gps.plot(ax=ax1, edgecolor='black')
gps.plot(ax=ax2, edgecolor='black')
gps.plot(ax=ax3, edgecolor='black')

norm = mpl.colors.Normalize(vmin=0,vmax=0.8)
sm = plt.cm.ScalarMappable(cmap='viridis', norm=norm)
sm.set_array([])

# print(ax3.collections[0])
# plt.colorbar(ax3.collections[0])
# plt.colorbar(ax=ax3, cax=ax3)
fig.colorbar(sm, ax=ax1)
cent = fig.colorbar(sm, ax=ax2)
fig.colorbar(sm, ax=ax3)

cent.set_label('m / d')

ax4=fig.add_subplot(324)

x_lbl = np.array([0, 1, 2])
gps_ax4 = gps['velocity (m/d)']
ca_ax4 = gps['CARST velo (m/d)']
au_ax4 = gps['autoRIFT velo (m/d)']
vm_ax4 = gps['vmap velo (m/d)']
ax4.bar(x_lbl - 0.3, gps_ax4, color='xkcd:gray', width=0.2, label='GPS')
ax4.bar(x_lbl - 0.1, ca_ax4, color='xkcd:red', width=0.2, label='CARST')
ax4.bar(x_lbl + 0.1, au_ax4, color='xkcd:green', width=0.2, label='autoRIFT')
ax4.bar(x_lbl + 0.3, vm_ax4, color='xkcd:blue', width=0.2, label='vmap')
ax4.set_xticks(x_lbl)
ax4.set_xticklabels(['pt right', 'pt middle', 'pt left'])
ax4.legend()

print('Landsat 8 Feature Tracking: 20180818-20180903')
```

```
Landsat 8 Feature Tracking: 20180818-20180903
```



By Whyjay Zheng, Shashank Bhushan, Max Van Wyk de Vries, Will Kochtitzky, David Shean

Validation #2: Feature tracking over a known, synthetic pixel offset, then calculate the difference between results and the ground truth

Contents

– To be updated in the future work –

This method uses **one source image** from the planned feature tracking pair. A arbitrary, synthetic pixel offset is applied to that image. The original image and the processed image form a pair, and we perform feature tracking using this pair. we calculate the difference between the offsets found by the FT algorithm and the ground truth (which is the arbitrary pixel offset). The statistics of this difference can be compared with the metrics to determine whether these metrics really indicate better FT parameter combinations.

Requirements and Limits

- There are several steps needed to bo done before running FT. Users have to determine a few parameters for creating the synthetic offset field and create the second image.
- Since it does not use the actual results from the planned feature tracking pair, there is a concern that this metric may fail to find the best parameter set for the actual pair when surface conditions change during the pair period (e.g. from snow melt).

Procedure

1. Prepare the source image as a Geotiff file.
2. Determine the parameters for the synthetic offset field and use `gftt.create_synthetic_offset` and `gftt.apply_synthetic_offset` to create pixel-shifted image. See this notebook for details.
3. Perform feature tracking algorithm and collect the resulting velocity field (V_x and V_y) as geotiff files.
4. Use `gftt.syn_shift_errors` to calculate the deviation from the ground truth (the synthetic shift field). See this notebook for details.

Analysis

1. Information, map area, importing modules

- Test area: **Kaskawulsh Glacier**
- Test packages: **CARST and Vmap**
- Test pair: **Landsat 8 20180304**

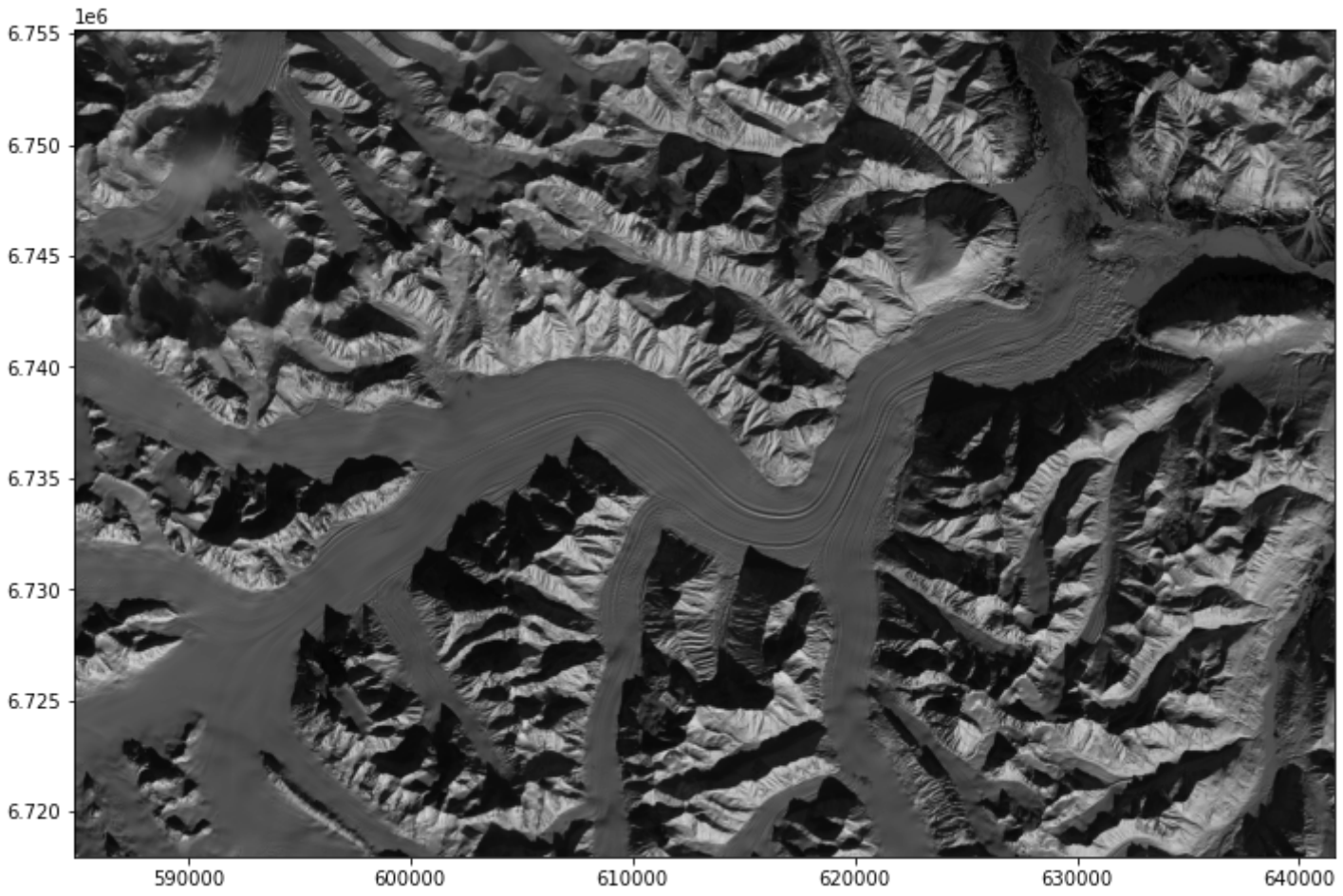
```
# Developer's setting
%load_ext autoreload
%autoreload 2
```

