EGU General Assembly 2022

IBS Center for Climate Physics

부산대학교 PUSAN NATIONAL UNIVERSITY

Institute for Basic Science

Scalable Feature Extraction and Tracking (SCAFET)

A general framework for feature extraction from large climate datasets

Arjun Babu Nellikkattil
IBS Center for Climate Physics
Department of Climate System, Pusan National University

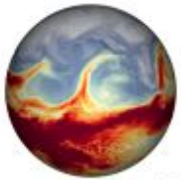
June-Yi Lee[1,2,3], Axel Timmermann[1,4], Sun-Seon Lee[1], Jung-Eun Chu[1]

[1]Center for Climate Physics, Institute for Basic Science, Busan, South Korea
[2]Department of Climate System, Pusan National University, Busan, South Korea
[3]Research Center for Climate Sciences, Pusan National University, Busan, South Korea
[4]Pusan National University, Busan, South Korea

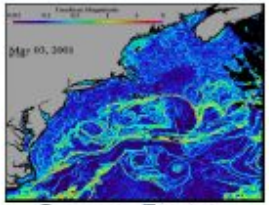
Background & Motivation

Features? Features are phenomena, structures or objects in a dataset, that are of interest for a certain research or engineering problem.

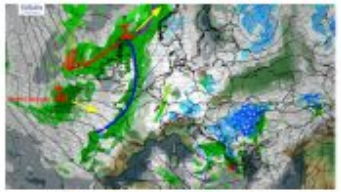
Cyclones
Source: Wikipedia



Atmospheric Rivers
Source: NASA

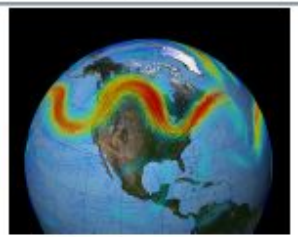


Ocean Fronts
Source: Igor Belkin (2021)

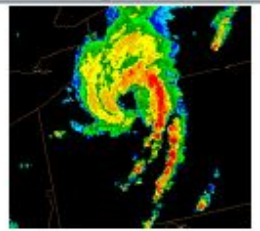


Frontal Systems
Source: Weather Channel

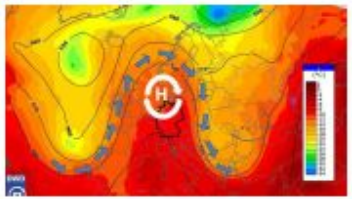
Convective Cells
Source: ESA



Jet Streams
Source: NASA



Squall Lines
Source: Wikipedia



Atmosp. Blocking
Source: IASS

Dust Storms
Source: NASA

Background & Motivation

Features? Features are phenomena, structures or objects in a dataset, that are of interest for a certain research or engineering problem.

- Lack of theoretical qualitative definition
- Defining threshold
  - Dataset, space and time

