FacetFlowNetwork

Flow accumulation on TINs of point-cloud data by facet flow networks (FFNs).

Install

The core of this module is written in C (see main.c). It is wrapped by Cython into a Python module, i.e., you will need Python, Cython and a C compiler.

```
git clone https://github.com/Rheinwalt/FacetFlowNetwork.git
cd FacetFlowNetwork
sudo python setup.py install
```

Usage

This module has one class called *ffn*, and all functionality is structured inside this class by class methods.

```
from FacetFlowNetwork import ffn
help(ffn)
```

Tutorials

Synthetic point cloud for a Gaussian hill

First, a tutorial on synthetic point-cloud data of a Gaussian hill surface. It covers the generation of 1000 points on a 1 by 1 meter region of interest, FFN construction, specific catchment area (SCA) estimation, visualization, input / output of FFNs, and export to LAS files. We recommend displaz as a LAS file viewer.

```
import numpy as np
from matplotlib import pyplot as pl
from FacetFlowNetwork import ffn

def GaussianHill(n = 1000):
    """
    Create a Gaussian hill sampling point cloud with n points.
    """
    x = np.random.random(n)
    y = np.random.random(n)
    z = np.exp(-x*x-y*y)
    return (x, y, z)

# construct FFN from a Gaussian hill
x, y, z = GaussianHill()
G = ffn(x, y, z)

```

```
# visualize the specific catchment area (SCA) for each facet of the FFN
pl.title('Gaussian hill FFN SCA estimate')
pl.tripcolor(x, y, G.tri, facecolors = G.sca(),
        vmax = 1, cmap = pl.cm.viridis_r)
cb = pl.colorbar()
cb.set_label('SCA [m]')
pl.xlabel('x [m]')
pl.ylabel('y [m]')
pl.gca().set_aspect('equal')
pl.savefig('Gauss.pdf')
```

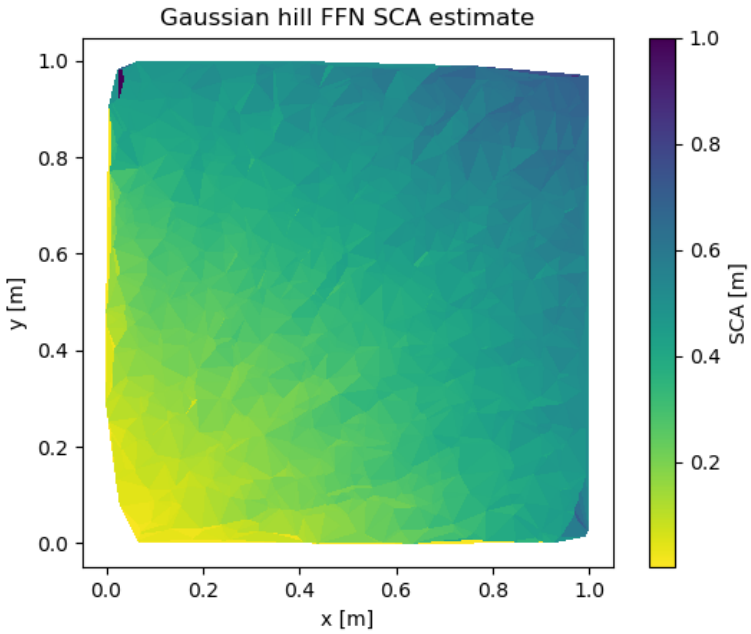

Figure 1: Gaussian hill FFN SCA

```

# store FFN to disk
G.save('Gauss.hdf')

# load FFN from disk
F = ffn(fname = 'Gauss.hdf')

# compare the differences, should be zero since both are identical
dsca = G.sca() - F.sca()
print('Sum of deviations: %f' % np.sum(dsca))
```

```
del F

# export FFN SCA to LAS file
rgbc = G.sca()
pnts = G.facet_centroids()
rgbc[rgbc > 1] = 1
G.export('Gauss.las', rgbc, pnts)

# alternatively, average FFN SCA to the original point cloud and
# export that to a LAS file
rgbc = G.fatp(G.sca())
rgbc[rgbc > 1] = 1
G.export('Gauss_fatp.las', rgbc)
```