```
import gftt
import rasterio
from rasterio.plot import show
import numpy as np
import matplotlib.pyplot as plt
from pygeotools.lib import warplib
from matplotlib import cm
```

```
# source file
image_geotiff = '../../data/LS8/LC08_L1TP_061018_20180304_20180319_01_T1_B8_s.TIF'
```

```
src = rasterio.open(image_geotiff)
```

```
fig, ax0 = plt.subplots(1, 1, figsize=(12, 8))
show(src, cmap='gray', ax=ax0);
```



2. Create pixel-shifted image

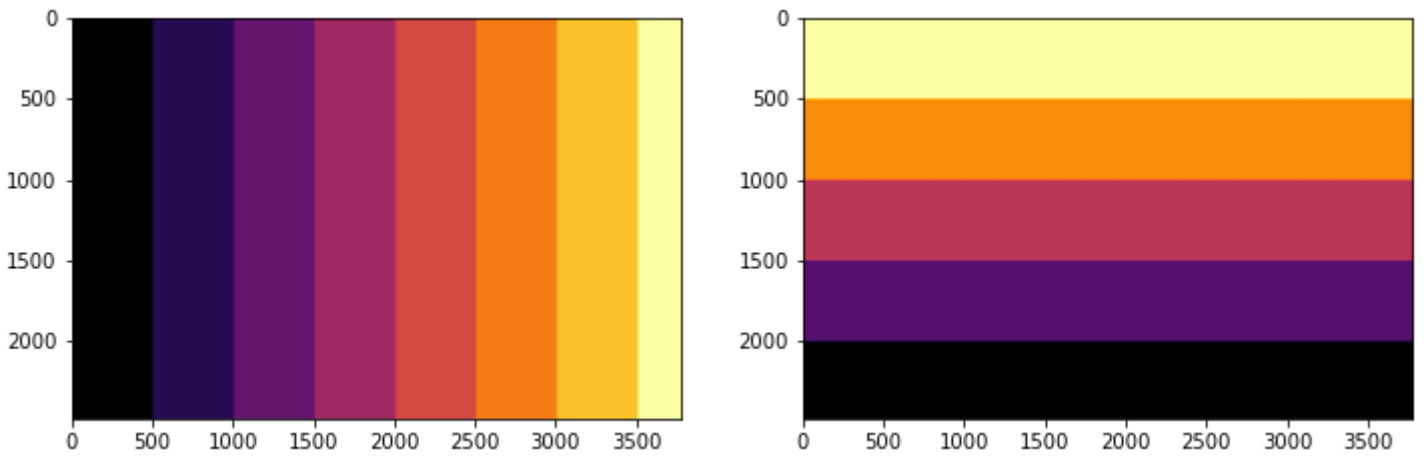
We use the defaul parameters below to create a sub-pixel shift field:

- block size = 500 pixels
- spline order = 1 (linear interpolation)