Mean statistics and Feature characteristics

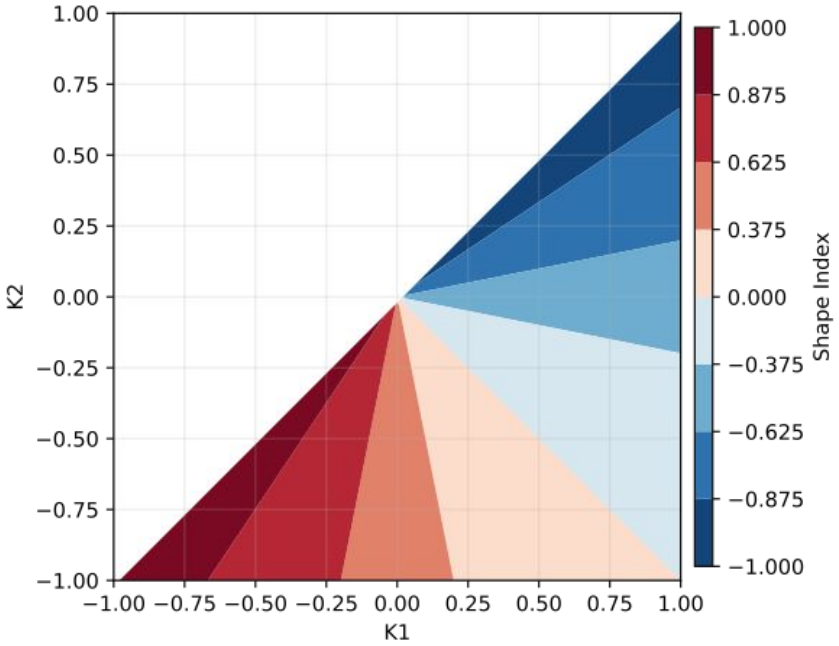
- ❏ The datasets
- ❏ The algorithm

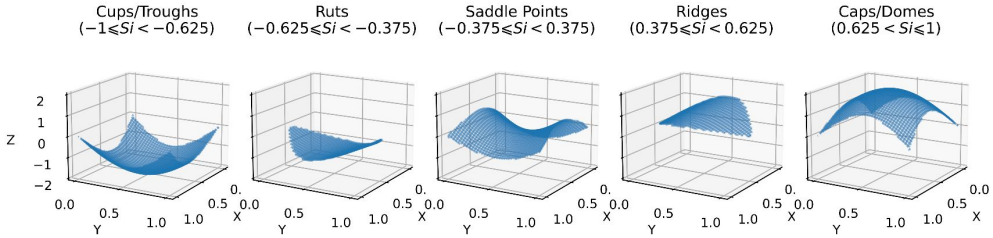
Background & Motivation

1. A scale and mean state independent feature extraction method
2. To extract a variety of features
3. Capable of processing different types of grids
4. Easily Implementable

Shape Index (Si)

To segment the given field into different types of features

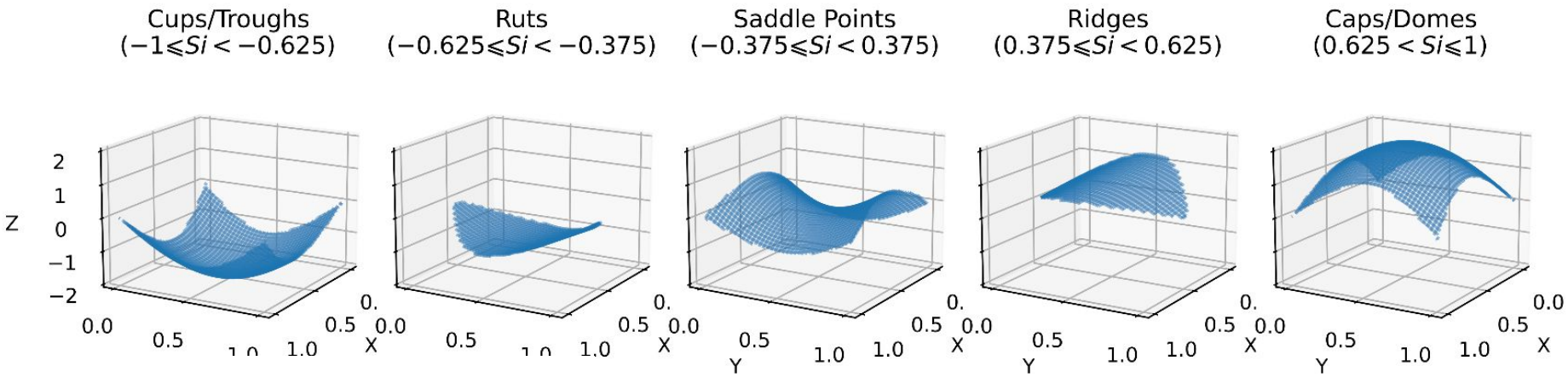




$$Si = \frac{2}{\pi} arctan \frac{k_2 + k_1}{k_2 - k_1} \quad (k_1 \geq k_2)$$

Koenderink, Jan J., and Andrea J. Van Doorn. "Surface shape and curvature scales." *Image and vision computing* 10.8 (1992): 557-564.