Lidar point cloud

Second, a tutorial with lidar point cloud data from Opentopography. Here, we use the alluvial fan lidar point cloud from opentopography.org/learn/lidarlandforms. Due to overlapping flight lines, measurement accuracy and LAS file precision, points might be very close to each other or even overlapping in terms of their xy coordinates. This makes triangulation impossible. Therefore, we remove points that are too close to points closest to the median elevation in a given neighborhood (r = 10cm). This thinning is done as part of our data loading function (see function data and thinning):

```
import numpy as np
from matplotlib import pyplot as pl
from FacetFlowNetwork import ffn

def thinning(x, y, z, r):
    from scipy.spatial import cKDTree as kdtree

    b = np.ones(len(x), dtype = 'bool')
    tree = kdtree(np.transpose((x, y)))
    lsts = tree.query_ball_tree(tree, r = r)
    for l in lsts:
        if len(l) > 1:
            la = np.array(l)
            dz = np.abs(z[la] - np.median(z[la]))
            b[la] = False
            b[la[np.argmin(dz)]] = True

    print('keeping %.3f %%' % (100.*len(b[b])/len(b)))
    return (x[b], y[b], z[b])

def data(fname):
```

```
    from laspy.file import File

    print('loading %s ..' % fname)
    f = File(fname, mode = 'r')
    x = f.x
    y = f.y
    z = f.z
    f.close()
    print('we have %.2e points' % len(x))

    print('add low noise ..')
    x += np.random.random(len(x)) / 1000.0
    y += np.random.random(len(x)) / 1000.0
    z += np.random.random(len(x)) / 1000.0

    print('remove points that are closer than 10cm ..')
    x, y, z = thinning(x, y, z, 0.1)
    print('we have %.2e points' % len(x))
    return (x, y, z)

x, y, z = data('Alluvial_fan.laz')
G = ffn(x, y, z)
G.save('Alluvial_fan.hdf')
```

The stored FFN uses more than 2 GB of storage so it is not uploaded to this repository. Computation took a couple of minutes on a desktop computer and most RAM was used by Scipy Delaunay triangulation.

We can now retrieve the FFN from disk and compute SCA. Further we will try out a few visualizations:

```
import numpy as np
from matplotlib import pyplot as pl
from matplotlib.colors import LogNorm
import matplotlib.tri as mtri
from FacetFlowNetwork import ffn

# load FFN from disk
G = ffn(fname = 'Alluvial_fan.hdf')

# LAS export with average SCA at points
var = np.log10(G.fatp(G.sca()))
G.export('Alluvial_fan_sca.las', var)

# LAS export with only the upper quartile SCA
pts = np.transpose((G.x, G.y, G.z))
```

```
sel = var > np.percentile(var, 75)
pts = pts[sel]
var = var[sel]
G.export('Alluvial_fan_sca_p75.las', var, pts)

# detailed map view of SCA around xc, yc
xc, yc = 451708, 4060410
ww = 50
sca = G.sca()
pts = G.facet_centroids()
xf, yf = pts[:, 0], pts[:, 1]
sel = (xc-ww <= xf)*(xf < xc+ww)*(yc-ww <= yf)*(yf < yc+ww)
tri, sca = G.tri[sel], sca[sel]

pl.figure(1, (8, 6))
pl.title('Alluvial fan FFN SCA estimate')
pl.tripcolor(G.x-xc+ww, G.y-yc+ww, tri, facecolors = sca,
             cmap = pl.cm.magma_r,
             norm = LogNorm(vmin = 0.1, vmax = 1e5))
cb = pl.colorbar()
cb.set_label('SCA [m]')
pl.xlim((0, ww+ww))
pl.ylim((0, ww+ww))
pl.xlabel('x - %i [m]' % (xc-ww))
pl.ylabel('y - %i [m]' % (yc-ww))
pl.gca().set_aspect('equal')
pl.savefig('Alluvial_fan_sca.pdf')
```

```
# detailed map view of SCA with gouraud shading
sca = G.fatp(G.sca())
x, y = G.x, G.y
sel = (xc-ww <= x)*(x < xc+ww)*(yc-ww <= y)*(y < yc+ww)
x, y, sca = x[sel]-xc+ww, y[sel]-yc+ww, sca[sel]
tri = mtri.Triangulation(x, y)

pl.close('all')
pl.figure(1, (8, 6))
pl.title('Alluvial fan FFN SCA estimate')
pl.tripcolor(tri, sca, shading = 'gouraud',
             cmap = pl.cm.magma_r,
             norm = LogNorm(vmin = 0.1, vmax = 1e5))
cb = pl.colorbar()
cb.set_label('SCA [m]')
pl.xlim((0, ww+ww))
pl.ylim((0, ww+ww))
```

