Alpha Shape Toolbox Documentation

Release 1.3.1

Kenneth E. Bellock

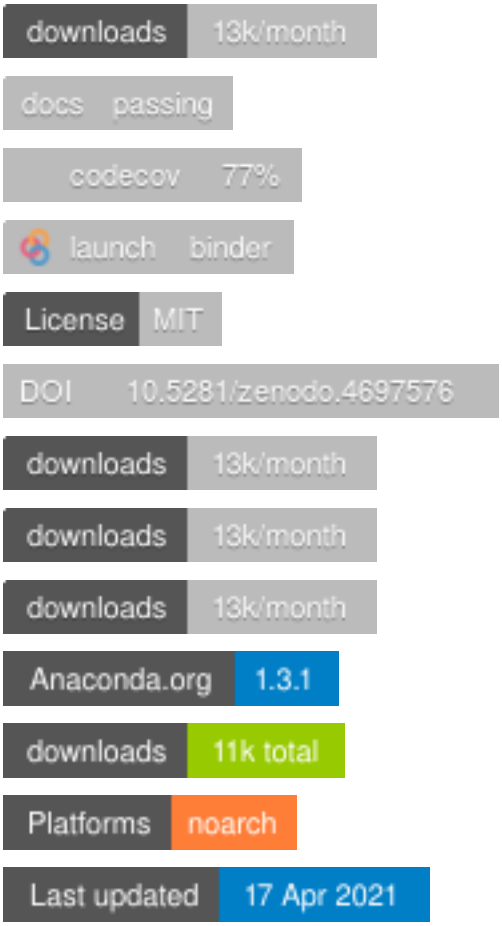
Mar 02, 2022

CONTENTS:

CHAPTER

ONE

ALPHA SHAPE TOOLBOX



Toolbox for generating n-dimensional alpha shapes.

Alpha shapes are often used to generalize bounding polygons containing sets of points. The alpha parameter is defined as the value `a`, such that an edge of a disk of radius 1/`a` can be drawn between any two edge members of a set of points and still contain all the points. The convex hull, a shape resembling what you would see if you wrapped a rubber band around pegs at all the data points, is an alpha shape where the alpha parameter is equal to zero. In this toolbox we will be generating alpha complexes, which are closely related to alpha shapes, but which consist of straight lines between the edge points instead of arcs of circles.

https://en.wikipedia.org/wiki/Alpha_shape

https://en.wikipedia.org/wiki/Convex_hull

Creating alpha shapes around sets of points usually requires a visually interactive step where the alpha parameter for a concave hull is determined by iterating over or bisecting values to approach a best fit. The alpha shape toolbox provides workflows to shorten the development loop on this manual process, or to bypass it completely by solving for an alpha shape with particular characteristics. A python API is provided to aid in the scripted generation of alpha shapes. A console application is also provided as an example usage of the alpha shape toolbox, and to facilitate generation of alpha shapes from the command line.

- Free software: MIT license
- Documentation: https://alphashape.readthedocs.io.

1.1 Features

1.1.1 Import Dependencies

```
import os
import sys
import pandas as pd
import numpy as np
from descartes import PolygonPatch
import matplotlib.pyplot as plt
sys.path.insert(0, os.path.dirname(os.getcwd()))
import alphashape
```

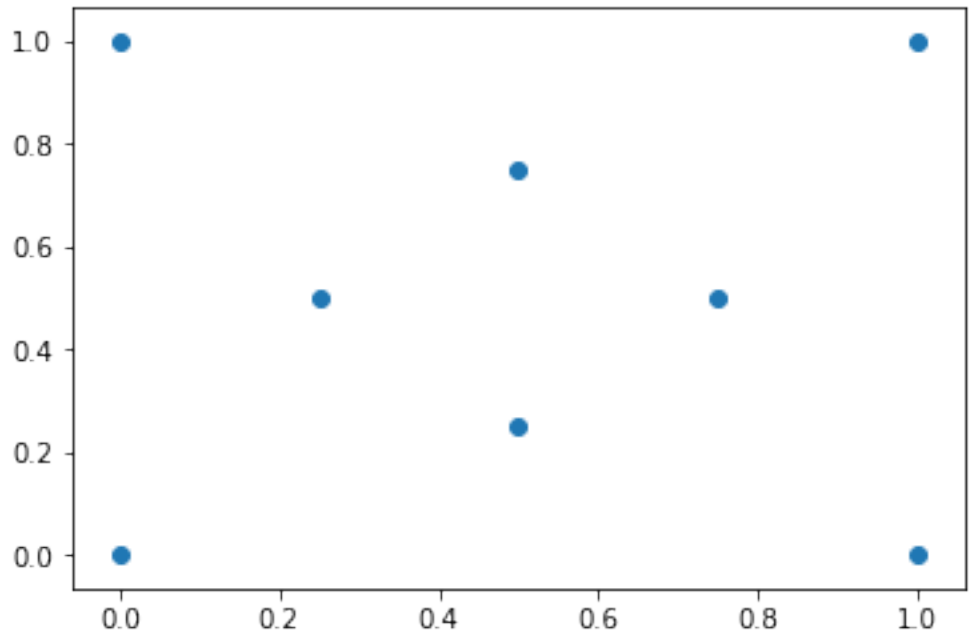
1.1.2 2 Dimensional Example

Define a set of points

```
points_2d = [(0., 0.), (0., 1.), (1., 1.), (1., 0.),
             (0.5, 0.25), (0.5, 0.75), (0.25, 0.5), (0.75, 0.5)]
```

Visualize Test Coordinates

```
fig, ax = plt.subplots()
ax.scatter(*zip(*points_2d))
plt.show()
```



Generate an Alpha Shape ($alpha=0.0$) (Convex Hull)

Every convex hull is an alpha shape, but not every alpha shape is a convex hull. When the `alphashape` function is called with an alpha parameter of 0, a convex hull will always be returned.

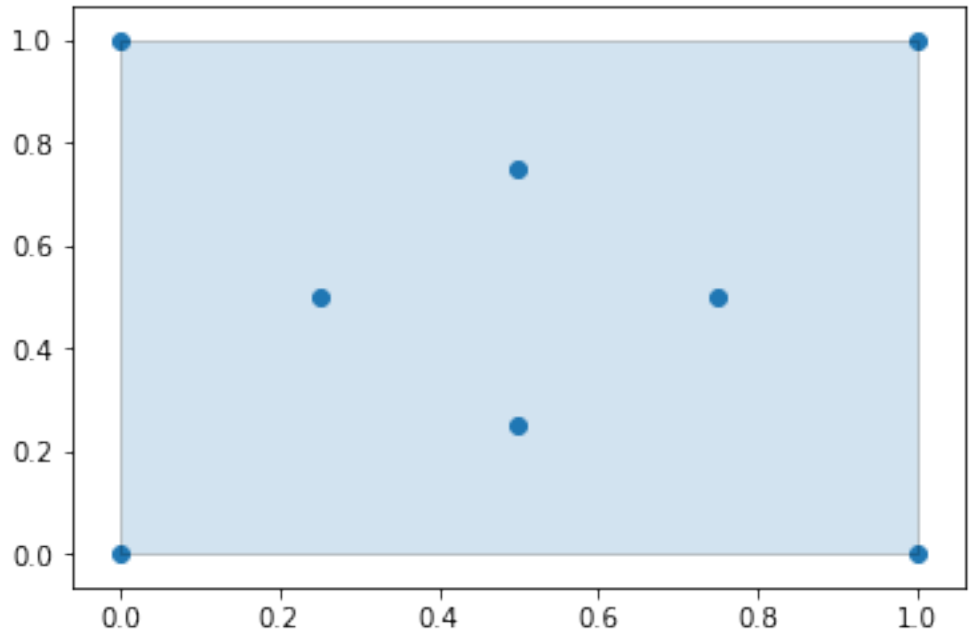
Create the alpha shape

You can visualize the shape within Jupyter notebooks using the built-in shapely renderer as shown below.

```
alpha_shape = alphashape.alphashape(points_2d, 0.)
alpha_shape
```