Shape Index (Si)



$$H_x = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_1 x_2} & \frac{\partial^2 f}{\partial x_2^2} \end{bmatrix}$$

$$Si = \frac{2}{\pi} arctan \frac{k_2 + k_1}{k_2 - k_1} \quad (k_1 \geq k_2)$$

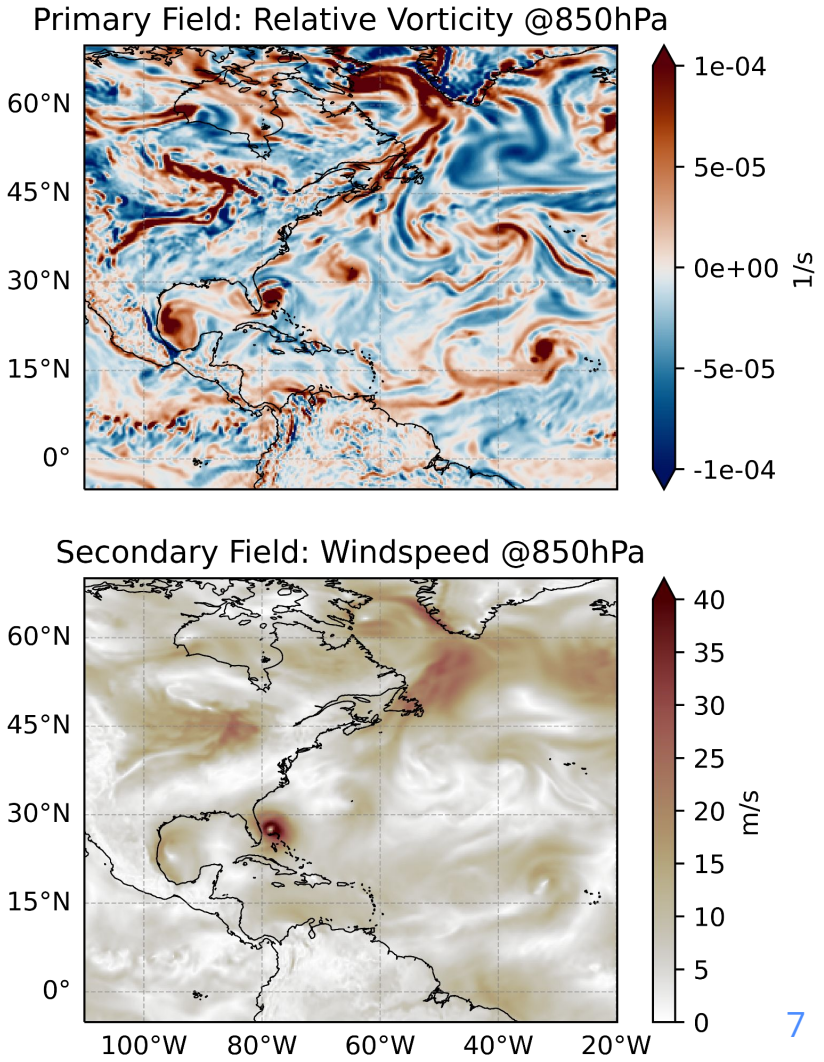
Koenderink, Jan J., and Andrea J. Van Doorn. "Surface shape and curvature scales." *Image and vision computing* 10.8 (1992): 557-564.

Define Object Properties

In [40]:
```
%%time

smooth_scale = 1.5e6
angle_threshold = 45
shape_index = [0.625,1]
min_length = 20e3
min_area = 1e11
min_duration = 6
max_distance_per_tstep = 1000e3
shape_eccentricity = [0.0,1.0]
lat_mask = [-0,0]
lon_mask = [360,0]

properties = obp.object_properties2D(ivt_grid_area,ivt_land,min_length,min_area,\
                    smooth_scale,angle_threshold,min_duration,max_distance_per_tstep,\
                    shape_index,shape_eccentricity,\
                    lon_mask,lat_mask)
```

```
CPU times: user 123 ms, sys: 18.6 ms, total: 141 ms
Wall time: 180 ms
```