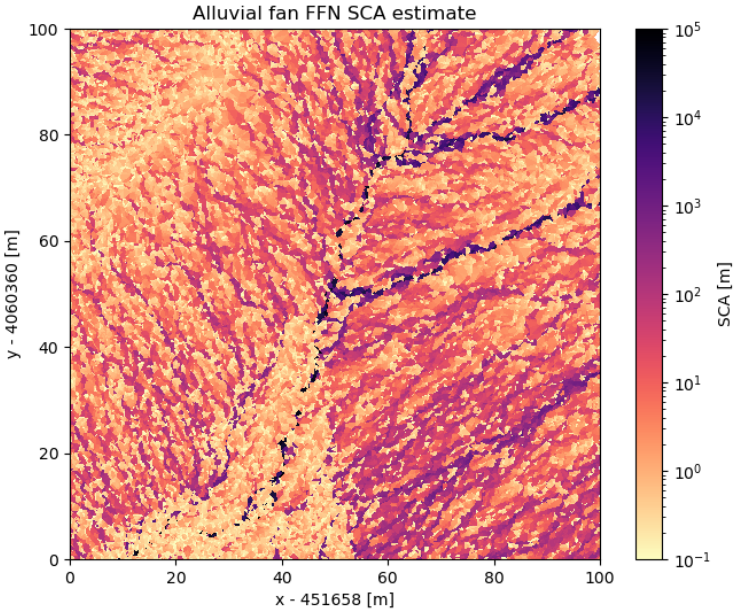

Figure 2: Alluvial fan point cloud FFN SCA estimate

```
pl.xlabel('x - %i [m]' % (xc-ww))
pl.ylabel('y - %i [m]' % (yc-ww))
pl.gca().set_aspect('equal')
pl.savefig('Alluvial_fan_sca_gouraud.pdf')
```

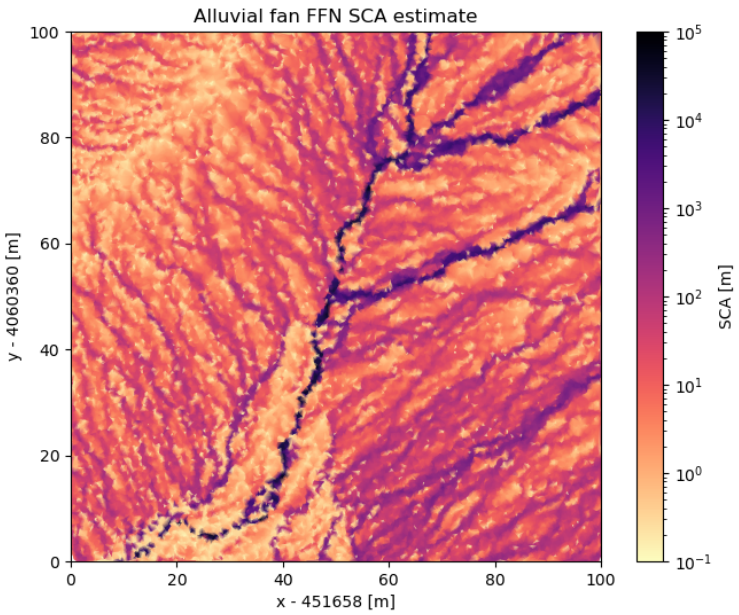

Figure 3: Alluvial fan point cloud FFN SCA estimate

Gridded digital elevation model (DEM)

There is a second class similar to *ffn* that reads a DEM instead of a point cloud:

```
from FacetFlowNetwork import demffn
```

Generating a simple Gaussian hill DEM,

```
pixelwidth = 0.01

# grid axes
xr = np.arange(-1.5, 1.5, pixelwidth)
yr = np.arange(-1.3, 1.3, pixelwidth)

# (x, y) coordinates for grid cells
x, y = np.meshgrid(xr, yr)

```

```
# Gaussian hill DEM
r = np.sqrt(x*x + y*y)
dem = np.exp(-r*r)
```

we can compute the FFN for that DEM with the corresponding class:

```
F = demffn(dem, pixelwidth)
```