Plotting the alpha shape over the input data with Matplotlib

```
fig, ax = plt.subplots()
ax.scatter(*zip(*points_2d))
ax.add_patch(PolygonPatch(alpha_shape, alpha=0.2))
plt.show()
```



Generate an Alpha Shape ($alpha=2.0$) (Concave Hull)

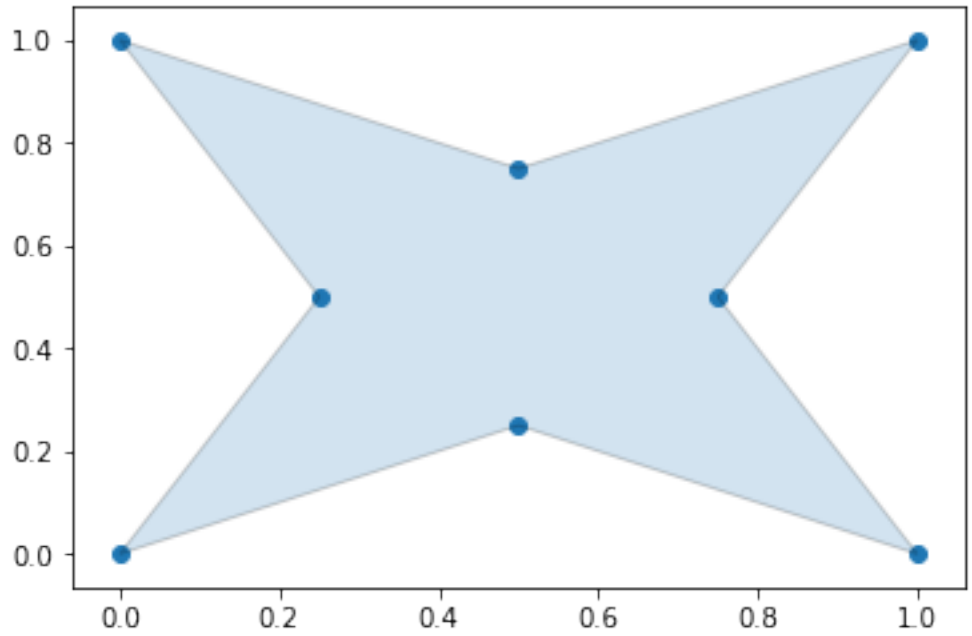
As we increase the alpha parameter value, the bounding shape will begin to fit the sample data with a more tightly fitting bounding box.

Create the alpha shape

```
alpha_shape = alphashape.alphashape(points_2d, 2.0)
alpha_shape
```

Plotting the alpha shape over the input data with Matplotlib

```
fig, ax = plt.subplots()
ax.scatter(*zip(*points_2d))
ax.add_patch(PolygonPatch(alpha_shape, alpha=0.2))
plt.show()
```



Generate an Alpha Shape ($alpha=3.5$)

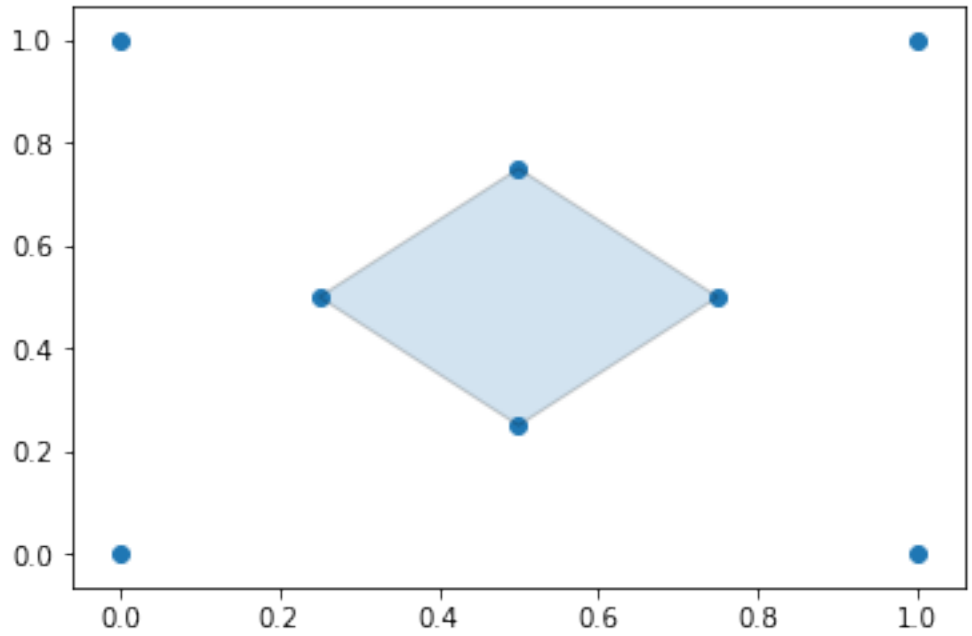
If you go too high on the alpha parameter, you will start to lose points from the original data set.

Create the alpha shape

```
alpha_shape = alphashape.alphashape(points_2d, 3.5)
alpha_shape
```

Plotting the alpha shape over the input data with Matplotlib

```
fig, ax = plt.subplots()
ax.scatter(*zip(*points_2d))
ax.add_patch(PolygonPatch(alpha_shape, alpha=0.2))
plt.show()
```



Generate an Alpha Shape (Alpha=5.0)

If you go too far, you will lose everything.

```
alpha_shape = alphashape.alphashape(points_2d, 5.0)
print(alpha_shape)
```