- smooth_scale = Approx. spatial scale of the feature
- shape_index > 0.625 (Caps and Domes)
- minimum_area = To remove small objects
- minimum_duration = Minimum no.of timesteps
- shape_eccentricity = Eccentricity of the object

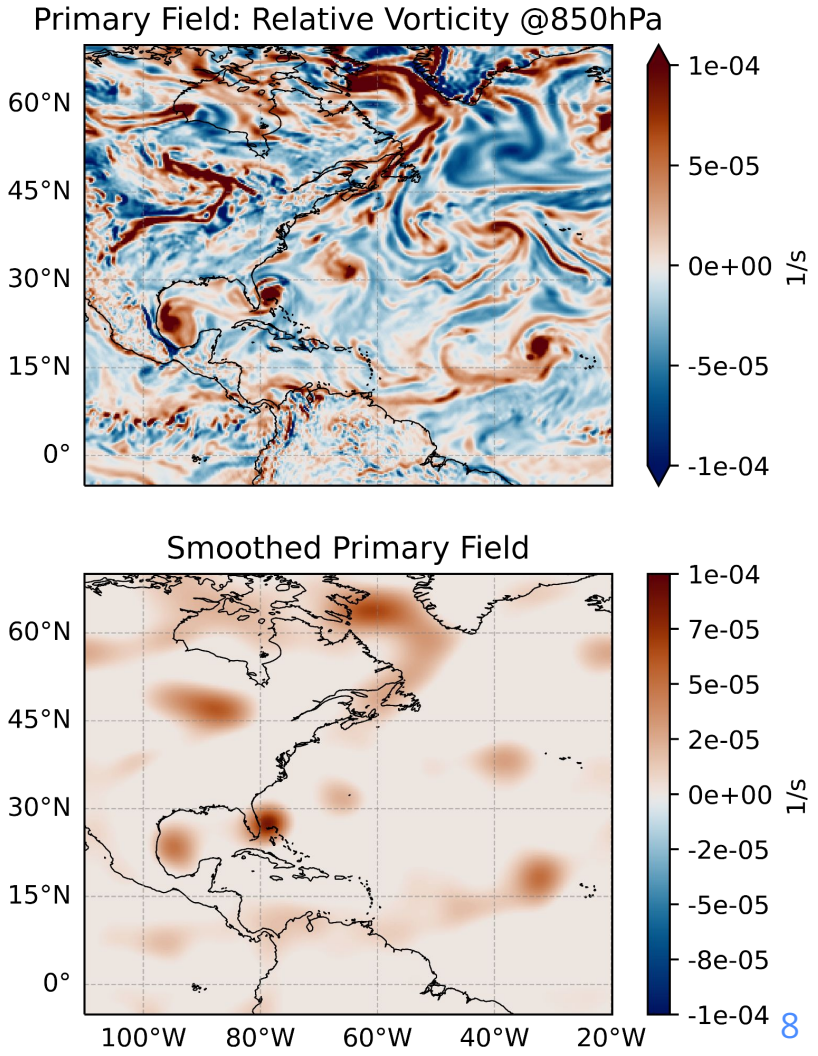


Grid Aware Spatial Smoothing

In [12]:
```
%%time
start = time.time()
rdetect = rd.ridgeDetector(cycs)
vor = rdetect.apply_smoother(cycs,properties)
vor = vor.rename({'rv':'mag'})
cyc = cycs.rename({'rv':'mag'})
cyc_us = cyc.where((cyc.mag>0)).fillna(0)
cyc_sm = vor.where((vor.mag>0)).fillna(0)
print('Finished smoothing in {} seconds'.format(time.time() - start))
```

```
Finished smoothing in 0.06625509262084961 seconds
CPU times: user 66.6 ms, sys: 74 µs, total: 66.7 ms
Wall time: 66.5 ms
```

