Contents

1 API Documentation 3
2 Important links 9
PEXTANT, short for python-SEXTANT, is a tool for path planning and analysis. It is intended to be integrated with the MINERVA suite for the BASALT project and is based off of SEXTANT from MIT’s Man Vehicle Laboratory.

Contents:
1.1 EnvironmentalModel Objects

An EnvironmentalModel essentially contains an elevation map and information about which

```python
class EnvironmentalModel(elevation_map, resolution, maximum_slope, planet="Earth", NW_UTM)
```

Initialize an EnvironmentalModel object with an elevation map and obstacle information.

**Parameters**

- `elevation_map` – A 2D numpy array representing the elevation at each point.
- `resolution` *(float)* – The resolution of our map
- `maximum_slope` *(float)* – The maximum traversable slope
- `planet` *(string)* – Can be set to “Earth”, “Mars”, or “Moon”. Will only be set to “Earth” for BASALT, currently serves no use.
- `NW_UTM` – A UTMCoord object representing the North-western most corner of the map

**Instance Variables:**

- `elevations`: a 2D numpy array representing the elevation at each point.
- `slopes`: a 2D numpy array representing the slope at each point.
- `obstacles`: a 2D numpy array representing whether or not each point exceeds the value for the maximum_slope.
- `resolution`: a float representing, in meters, the resolution of the elevation map.
- `numRows, numCols`: number of rows and columns in the elevation map. Included for convenience.
- `planet`: the planet (probably “Earth”).
- `NW_UTM`: the Northwestern-most point of the map as a UTMCoord object.
- `special_obstacles`: these are obstacles from the setObstacle method.

**setMaxSlope** *(slope)*

Sets the maximum traversable slope. All points with a higher slope are marked as impassable obstacles.

**Parameters**

- `slope` *(float)* – The maximum allowed slope

**setObstacle** *(coordinates)*

Sets an obstacle at the location designated by the coordinates. Likely used for regions that are impassable due to reasons other than slope.
Parameters `coordinates` – Can be a UTMCoord, LatLongCoord, or tuple.

`eraseObstacle(coordinates)`
Erases an obstacle at the location designated by the coordinates.

`getElevation(coordinates)`
Retrieve the elevation at the location designated by the coordinates.

`loadElevationMap(file, maxSlope = 15, planet = "Earth", NWCorner = None, SECorner = None, desiredRes = None, no_val = -10000)`
Returns an EnvironmentalModel object given a geoTIFF or text file. Currently only reliably works with NAD83 and files using UTM rather than lat/long.

Parameters

- `file` – A file location of a geoTIFF or a text file representing an elevation map.
- `SECorner` (`NWCorner,`) – The coordinates of the northwestern-most corner and southeastern-most corner if a crop of the original map is desired.
- `desiredRes` – An optional input for downscaling the resolution of the elevation map.
- `no_val` – All spots labelled with the “no_val” number will be considered to be portions of the data that are incomplete

1.2 Coordinate-representing objects

Three types of objects are used to represent coordinates: tuples, which are assumed to be a row/column pair, UTMCoord, and LatLongCoord. At the moment the assumption is that all Coordinate-representing objects use NAD83, though this can be easily changed. Generally, any function requiring coordinates will accept any of these three types of coordinate objects.

`class UTMCoord(easting, northing, zone_number, zone_letter)`
Initializes an object representing UTM coordinates of a point

Parameters

- `easting` (`float`) – Easting value of the coordinate
- `northing` (`float`) – Northing value of the coordinate
- `zone` (`int`) – The zone number of the coordinate
- `zone_letter` – The zone letter of the coordinate (future versions may accept “North” or “South” as well)

`class LatLongCoord(lat, long)`
Initializes an object representing a point by latitude and longitude

Parameters

- `latitude` (`float`) – Latitude value (values north of the equator are positive, values south are negative)
- `longitude` (`float`) – Longitude value (values east of the prime meridian are positive, values west are negative)
1.3 ExplorerModel

All ExplorerModel-type objects contain basic information about a unit, as well as distance, time, and energy cost functions. A few assumptions are made:

- The velocity and metabolic rate of an explorer is solely a function of slope
- The explorer will not become “tired” as time goes on

class ExplorerModel (mass, parameters = None)
Initialize an object representing an explorer. Note that energy and time cost functions are missing

Parameters

- **mass** (float) – The mass of the explorer
- **parameters** – A parameters object which can be used to calculate shadowing. As the current version of Pextant does not support shadowing this currently has no purpose.

distance (path_length)
Returns the distance given a path length.

velocity (slope)
Returns the velocity of the explorer given the slope of the surface.

time (path_length, slope)
Returns the amount of time it takes to cross a path given the path length and slope. Calculated by dividing distance by velocity.