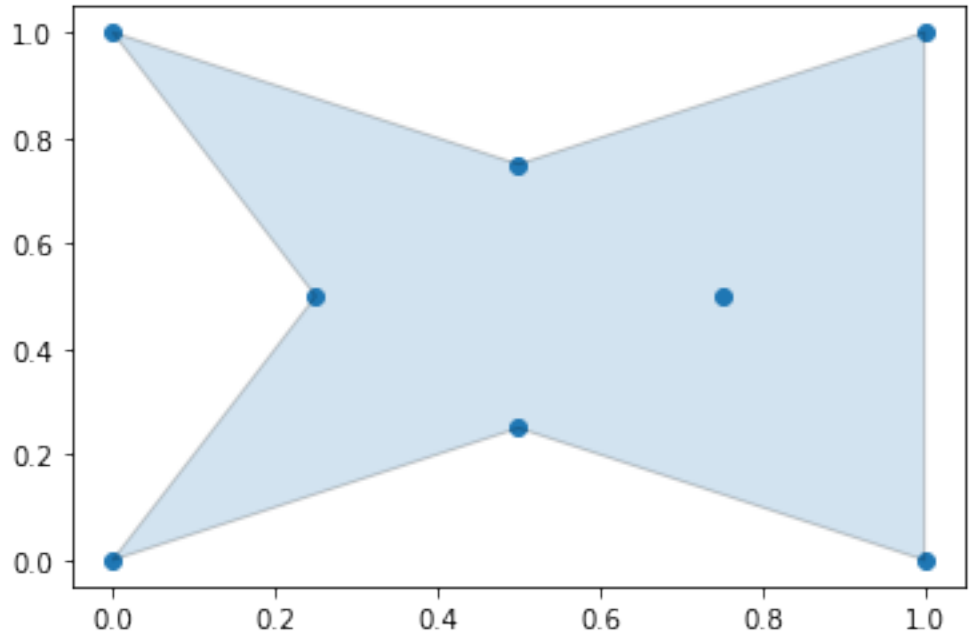
```
GEOMETRYCOLLECTION EMPTY
```

1.2 Using a varying Alpha Parameter

The alpha parameter can be defined locally within a region of points by supplying a callback that will return what alpha parameter to use. This can be utilized to create tighter fitting alpha shapes where point densitities are different in different regions of a data set. In the following example, the alpha parameter is changed based off of the value of the x-coordinate of the points.

```
alpha_shape = alphashape.alphashape(
    points_2d,
    lambda ind, r: 1.0 + any(np.array(points_2d)[ind][:,0] == 0.0))
alpha_shape
```

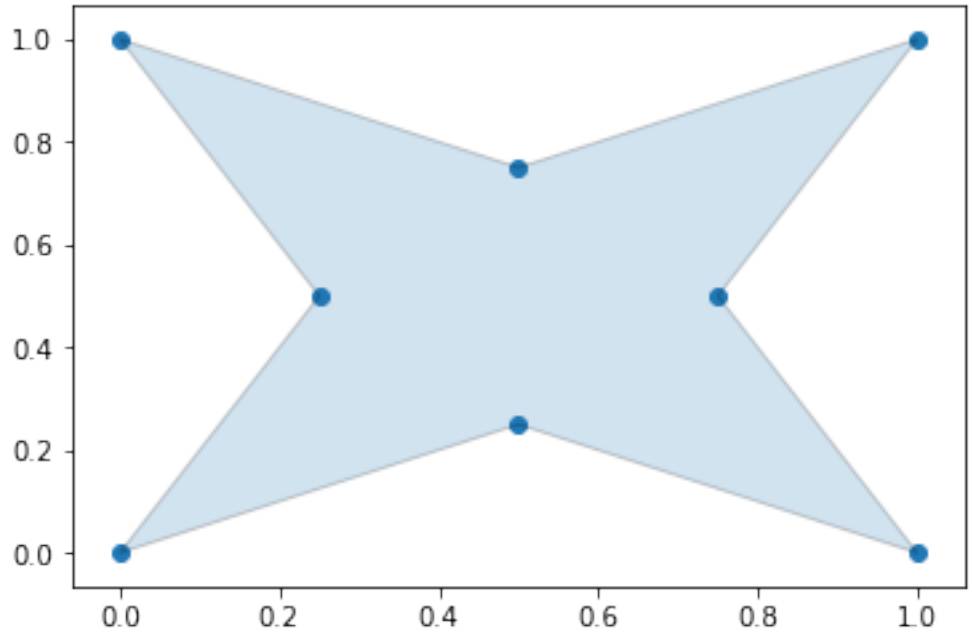
```
fig, ax = plt.subplots()
ax.scatter(*zip(*points_2d))
ax.add_patch(PolygonPatch(alpha_shape, alpha=0.2))
plt.show()
```



The alpha parameter can be solved for if it is not provided as an argument, but with large datasets this can take a long time to calculate.

```
alpha_shape = alphashape.alphashape(points_2d)
alpha_shape
```

```
fig, ax = plt.subplots()
ax.scatter(*zip(*points_2d))
ax.add_patch(PolygonPatch(alpha_shape, alpha=0.2))
plt.show()
```



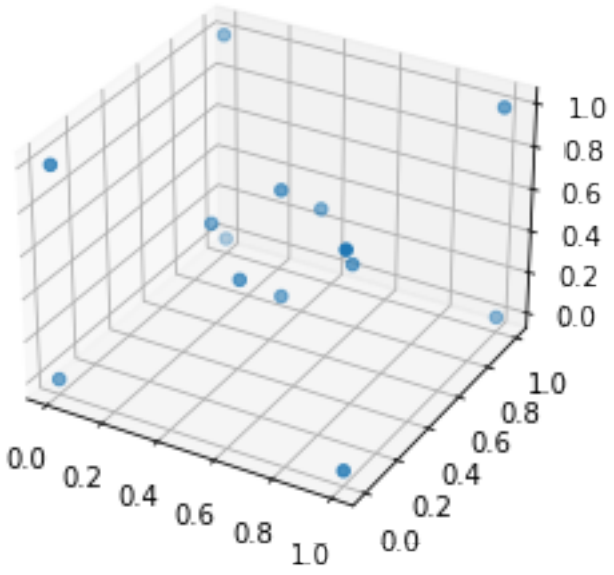
1.2.1 3 Dimensional Example

Define a set of points

```
points_3d = [
    (0., 0., 0.), (0., 0., 1.), (0., 1., 0.),
    (1., 0., 0.), (1., 1., 0.), (1., 0., 1.),
    (0., 1., 1.), (1., 1., 1.), (.25, .5, .5),
    (.5, .25, .5), (.5, .5, .25), (.75, .5, .5),
    (.5, .75, .5), (.5, .5, .75)
]
```

Visualize Test Coordinates

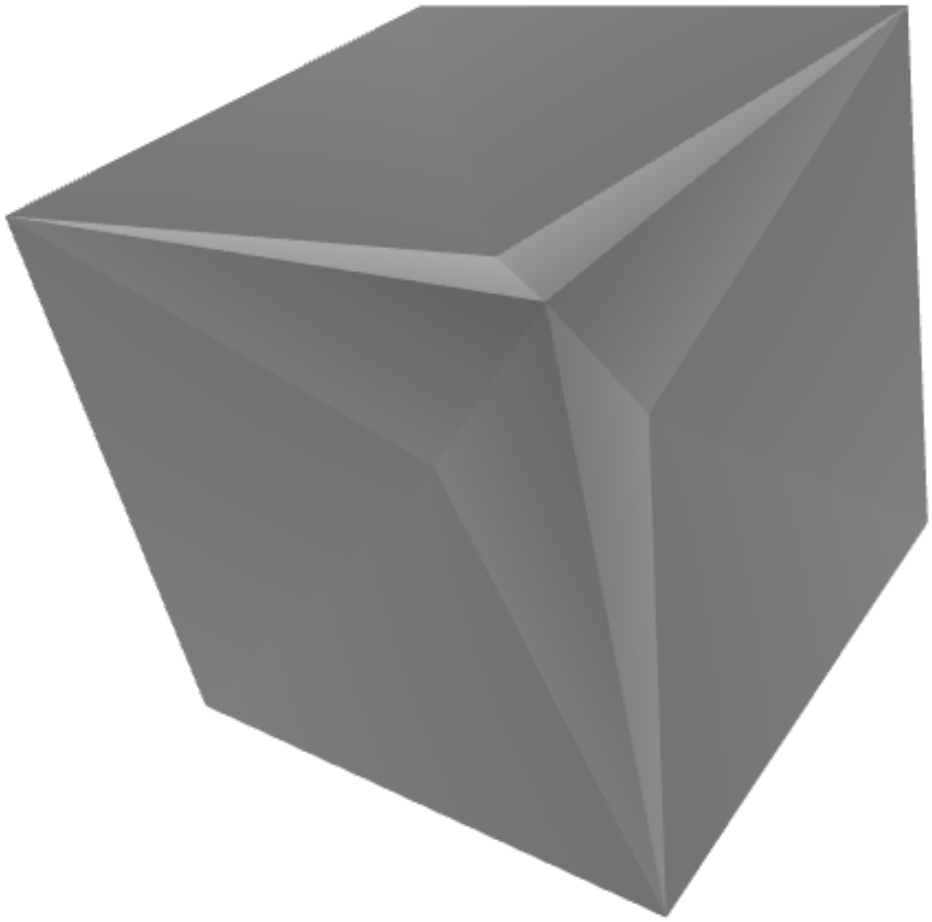
```
fig = plt.figure()
ax = plt.axes(projection='3d')
ax.scatter(df_3d['x'], df_3d['y'], df_3d['z'])
plt.show()
```