```
shift_arx, shift_ary = gftt.create_synthetic_offset(image_geotiff)
shift_arx
```

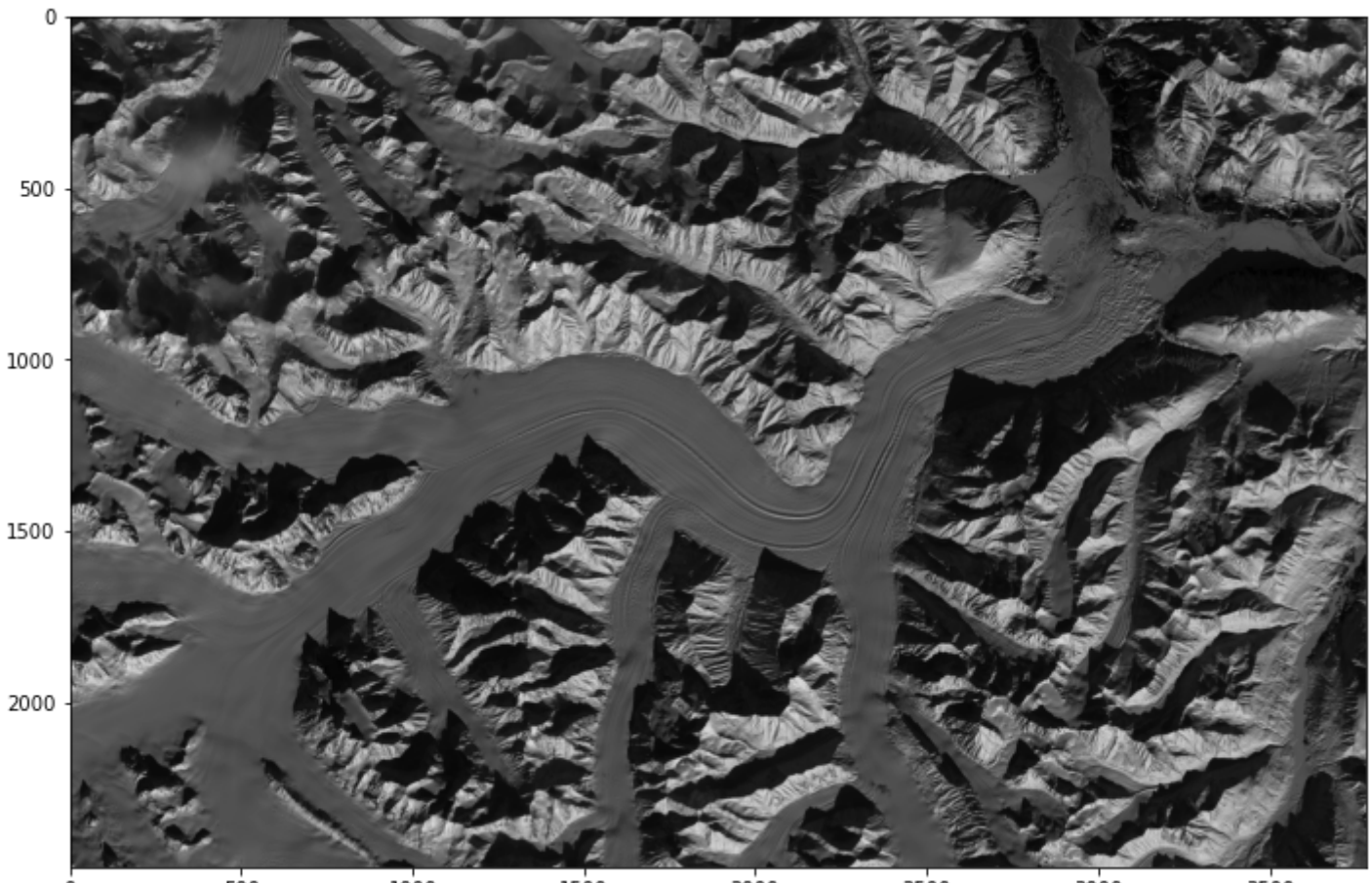
```
array([[0.1, 0.1, 0.1, ..., 0.8, 0.8, 0.8],
       [0.1, 0.1, 0.1, ..., 0.8, 0.8, 0.8],
       [0.1, 0.1, 0.1, ..., 0.8, 0.8, 0.8],
       ...,
       [0.1, 0.1, 0.1, ..., 0.8, 0.8, 0.8],
       [0.1, 0.1, 0.1, ..., 0.8, 0.8, 0.8],
       [0.1, 0.1, 0.1, ..., 0.8, 0.8, 0.8]])
```