We can visualize the SCA for that FFN as a raster with imshow:

```
sca = f.fatp(f.sca())
sca.shape = dem.shape

pl.figure(1, (19.2, 10.8))
im = pl.imshow(sca, origin = 'lower',
        interpolation = 'none', extent = [0,1.5,0,1.3],
        cmap = pl.cm.viridis_r,
        vmin = 0, vmax = tsca.max())
cb = pl.colorbar()
cb.set_label('SCA [m]')
pl.show()
```

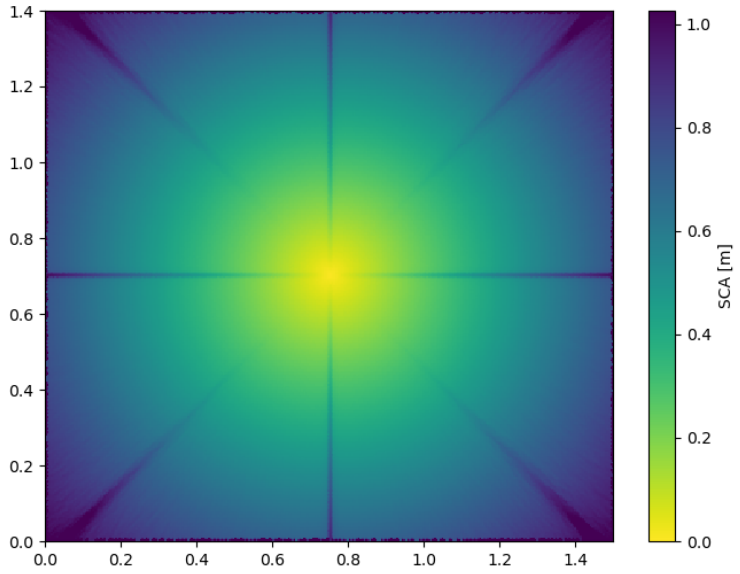

Figure 4: Gaussian hill DEM FFN SCA

The grid effects can also be quantified by the relative differences between the numerical SCA estimates and the theoretical SCA for a Gaussian hill:

```
# theoretical SCA
tsca = r / 2.0

pl.figure(1, (19.2, 10.8))
im = pl.imshow(100*(sca-tsca)/tsca, origin = 'lower',
        interpolation = 'none', extent = [0,1.5,0,1.3],
        cmap = pl.cm.bwr,
        vmin = -25, vmax = 25)
cb = pl.colorbar()
cb.set_label(r'Relative SCA error [%]')
pl.show()
```

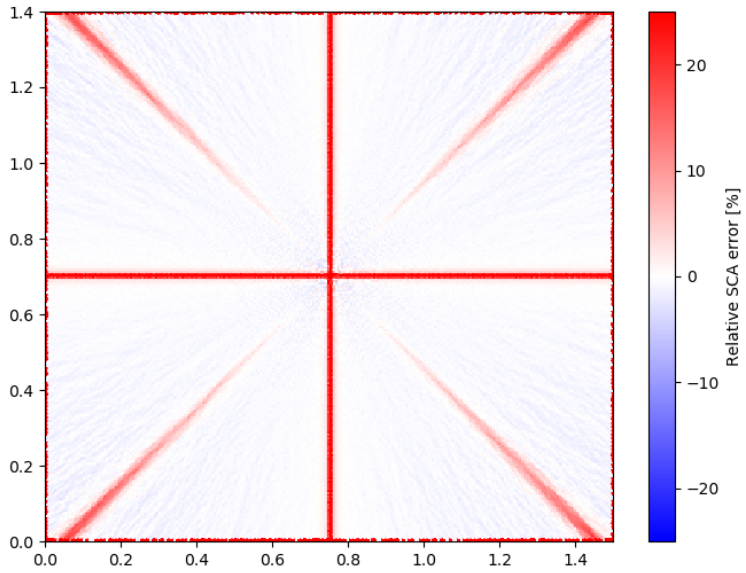

Figure 5: Gaussian hill DEM FFN SCA relative error

Bugs

The C routines inside this module might crash for large point clouds if the stack size on your system is too small. In that case it helps to have an unlimited stack for your session:

```
ulimit -s unlimited
```

Publication

This software is associated to *A network-based flow accumulation algorithm for point clouds: Facet-Flow Networks (FFN)*, Journal of Geophysical Research: Earth Surface, 2019 (10.1029/2018JF004827).