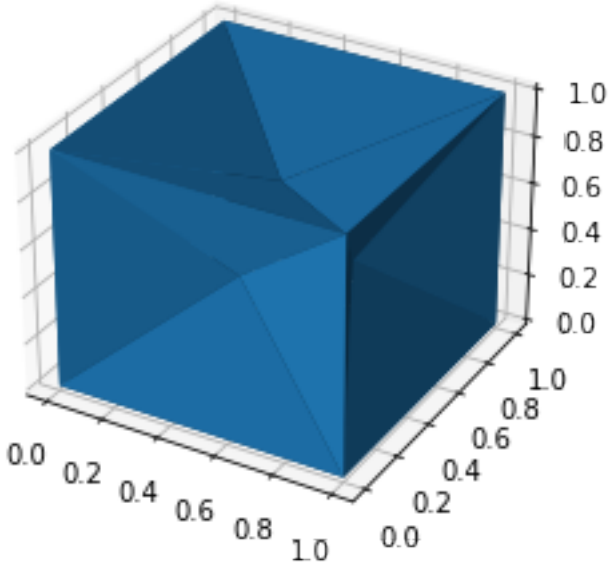
Alphashape with Static Alpha Parameter

You can visualize the shape within Jupyter notebooks using the built-in trimesh renderer by calling the `.show()` method as shown below.

```
alpha_shape = alphashape.alphashape(points_3d, 1.1)
alpha_shape.show()
```



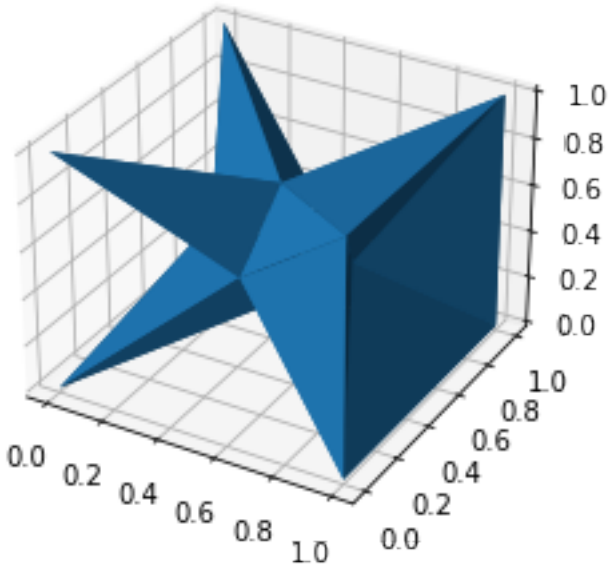
```
fig = plt.figure()
ax = plt.axes(projection='3d')
ax.plot_trisurf(*zip(*alpha_shape.vertices), triangles=alpha_shape.faces)
plt.show()
```



Alphashape with Dymanic Alpha Parameter

```
alpha_shape = alphashape.alphashape(points_3d, lambda ind, r: 1.0 + any(
    np.array(points_3d)[ind][:,0] == 0.0))
alpha_shape.show()
```

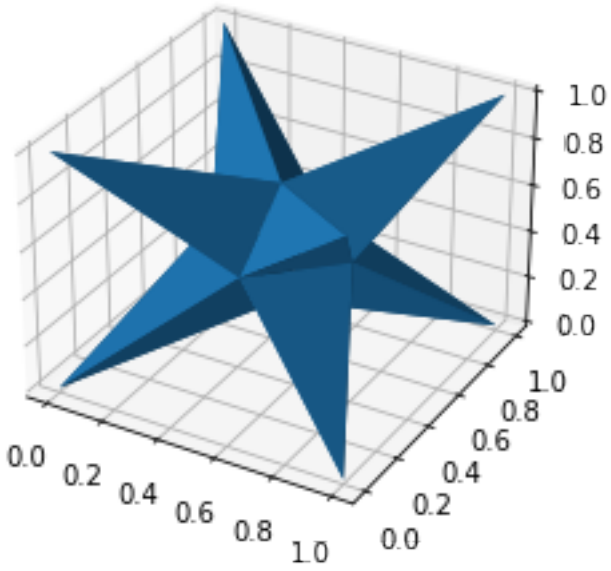
```
fig = plt.figure()
ax = plt.axes(projection='3d')
ax.plot_trisurf(*zip(*alpha_shape.vertices), triangles=alpha_shape.faces)
plt.show()
```



Alphashape found by solving for the Alpha Parameter

```
alpha_shape = alphashape.alphashape(points_3d)
alpha_shape.show()
```

```
fig = plt.figure()
ax = plt.axes(projection='3d')
ax.plot_trisurf(*zip(*alpha_shape.vertices), triangles=alpha_shape.faces)
plt.show()
```



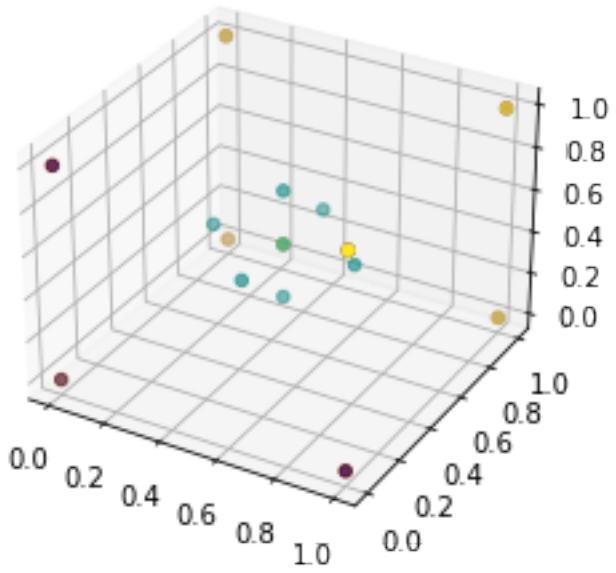
1.2.2 4 Dimensional Example

Define a set of points

```
points_4d = [
    (0., 0., 0., 0.), (0., 0., 0., 1.), (0., 0., 1., 0.),
    (0., 1., 0., 0.), (0., 1., 1., 0.), (0., 1., 0., 1.),
    (0., 0., 1., 1.), (0., 1., 1., 1.), (1., 0., 0., 0.),
    (1., 0., 0., 1.), (1., 0., 1., 0.), (1., 1., 0., 0.),
    (1., 1., 1., 0.), (1., 1., 0., 1.), (1., 0., 1., 1.),
    (1., 1., 1., 1.), (.25, .5, .5, .5), (.5, .25, .5, .5),
    (.5, .5, .25, .5), (.5, .5, .5, .25), (.75, .5, .5, .5),
    (.5, .75, .5, .5), (.5, .5, .75, .5), (.5, .5, .5, .75)
]
df_4d = pd.DataFrame(points_4d, columns=['x', 'y', 'z', 'r'])
```