```
fig, ax1 = plt.subplots(1, 2, figsize=(12, 4))
ax1[0].imshow(shift_arx, cmap='inferno', vmin = 0.1, vmax = 0.8)
ax1[1].imshow(shift_ary, cmap='inferno', vmin = -0.5, vmax = -0.1);
```




```
shifted_image = gftt.apply_synthetic_offset(image_geotiff, shift_arx, shift_ary)
```

```
fig, ax2 = plt.subplots(1, 1, figsize=(12, 8))
ax2.imshow(shifted_image, cmap='gray');
```



Note that this should look really like the source image as the amount of offset is all within 1 pixel. We can save this image as a geotiff and perform feature tracking using whatever package to be tested.

```
kwds = src.profile
with rasterio.open('shifted_LS8_20180304.tif', 'w', **kwds) as dst:
    dst.write(shifted_image, 1)
```

```
kwds2 = src.profile
kwds2['dtype'] = 'float32'
with rasterio.open('shifted_array_x.tif', 'w', **kwds2) as dst:
    dst.write(shift_arx, 1)
with rasterio.open('shifted_array_y.tif', 'w', **kwds2) as dst:
    dst.write(shift_ary, 1)
```

3. Run feature tracking using selected algorithm and parameter set

The pair to be used:

- `data/LS8/LC08_L1TP_061018_20180304_20180319_01_T1_B8_s.TIF`
- `./shifted_LS8_20180304.tif`

CARST results are stored in `data/synthetic_offset_results_carst`.

4. Check the results

Here are paths to the FT results. We have 4 tests, and each of them uses a different kernel size. (8, 16, 36, and 52 pixels, respectively.)

```
result_x_geotiffs = ['../../data/synthetic_offset_results_carst/pair001_20180304-
20180304syn_kerne8/20180304-20180305_velo-raw_vx.tif',
                     '../../data/synthetic_offset_results_carst/pair001_20180304-
20180304syn_kerne16/20180304-20180305_velo-raw_vx.tif',
                     '../../data/synthetic_offset_results_carst/pair001_20180304-
20180304syn_kerne36/20180304-20180305_velo-raw_vx.tif',
                     '../../data/synthetic_offset_results_carst/pair001_20180304-
20180304syn_kerne52/20180304-20180305_velo-raw_vx.tif']
result_y_geotiffs = ['../../data/synthetic_offset_results_carst/pair001_20180304-
20180304syn_kerne8/20180304-20180305_velo-raw_vy.tif',
                     '../../data/synthetic_offset_results_carst/pair001_20180304-
20180304syn_kerne16/20180304-20180305_velo-raw_vy.tif',
                     '../../data/synthetic_offset_results_carst/pair001_20180304-
20180304syn_kerne36/20180304-20180305_velo-raw_vy.tif',
                     '../../data/synthetic_offset_results_carst/pair001_20180304-
20180304syn_kerne52/20180304-20180305_velo-raw_vy.tif']
```

As we read these files, we also warp the shift field (Section 2) in order to perform the pixel-to-pixel comparison (raster differencing) in the next sction. This step is necessary because a skip of 8 pixels is applied during the FT process.

```
result_x = []
input_x  = []
result_y = []
input_y  = []

for fn in result_x_geotiffs:
    ds_list = warplib.memwarp_multi_fn(['shifted_array_x.tif', fn], res='last',
extent='last', t_srs='last', r='near')
    input_x.append(ds_list[0].ReadAsArray(0))
    result_x.append(ds_list[1].ReadAsArray(0))
for fn in result_y_geotiffs:
    ds_list = warplib.memwarp_multi_fn(['shifted_array_y.tif', fn], res='last',
extent='last', t_srs='last', r='near')
    input_y.append(ds_list[0].ReadAsArray(0))
    result_y.append(ds_list[1].ReadAsArray(0))
```

```
Warping all inputs to the following:
Resolution: 120.0
Extent: [584992.5, 6718102.5, 641392.5, 6755062.5]
Projection: '+proj=utm +zone=7 +datum=WGS84 +units=m +no_defs'
Resampling alg: near