Apply smoothing to remove variability smaller than the scales you require



Feature Extraction

Extract Caps and Domes

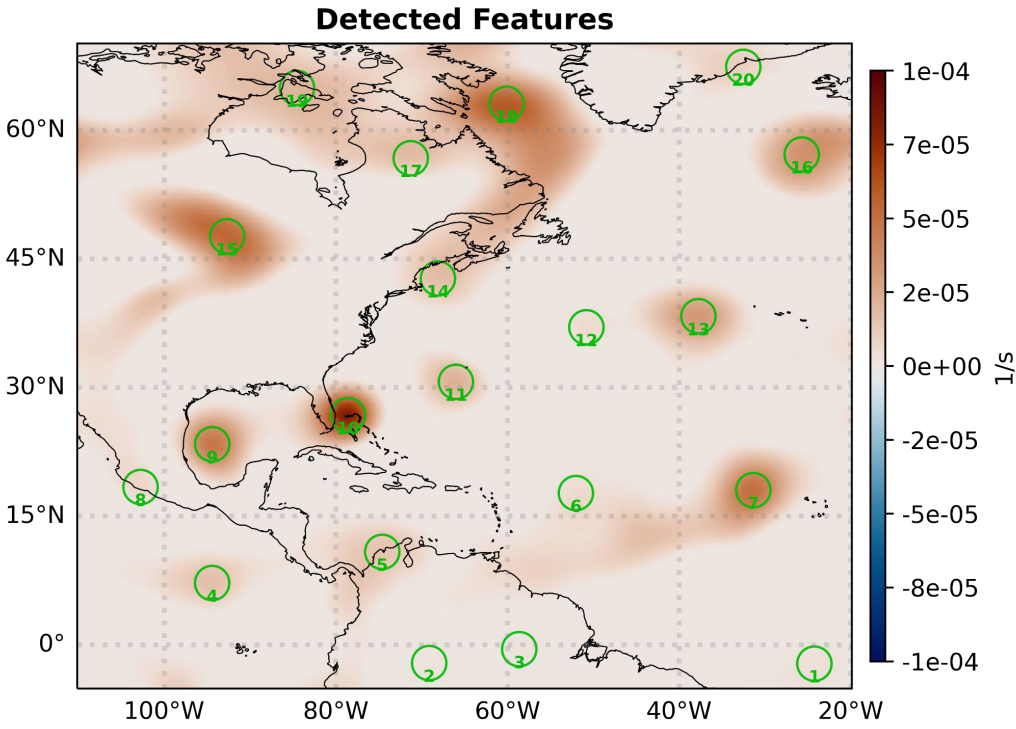
In [13]:
```
%%time
# Detect Ridges
start = time.time()
ridges = rdetect.apply_ridge_detection(cyc_sm,properties)
print('Finished ridge detection in {} seconds'.format(time.time() - start))
```

```
Finished ridge detection in 94.03190279006958 seconds
CPU times: user 1min 4s, sys: 1min 1s, total: 2min 5s
Wall time: 1min 34s
```

Further Optional Filtering

In [14]:
```
%%time
start = time.time()

props_mag = xr.concat([ws.expand_dims('Channel'),\
        cyc_us.mag.expand_dims('Channel')], dim='Channel')
props_mag = props_mag.to_dataset(name='mag')

# Filtering
obfilter = obf.filterObjects(ridges)
filtered = obfilter.apply_filter(ridges.load(),\
        props_mag.load(),['max_intensity','mean_intensity'],
        [0,0],properties,'ridges')
print('Finished Filtering in {} seconds'.format(time.time() - start))
```

```
Finished Filtering in 51.06637668609619 seconds
CPU times: user 20.9 s, sys: 1min 59s, total: 2min 20s
Wall time: 51.1 s
```