Visualize Test Coordinates

```
fig = plt.figure()
ax = plt.axes(projection='3d')
ax.scatter(df_4d['x'], df_4d['y'], df_4d['z'], c=df_4d['r'])
plt.show()
```



The Edges of a 4 Dimensional Alpha Shape are Tetrahedrons Defined by the Following Coordinates (No Visualizations)

```
alphashape.alphashape(points_4d, 1.0)
```

```
{(16, 1, 2, 0),
 (16, 1, 3, 0),
 (16, 2, 3, 0),
 (16, 4, 2, 3),
 (16, 4, 7, 2),
 (16, 4, 7, 3),
 (16, 5, 1, 3),
 (16, 5, 7, 1),
 (16, 5, 7, 3),
 (16, 6, 1, 2),
 (16, 6, 7, 1),
 (16, 6, 7, 2),
 (17, 1, 2, 0),
 (17, 1, 8, 0),
 (17, 2, 8, 0),
 (17, 6, 1, 2),
 (17, 6, 14, 1),
 (17, 6, 14, 2),
 (17, 9, 1, 8),
 (17, 9, 14, 1),
```

(continues on next page)

(continued from previous page)

```
(17, 9, 14, 8),
(17, 10, 2, 8),
(17, 10, 14, 2),
(17, 10, 14, 8),
(18, 1, 3, 0),
(18, 1, 8, 0),
(18, 3, 8, 0),
(18, 5, 1, 3),
(18, 5, 13, 1),
(18, 5, 13, 3),
(18, 9, 1, 8),
(18, 9, 13, 1),
(18, 9, 13, 8),
(18, 11, 3, 8),
(18, 11, 13, 3),
(18, 11, 13, 8),
(19, 2, 3, 0),
(19, 2, 8, 0),
(19, 3, 8, 0),
(19, 4, 2, 3),
(19, 4, 12, 2),
(19, 4, 12, 3),
(19, 10, 2, 8),
(19, 10, 12, 2),
(19, 10, 12, 8),
(19, 11, 3, 8),
(19, 11, 12, 3),
(19, 11, 12, 8),
(20, 9, 13, 8),
(20, 9, 14, 8),
(20, 9, 14, 13),
(20, 10, 12, 8),
(20, 10, 14, 8),
(20, 10, 14, 12),
(20, 11, 12, 8),
(20, 11, 13, 8),
(20, 11, 13, 12),
(20, 13, 12, 15),
(20, 14, 12, 15),
(20, 14, 13, 15),
(21, 4, 7, 3),
(21, 4, 7, 12),
(21, 4, 12, 3),
(21, 5, 7, 3),
(21, 5, 7, 13),
(21, 5, 13, 3),
(21, 7, 12, 15),
(21, 7, 13, 15),
(21, 11, 12, 3),
(21, 11, 13, 3),
(21, 11, 13, 12),
(21, 13, 12, 15),
(22, 4, 7, 2),
(22, 4, 7, 12),
(22, 4, 12, 2),
(22, 6, 7, 2),
(22, 6, 7, 14),
```

(continues on next page)

(continued from previous page)

```
 (22, 6, 14, 2),
 (22, 7, 12, 15),
 (22, 7, 14, 15),
 (22, 10, 12, 2),
 (22, 10, 14, 2),
 (22, 10, 14, 12),
 (22, 14, 12, 15),
 (23, 5, 7, 1),
 (23, 5, 7, 13),
 (23, 5, 13, 1),
 (23, 6, 7, 1),
 (23, 6, 7, 14),
 (23, 6, 14, 1),
 (23, 7, 13, 15),
 (23, 7, 14, 15),
 (23, 9, 13, 1),
 (23, 9, 14, 1),
 (23, 9, 14, 13),
 (23, 14, 13, 15)}
```

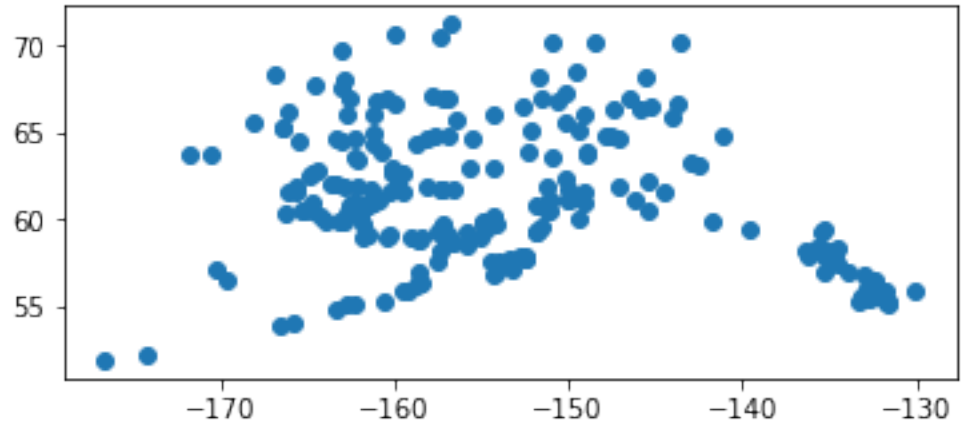
1.3 Alpha Shapes with GeoPandas

The data used in this notebook can be obtained from the Alaska Department of Transportation and Public Facilities website at the link below. It consists of a point collection for each of the public airports in Alaska.

http://www.dot.alaska.gov/stwdplng/mapping/shapefiles.shtml

```
import os
import geopandas
data = os.path.join(os.getcwd(), 'data', 'Public_Airports_March2018.shp')
gdf = geopandas.read_file(data)
```

```
%matplotlib inline
gdf.plot()
```

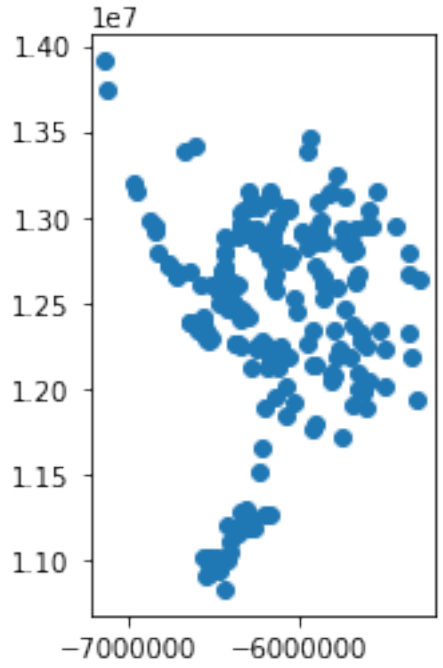



```
gdf.crs
```

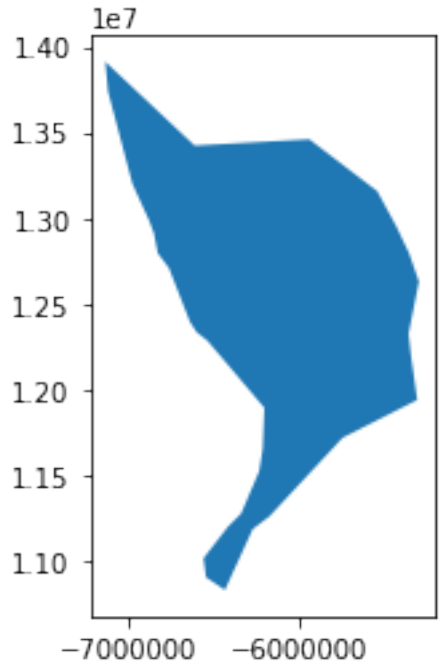
```
{'init': 'epsg:4269'}
```

The alpha shape will be generated in the coordinate frame the geodataframe is in. In this example, we will project into an Albers Equal Area projection, construct our alpha shape in that coordinate system, and then convert back to the source projection.