1 of 2: shifted_array_x.tif
nl: 308 ns: 470 res: 120.000
2 of 2: ../../data/synthetic_offset_results_carst/pair001_20180304-
20180304syn_kerne8/20180304-20180305_velo-raw_vx.tif

Warping all inputs to the following:
Resolution: 120.0
Extent: [585052.5, 6718162.5, 641332.5, 6755002.5]
Projection: '+proj=utm +zone=7 +datum=WGS84 +units=m +no_defs'
Resampling alg: near

1 of 2: shifted_array_x.tif
nl: 307 ns: 469 res: 120.000
2 of 2: ../../data/synthetic_offset_results_carst/pair001_20180304-
20180304syn_kerne16/20180304-20180305_velo-raw_vx.tif

Warping all inputs to the following:
Resolution: 120.0
Extent: [585202.5, 6718252.5, 641242.5, 6754852.5]
Projection: '+proj=utm +zone=7 +datum=WGS84 +units=m +no_defs'
Resampling alg: near

1 of 2: shifted_array_x.tif
nl: 305 ns: 467 res: 120.000
2 of 2: ../../data/synthetic_offset_results_carst/pair001_20180304-
20180304syn_kerne36/20180304-20180305_velo-raw_vx.tif

Warping all inputs to the following:
Resolution: 120.0
Extent: [585322.5, 6718372.5, 641122.5, 6754732.5]
Projection: '+proj=utm +zone=7 +datum=WGS84 +units=m +no_defs'
Resampling alg: near

1 of 2: shifted_array_x.tif
nl: 303 ns: 465 res: 120.000
2 of 2: ../../data/synthetic_offset_results_carst/pair001_20180304-
20180304syn_kerne52/20180304-20180305_velo-raw_vx.tif

Warping all inputs to the following:
Resolution: 120.0
Extent: [584992.5, 6718102.5, 641392.5, 6755062.5]
Projection: '+proj=utm +zone=7 +datum=WGS84 +units=m +no_defs'
Resampling alg: near

1 of 2: shifted_array_y.tif
nl: 308 ns: 470 res: 120.000
2 of 2: ../../data/synthetic_offset_results_carst/pair001_20180304-
20180304syn_kerne8/20180304-20180305_velo-raw_vy.tif

Warping all inputs to the following:
Resolution: 120.0
Extent: [585052.5, 6718162.5, 641332.5, 6755002.5]
Projection: '+proj=utm +zone=7 +datum=WGS84 +units=m +no_defs'
Resampling alg: near

1 of 2: shifted_array_y.tif
nl: 307 ns: 469 res: 120.000
2 of 2: ../../data/synthetic_offset_results_carst/pair001_20180304-
20180304syn_kerne16/20180304-20180305_velo-raw_vy.tif

Warping all inputs to the following:
Resolution: 120.0
Extent: [585202.5, 6718252.5, 641242.5, 6754852.5]
Projection: '+proj=utm +zone=7 +datum=WGS84 +units=m +no_defs'
Resampling alg: near

1 of 2: shifted_array_y.tif
nl: 305 ns: 467 res: 120.000
2 of 2: ../../data/synthetic_offset_results_carst/pair001_20180304-
20180304syn_kerne36/20180304-20180305_velo-raw_vy.tif

Warping all inputs to the following:
Resolution: 120.0
Extent: [585322.5, 6718372.5, 641122.5, 6754732.5]
Projection: '+proj=utm +zone=7 +datum=WGS84 +units=m +no_defs'
Resampling alg: near