Tracking

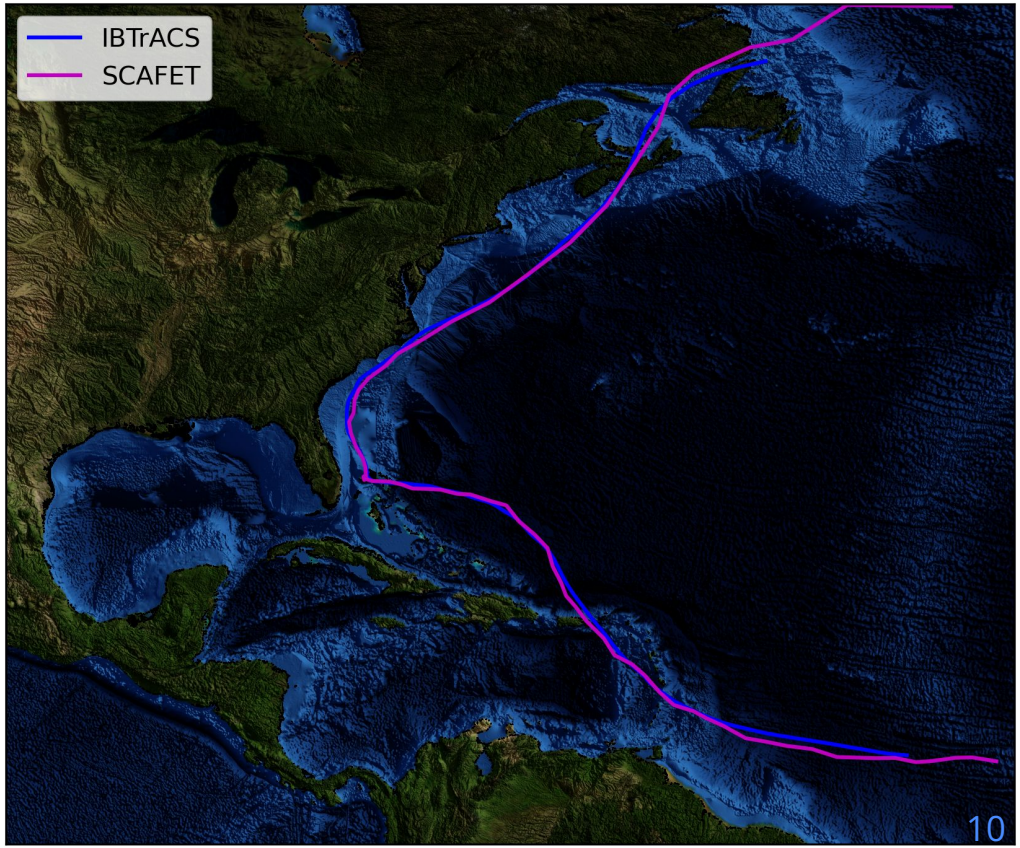
In [17]:
```
%%time

start = time.time()
cycprops = filtered[0].assign_coords(time=pd.to_datetime(filtered[0].time))
cycobjs= filtered[1].assign_coords(time=pd.to_datetime(filtered[1].time))

properties.obj['Min_Duration']=9
properties.obj['Max_Distance']=500

# latlon = ['mx_lat1','mx_lon1']
latlon = ['wclat','wclon']

tracker = obt.Tracker(latlon,properties)
tracked = tracker.apply_tracking(cycprops,cycobjs)
print('Finished tracking in {} seconds'.format(time.time() - start))
```

```
100%|██████████| 83/83 [00:07<00:00, 10.48it/s]
```

```
Finished tracking in 8.633085250854492 seconds
CPU times: user 8.55 s, sys: 88 ms, total: 8.64 s
Wall time: 8.63 s
```

Tracking based on centroid of the detected object

- Minimum Duration = 54 hours



Summary

- Types of features
  - Tropical & Extratropical cyclones, Atmospheric rivers, SST fronts, Atmospheric Blocking, Jetstreams
- Types of grids
  - Various rectilinear grids
  - POP grid
- Easily Implementable python package

Summary

- Types of features
  - Tropical & Extratropical cyclones, Atmospheric rivers, SST fronts, Atmospheric Blocking, Jetstreams
- Types of grids
  - Various rectilinear grids
  - POP grid
- Easily Implementable python package

Arjun Babu Nellikkattil
IBS Center for Climate Physics
Geumjeong-gu, Busan 46241,
Republic of Korea
arjunbabun@pusan.ac.kr
arjunbabun11@gmail.com