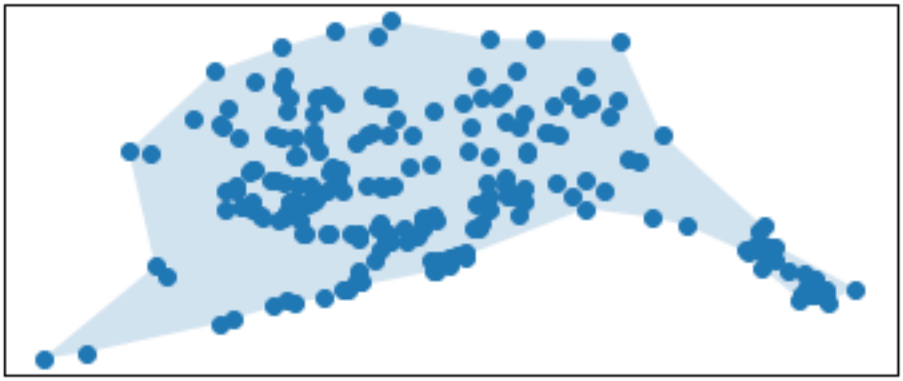
```
import cartopy.crs as ccrs
gdf_proj = gdf.to_crs(ccrs.AlbersEqualArea().proj4_init)
gdf_proj.plot()
```




```
import alphashape
alpha_shape = alphashape.alphashape(gdf_proj)
alpha_shape.plot()
```




```
import matplotlib.pyplot as plt
ax = plt.axes(projection=ccrs.PlateCarree())
ax.scatter([p.x for p in gdf_proj['geometry']],
           [p.y for p in gdf_proj['geometry']],
           transform=ccrs.AlbersEqualArea())
ax.add_geometries(
    alpha_shape['geometry'],
    crs=ccrs.AlbersEqualArea(), alpha=.2)
plt.show()
```

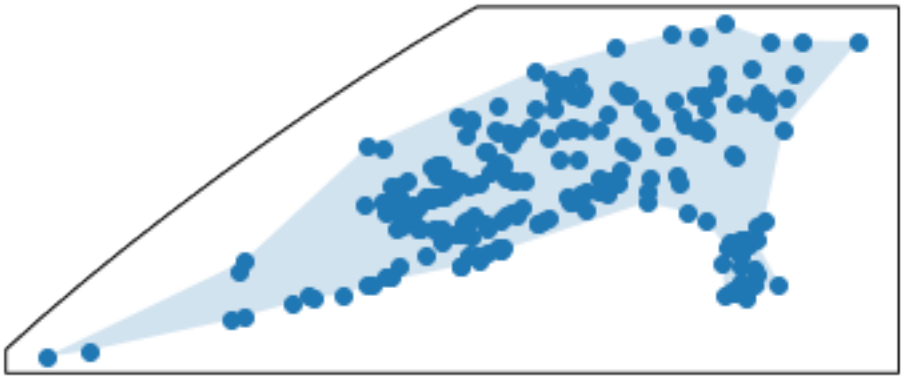


```
import matplotlib.pyplot as plt
ax = plt.axes(projection=ccrs.Robinson())
ax.scatter([p.x for p in gdf_proj['geometry']],
           [p.y for p in gdf_proj['geometry']],
           transform=ccrs.AlbersEqualArea())
ax.add_geometries(
    alpha_shape['geometry'],
    crs=ccrs.AlbersEqualArea(), alpha=.2)
```

(continues on next page)

(continued from previous page)

```
plt.show()
```



1.4 St. Sulpice Point Cloud Data

The following data can be obtained from the Lib E57 example data set found at the link below. To reduce the amount of data included in the `alphashape` toolbox repository, only a subset of point data was converted to a shapefile format and all data except point locations were dropped.

http://www.libe57.org/data.html

```
import os
import geopandas
data = os.path.join(os.getcwd(), 'data', 'Trimble_StSulpice-Cloud-50mm.shp')
gdf = geopandas.read_file(data)
```

```
from alphashape import alphashape
alphashape([point.coords[0] for point in gdf['geometry'][0]], 0.7).show()
```

1.4.1 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER

TWO

INSTALLATION

2.1 Stable release

To install Alpha Shape Toolbox, run this command in your terminal:

```
$ pip install alphashape
```

This is the preferred method to install Alpha Shape Toolbox, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

Alternatively, if you make use of Anaconda for package installation and management, Alpha Shape can be installed from conda-forge:

```
$ conda install alphashape
```

Note you will receive a 'package not found error' if conda-forge has not already been added to your channels. To add conda-forge:

```
$ conda config --add channels conda-forge
$ conda config --set channel_priority strict
```

2.2 From sources

The sources for Alpha Shape Toolbox can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/bellockk/alphashape
```

Or download the tarball:

```
$ curl  -OL https://github.com/bellockk/alphashape/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER

THREE

USAGE

To use the Alpha Shape Toolbox in a project:

```
import alphashape
```

Reference the examples gallery for further demonstration of using the Python API.

To use the Alpha Shape Toolbox from the command line:

```
Usage: alphashape [OPTIONS] SOURCE TARGET

  Example console appication using the alphashape toolbox.

  Given an input shapefile or GeoJSON INPUT with point geometry, write out a
  new OUTPUT that contains the geometries resulting from execting the alpha
  shape toolbox.

  The alpha parameter is optional.  If provided it will return the alpha
  shape for the given value, if one is not provided, the tightest fitting
  alpha shape that contains all input points will be solved for.

  The EPSG code of a coordinate system can also be given to conduct the
  alpha shape analysis in.  If one is not given the coordinate system of the
  input data will be used.

  The output file will always have the same coordinate system as the source
  file.

  The output file format will be determined by the extension of the provided
  target filename and can be written out in shapefile format or GeoJSON.

Options:
  -a, --alpha FLOAT    Alpha parameter
  -e, --epsg INTEGER   EPSG code to create alpha shape in
  -v, --verbosity LVL  Either CRITICAL, ERROR, WARNING, INFO or DEBUG
  --help               Show this message and exit.
```

CHAPTER

FOUR

GALLERY

Alpha Shape Toolbox examples:

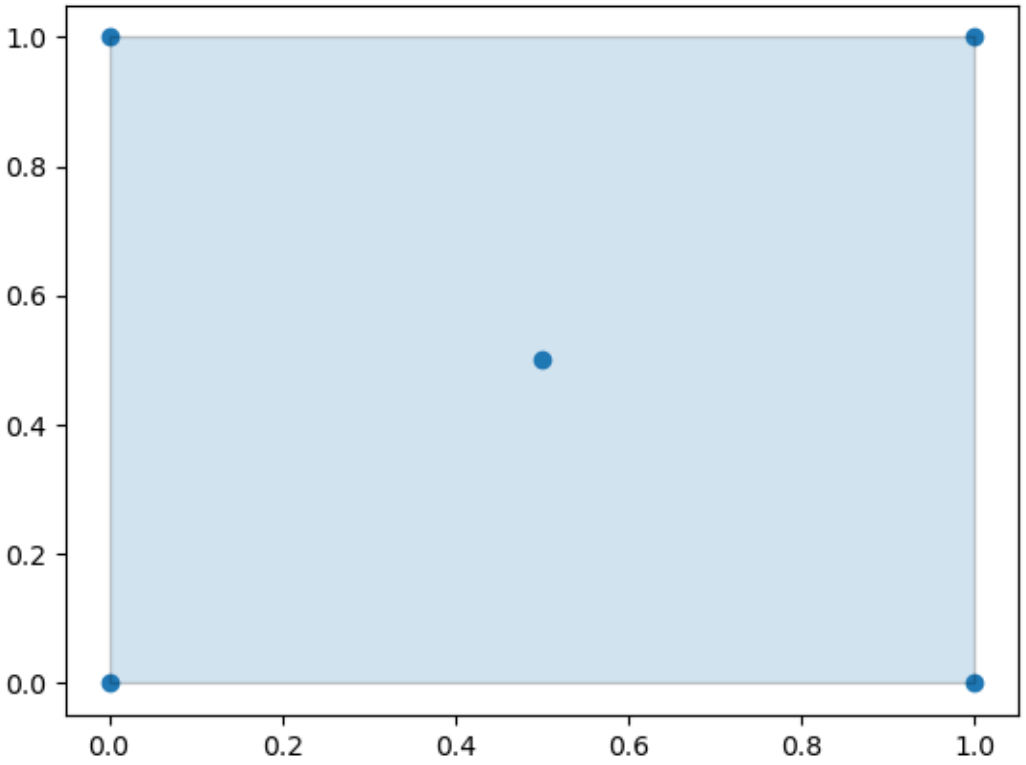
orphan

4.1 Alpha Shape Gallery

Examples are listed below.

4.1.1 Simple Example

Simple example using the alpha shape toolbox

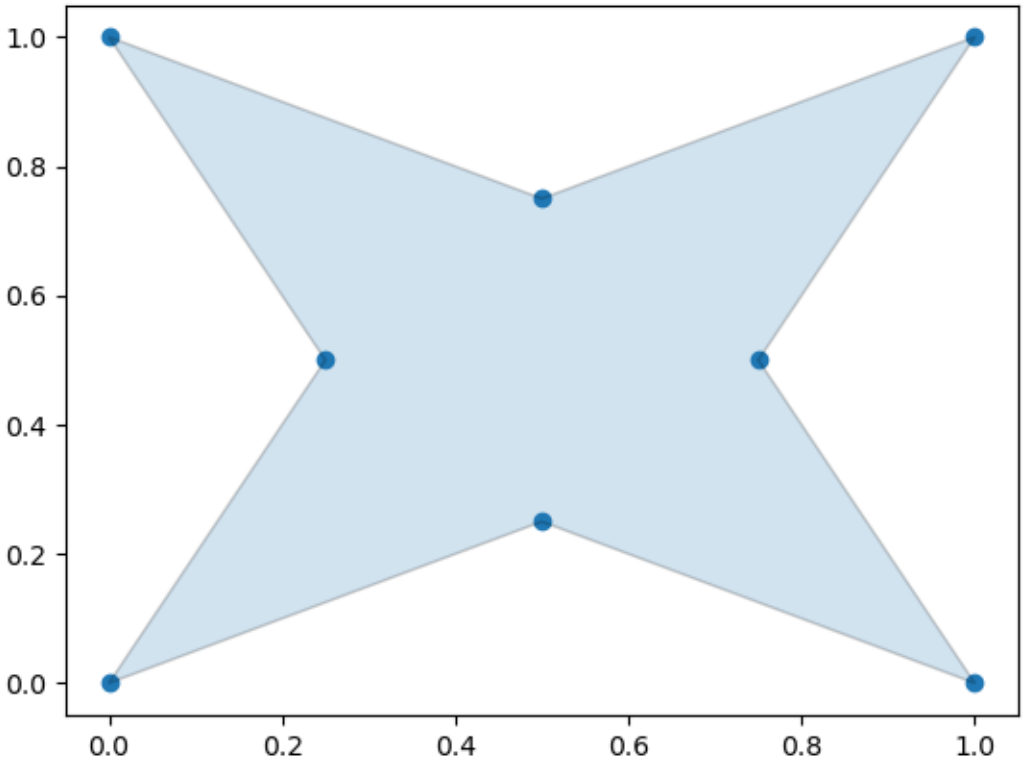



```
import alphashape
import matplotlib.pyplot as plt
from descartes import PolygonPatch

# Define input points
points = [(0., 0.), (0., 1.), (1., 1.), (1., 0.), (0.5, 0.5)]

# Define alpha parameter
alpha = 0.

# Generate the alpha shape
alpha_shape = alphashape.alphashape(points, alpha)

# Initialize plot
fig, ax = plt.subplots()

# Plot input points
ax.scatter(*zip(*points))

# Plot alpha shape
ax.add_patch(PolygonPatch(alpha_shape, alpha=.2))

plt.show()
```

Total running time of the script: (0 minutes 0.108 seconds)

4.1.2 Optimized Alpha Example

Using the optimized alpha function for obtaining the alpha parameter.




```
import alphashape
import matplotlib.pyplot as plt
from descartes import PolygonPatch

# Define input points
points = [(0., 0.), (0., 1.), (1., 1.), (1., 0.),
          (0.5, 0.25), (0.5, 0.75), (0.25, 0.5), (0.75, 0.5)]

# Determine the optimized alpha parameter
alpha = alphashape.optimizealpha(points)

# Generate the alpha shape
alpha_shape = alphashape.alphashape(points, alpha)

# Initialize plot
fig, ax = plt.subplots()

# Plot input points
ax.scatter(*zip(*points))

# Plot alpha shape
```

(continues on next page)

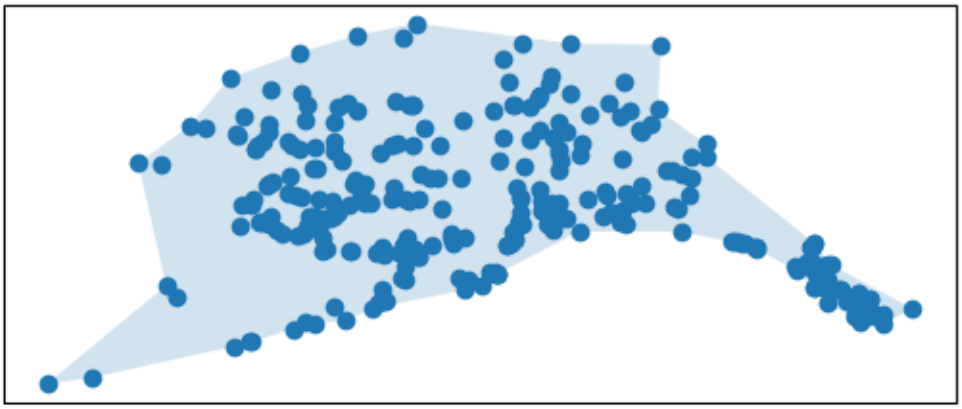
(continued from previous page)

```
ax.add_patch(PolygonPatch(alpha_shape, alpha=.2))

plt.show()
```