1 of 2: shifted_array_y.tif
nl: 303 ns: 465 res: 120.000
2 of 2: ../../data/synthetic_offset_results_carst/pair001_20180304-
20180304syn_kerne52/20180304-20180305_velo-raw_vy.tif
```

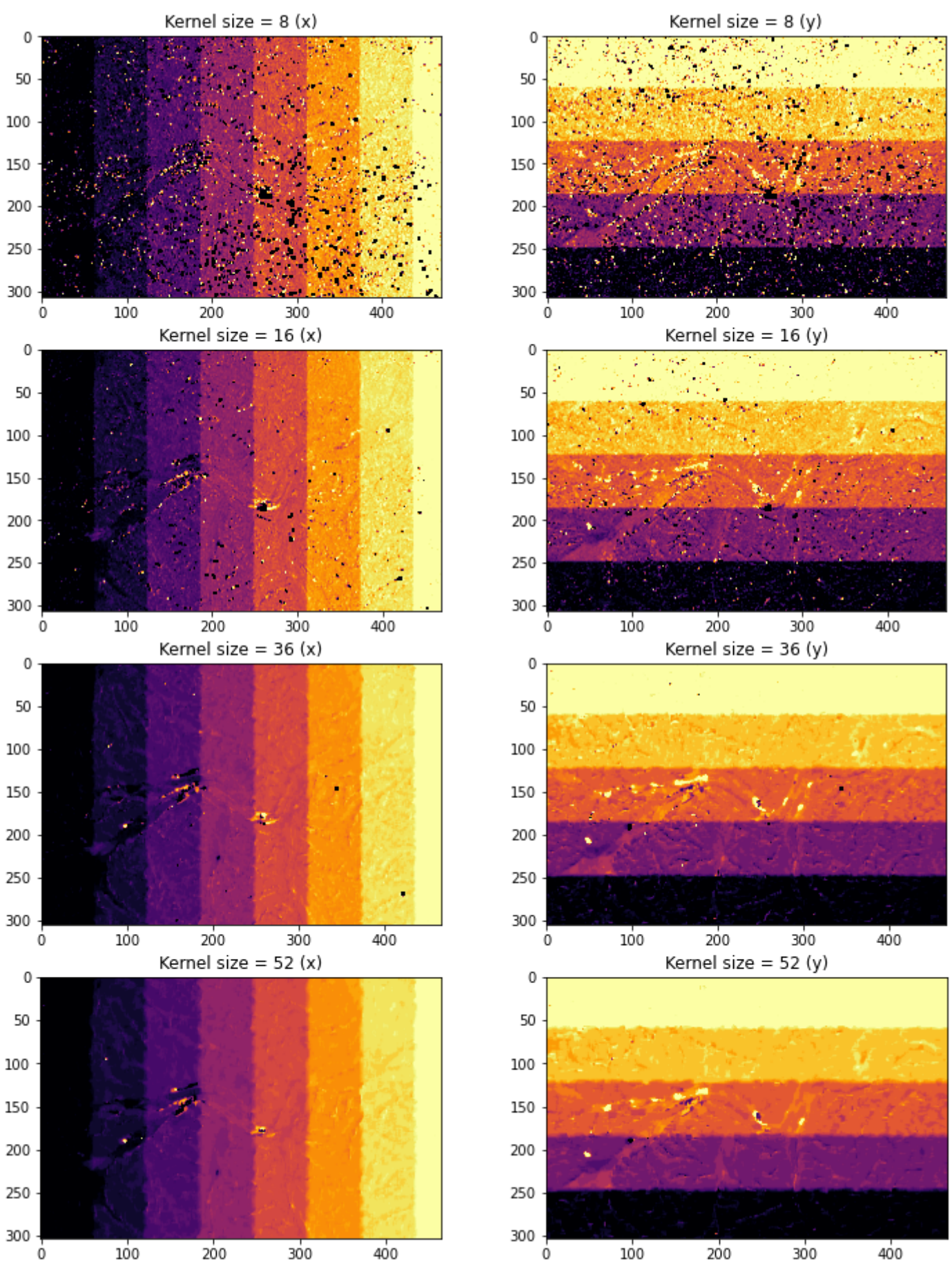
```
20180304syn_kernel52/20180304-20180305_vel0-raw_vy.tif
```

Also, CARST's offset measurements are in meters (or meters/day if date strings are given) instead of pixels, so we have to convert the unit in order to compare with the input shift field.

```
for i in range(len(result_x)):
    nodata_loc = result_x[i] < -9990       # NoDataValue locations
    result_x[i] /= -15                     # for unknown reason, CARST results seem to be
sign-reversed
    result_y[i] /= 15
    result_x[i][nodata_loc] = -9999
    result_y[i][nodata_loc] = -9999
```

This figure shows what the results look like:

```
fig, ax3 = plt.subplots(4, 2, figsize=(12, 16))
title_string = [['Kernel size = 8 (x)', 'Kernel size = 8 (y)'],
                ['Kernel size = 16 (x)', 'Kernel size = 16 (y)'],
                ['Kernel size = 36 (x)', 'Kernel size = 36 (y)'],
                ['Kernel size = 52 (x)', 'Kernel size = 52 (y)']]
for i in range(len(result_x)):
    ax3[i, 0].imshow(result_x[i], vmin = 0.1, vmax = 0.8, cmap='inferno')
    ax3[i, 0].set_title(title_string[i][0])
    ax3[i, 1].imshow(result_y[i], vmin = -0.5, vmax = -0.1, cmap='inferno')
    ax3[i, 1].set_title(title_string[i][1])
```



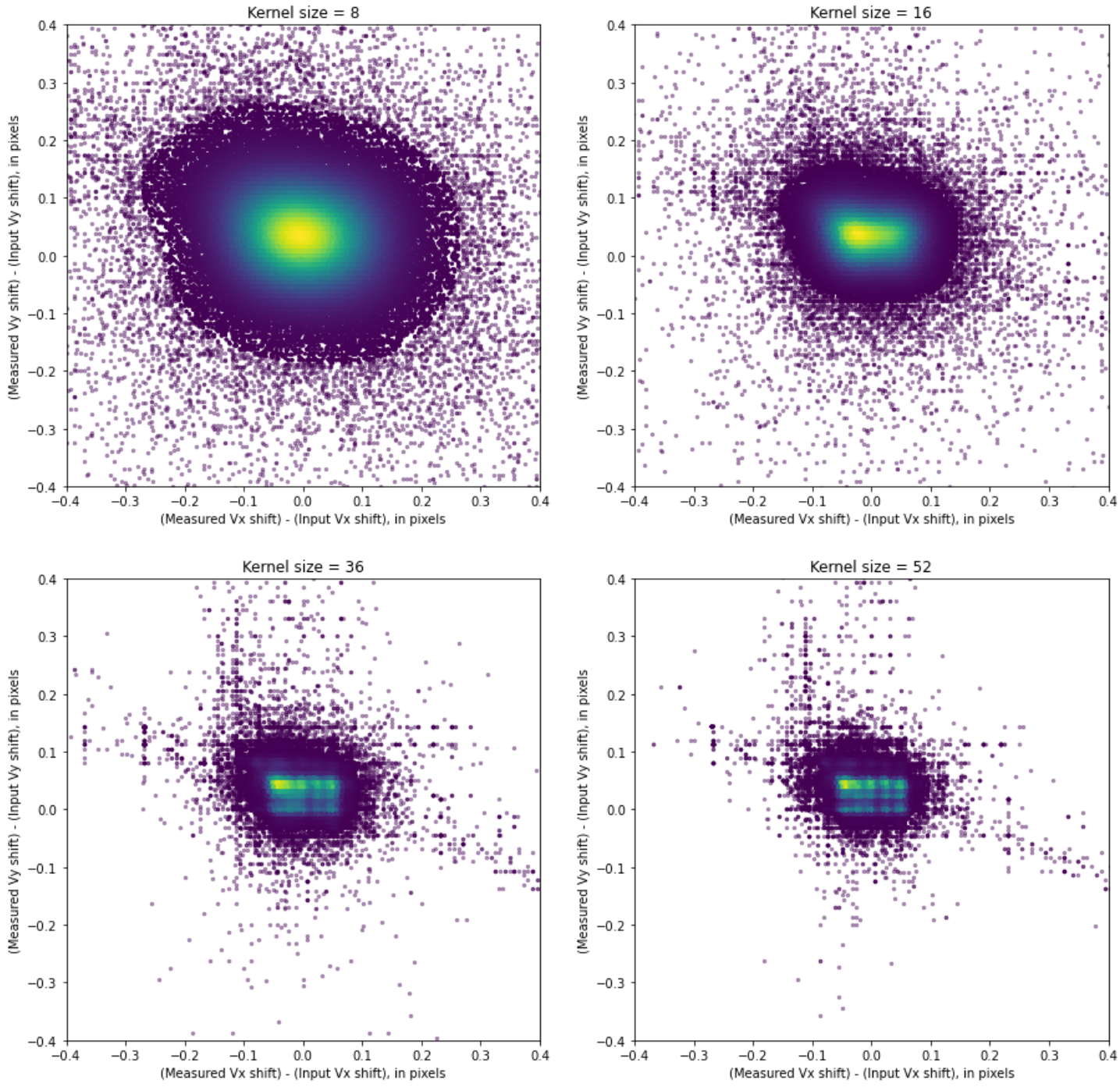
5. Calculate the feature tracking deviation from the ground truth

Now we use `gftt.syn_shift_errors` to calculate the difference between the input shift field and the measured shift field. (This takes some time – ~15-20 minutes)

```
diff_x = []
diff_y = []
z = []
thres_idx = []

for i in range(len(result_x)):
    tmp_x, tmp_y, tmp_z, tmp_idx = gftt.syn_shift_errors(ref_vx=input_x[i],
vx=result_x[i], ref_vy=input_y[i], vy=result_y[i], plot=False)
    diff_x.append(tmp_x)
    diff_y.append(tmp_y)
    z.append(tmp_z)
    thres_idx.append(tmp_idx)
```

```
Start calculating KDE, this may take a while...
KDE Done!
Start calculating KDE, this may take a while...
KDE Done!
Start calculating KDE, this may take a while...
KDE Done!
Start calculating KDE, this may take a while...
KDE Done!
```

```
fig, ax4 = plt.subplots(2, 2, figsize=(15, 15))

viridis = cm.get_cmap('viridis', 12)
pt_style = {'s': 6, 'edgecolor': None}
xbound = (-0.4, 0.4)
ybound = (-0.4, 0.4)
title_string = ['Kernel size = 8',
                'Kernel size = 16',
                'Kernel size = 36',
                'Kernel size = 52']

for i in range(len(result_x)):
    j = i // 2
    k = i % 2
    ax4[j, k].scatter(diff_x[i][thres_idx[i]],  diff_y[i][thres_idx[i]], c=z[i]
[thres_idx[i]], **pt_style)
    ax4[j, k].scatter(diff_x[i][~thres_idx[i]], diff_y[i][~thres_idx[i]],
color=viridis(0), alpha=0.4, **pt_style)
    ax4[j, k].set_xlim(xbound)
    ax4[j, k].set_ylim(ybound)
    ax4[j, k].set_title(title_string[i])
    ax4[j, k].set_xlabel('(Measured Vx shift) - (Input Vx shift), in pixels')
    ax4[j, k].set_ylabel('(Measured Vy shift) - (Input Vy shift), in pixels')
```




```
for i in range(4):
    print(np.sum(thres_idx[i]) / diff_x[i].shape)
```

```
[0.91648297]
[0.94214847]
[0.9614299]
[0.96425707]
```

```
for i in range(4):
    print(title_string[i])
    print('Vx std: {:5f}  pixels'.format(np.std(diff_x[i], ddof=1)))
    print('Vy std: {:5f}  pixels'.format(np.std(diff_y[i], ddof=1)))
```

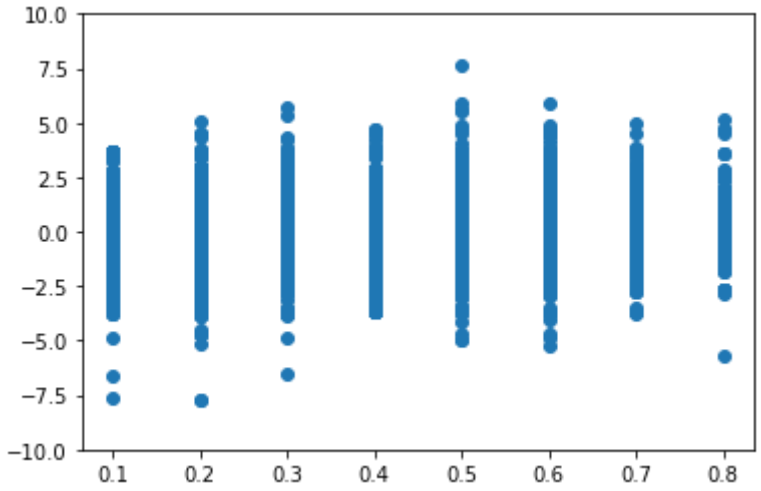
```
Kernel size = 8
Vx std: 0.309122  pixels
Vy std: 0.262493  pixels
Kernel size = 16
Vx std: 0.138230  pixels
Vy std: 0.111232  pixels
Kernel size = 36
Vx std: 0.055448  pixels
Vy std: 0.043165  pixels
Kernel size = 52
Vx std: 0.045505  pixels
Vy std: 0.033561  pixels
```

Done!

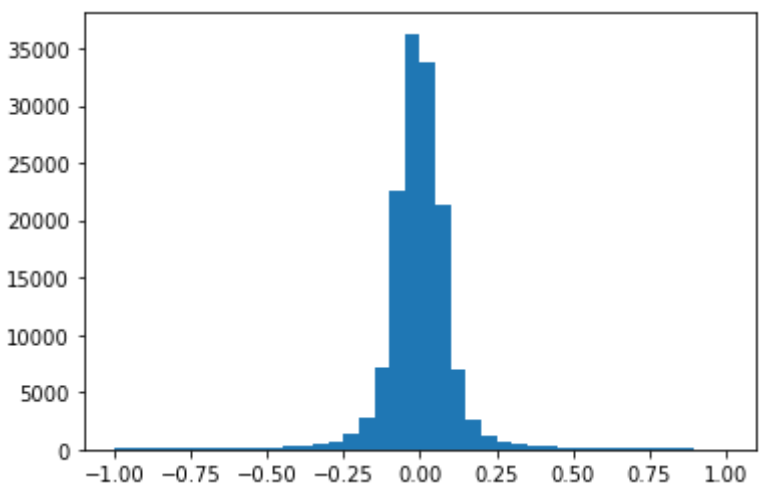
Other plots:

```
plt.scatter(input_x[0].flatten(), result_x[0].flatten())
plt.ylim(-10, 10)
```

```
(-10.0, 10.0)
```




```
plt.hist(diff_x[0], bins=np.linspace(-1, 1, 41));
```




```
plt.hist(diff_y[0], bins=np.linspace(-1, 1, 41));
```