Total running time of the script: (0 minutes 1.239 seconds)

4.1.3 Alpha Shapes with GeoPandas GeoDataFrame

This example opens a shapefile with GeoPandas, and generates a new GeoDataFrame with the alpha shape as its only geometry. It then plots the geodataframe with cartopy.



```
import os
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
import geopandas
import alphashape

try:
    DATA = os.path.abspath(os.path.join(os.path.dirname(__file__), 'data'))
except NameError:
    DATA = os.path.abspath(os.path.join(os.path.dirname(os.getcwd()),
                                        'examples', 'data'))

```

(continues on next page)

(continued from previous page)

```
# Define input points
gdf = geopandas.read_file(os.path.join(DATA, 'Public_Airports_March2018.shp'))

# Generate the alpha shape
alpha_shape = alphashape.alphashape(gdf)

# Initialize plot
ax = plt.axes(projection=ccrs.PlateCarree())

# Plot input points
gdf_proj = gdf.to_crs(ccrs.Robinson().proj4_init)
ax.scatter([p.x for p in gdf_proj['geometry']],
           [p.y for p in gdf_proj['geometry']],
           transform=ccrs.Robinson())

# Plot alpha shape
ax.add_geometries(
    alpha_shape.to_crs(ccrs.Robinson().proj4_init)['geometry'],
    crs=ccrs.Robinson(), alpha=.2)

plt.show()
```

Total running time of the script: (0 minutes 51.791 seconds)

CHAPTER

FIVE

API REFERENCE

5.1 alphashape package

5.1.1 Module contents

Top-level package for Alpha Shape Toolbox.

`alphashape.`**`alphashape`**(*points: Union[List[Tuple[float]], numpy.ndarray]*, *alpha: Union[None, float] = None*)

Compute the alpha shape (concave hull) of a set of points. If the number of points in the input is three or less, the convex hull is returned to the user. For two points, the convex hull collapses to a *LineString*; for one point, a *Point*.

Parameters

- **`points`** (list or `shapely.geometry.MultiPoint` or `geopandas.GeoDataFrame`) – an iterable container of points
- **`alpha`** (*`float`*) – alpha value

Returns `shapely.geometry.Polygon` or `shapely.geometry.LineString` or `shapely.geometry.Point` or `geopandas.GeoDataFrame`: the resulting geometry

`alphashape.`**`alphasimplices`**(*points: Union[List[Tuple[float]], numpy.ndarray]*) → Union[List[Tuple[float]], numpy.ndarray]

Returns an iterator of simplices and their circumradii of the given set of points.

Parameters **`points`** – An *N*`x`*M* array of points.

Yields A simplex, and its circumradius as a tuple.

`alphashape.`**`circumcenter`**(*points: Union[List[Tuple[float]], numpy.ndarray]*) → numpy.ndarray

Calculate the circumcenter of a set of points in barycentric coordinates.

Parameters **`points`** – An *N*`x`*K* array of points which define an (*N*-1) simplex in K dimensional space. *N* and *K* must satisfy 1 <= *N* <= *K* and *K* >= 1.

Returns The circumcenter of a set of points in barycentric coordinates.

`alphashape.`**`circumradius`**(*points: Union[List[Tuple[float]], numpy.ndarray]*) → float

Calculte the circumradius of a given set of points.

Parameters **`points`** – An *N*`x`*K* array of points which define an (*N*-1) simplex in K dimensional space. *N* and *K* must satisfy 1 <= *N* <= *K* and *K* >= 1.

Returns The circumradius of a given set of points.

alphashape.**optimizealpha**(*points: Union[List[Tuple[float]], numpy.ndarray]*, *max_iterations: int = 10000*, *lower: float = 0.0*, *upper: float = 1.7976931348623157e+308*, *silent: bool = False*)

Solve for the alpha parameter.

Attempt to determine the alpha parameter that best wraps the given set of points in one polygon without dropping any points.

Note: If the solver fails to find a solution, a value of zero will be returned, which when used with the alphashape function will safely return a convex hull around the points.

Parameters

- **points** – an iterable container of points
- **max_iterations** (*int*) – maximum number of iterations while finding the solution
- **lower** – lower limit for optimization
- **upper** – upper limit for optimization
- **silent** – silence warnings

Returns The optimized alpha parameter

Return type float

CHAPTER

SIX

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

6.1 Types of Contributions

6.1.1 Report Bugs

Report bugs at https://github.com/bellockk/alphashape/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

6.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

6.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

6.1.4 Write Documentation

Alpha Shape Toolbox could always use more documentation, whether as part of the official Alpha Shape Toolbox docs, in docstrings, or even on the web in blog posts, articles, and such.

6.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/bellockk/alphashape/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

6.2 Get Started!

Ready to contribute? Here's how to set up *alphashape* for local development.

1. Fork the *alphashape* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/alphashape.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv alphashape
$ cd alphashape/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 alphashape tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

6.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.md.
3. The pull request should work for Python 2.7, 3.5 and 3.6, and for PyPy. Check https://travis-ci.org/bellockk/alphashape/pull_requests and make sure that the tests pass for all supported Python versions.

6.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_alphashape
```

6.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.md). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER

SEVEN

CREDITS

7.1 Development Lead

- Kenneth E. Bellock <ken@bellock.net>

7.2 Contributors

None yet. Why not be the first?

CHAPTER

EIGHT

HISTORY

8.1 1.3.1 (2021-04-16)

- Small bug fixes
- Documentation cleanup

8.2 1.3.0 (2021-04-02)

- Support for generating alphashapes for 3 or more dimensional input data.
- GeoJSON support in command line interface.

8.3 1.2.1 (2021-03-13)

- Adding in support for Python 3.6 and 3.9

8.4 1.2.0 (2021-02-25)

- Updated dependencies for geopandas notebook examples.
- Updated source information for Alaska Airports example data set.
- Dropping support for Python 3.6.

8.5 1.1.0 (2020-08-19)

- Updated dependency version numbers.
- Including optional bounds for alpha paramter solver.

8.6 1.0.1 (2019-05-06)

- Added gallery plot for optimized alpha function.
- Documentation cleanup.

8.7 1.0.0 (2019-05-06)

- #1 Update features in README.md
- #2 Create Application Utilizing the alphashape Toolbox

8.8 0.1.10 (2019-05-05)

- Correcting formatting on PyPi long description.

8.9 0.1.9 (2019-05-05)

- #7 Include GeoPandas Integration

8.10 0.1.8 (2019-05-05)

- #8 Include capability to optimize alpha parameter

8.11 0.1.7 (2019-04-26)

- Complete code coverage of existing capabilities.

8.12 0.1.6 (2019-04-24)

- #6 Include Jupyter Notebook in Examples

8.13 0.1.5 (2019-04-24)

- #5 Create an Example Gallery in the Documentation

8.14 0.1.4 (2019-04-24)

- Bug fixes.

8.15 0.1.3 (2019-04-24)

- Bug fixes.

8.16 0.1.2 (2019-04-24)

- Bug fixes.

8.17 0.1.1 (2019-04-24)

- Bug fixes.

8.18 0.1.0 (2019-04-23)

- First release on PyPI.

CHAPTER

NINE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

a

INDEX