energyRate (slope, gravity)
Returns the rate of energy expenditure based on the slope of the ground

energy (path_length, slope, gravity)
Returns the amount of energy it takes to cross a path given the path length and slope. Calculated by multiplying energyRate by time.

class Rover (mass[, parameters = None, constant_speed = 15, additional_energy = 1500 ])
An instance of ExplorerModel representing a Rover. Contains all instance variables of ExplorerModel as well as:

Parameters

- **speed** (float) – The constant speed that the rover moves at
- **P_e** (float) – The collection of all additional electronic components on the rover, estimated to be 1500W
- **type** – Set to ‘Rover’

Includes specialized metabolic cost functions from Carr 2001.

class Astronaut (mass, parameters = None)
An instance of ExplorerModel representing a lunar Astronaut. Contains all instance variables of ExplorerModel as well as:

Parameters **type** – Set to ‘Astronaut’

Includes metabolic cost functions from Santee 2001, as well as a velocity function from Marquez 2007 (based on data from Waligoria and Horrigan 1975).

class BASALTExplorer (mass, parameters = None)
An instance of ExplorerModel representing a BASALT scientist. Currently empty; will be completed after an analysis of data from the August COTM missing, in order to derive a velocity function.
1.4 ActivityPoint

The ActivityPoint object represents points of interest for the explorer, likely spots for observation or data collection. It’s possible that future versions of Pextant may have extensions of ActivityPoint.

class ActivityPoint (coordinates, duration = 0, uuid = None)
    Initialize an ActivityPoint representing a waypoint.

    Parameters
    • coordinates – A tuple representing the location of the waypoint
    • duration (float) – The amount of time spent at the ActivityPoint, in seconds.
    • uuid (string) – A uuid value for the activityPoint

setCoordinates (coordinates)
    Sets the coordinates of the ActivityPoint to a new value. This can be a row/column tuple, a UTMCoord, or a LatLongCoord Object.

setDuration (duration)
    Sets the duration of the activityPoint.

1.5 PathFinder

class PathFinder (explorer_model, environmental_model)
    Initialize a PathFinder Object used to calculate and analyse paths.

    Parameters
    • explorer_model – An ExplorerModel object representing the explorer
    • environmental_model – An EnvironmentalModel object representing the map

aStarSearch (start, end, optimize_on)
    Returns a path through the start node and the end node using the A* search algorithm.

    Parameters
    • start – An ActivityPoint object, and the starting point of the search.
    • end – Also an ActivityPoint object
    • optimize_on – A string denoting what factor to optimize on, such as “Energy” or “Time”

fieldDStarSearch (start, end, optimize_on, numTestPoints = 11)
    Returns a path through the start node and the end node using the Field D* algorithm. Longer processing time than A*, but allows for more than the 8 cardinal directions, resulting in more “fluid” paths.

    Parameters
    • start – An ActivityPoint object, and the starting point of the search.
    • end – Also an ActivityPoint object
    • optimize_on – A string denoting what factor to optimize on, such as “Energy” or “Time”
    • numTestPoints (int) – A number used in the costFunction calculations. Higher values will involve more accuracy but increased time.

aStarCompletePath (optimize_on, activityPoints, returnType = “JSON”, fileName = None)
    Returns a path through all of the ActivityPoint objects in exploration_objects in order. The path takes the form of a long list of row/column tuples. Currently runs with the A* search algorithm.
Parameters

- `optimize_on` – Determine what factor to optimize on (can be “Energy”, “Time”, or “Distance”)
- `activityPoints` – A list of `activityPoint` objects representing the places to visit, in order
- `returnType` – A string representing the format of the path to be returned. Options are ‘tuple’, ‘JSON’, and ‘csv’
- `fileName` – The optional name of the file to be written to

`fieldDStarCompletePath(optimize_on, waypoints, returnType = "JSON", fileName = None, numtestPoints = 11)`

Similar to `aStarCompletePath`, except uses the field D* algorithm. Currently still under development.
Important links

- Pypi page: http://pypi.python.org/pypi/Pextant
- Documentation: http://sextant.rtfd.org/
- Repository: https://github.com/kezilu/sextant
- Issues: https://github.com/kezilu/sextant/issues
- BASALT website: http://spacescience.arc.nasa.gov/basalt/
Index

A
ActivityPoint (built-in class), 6
aStarCompletePath(), 6
aStarSearch(), 6
Astronaut (built-in class), 5

B
BASALTExplorer (built-in class), 5

D
distance(), 5

e
energy(), 5
energyRate(), 5
EnvironmentalModel (built-in class), 3
eraseObstacle(), 4
ExplorerModel (built-in class), 5

F
fieldDStarCompletePath(), 7
fieldDStarSearch(), 6

g
getElevation(), 4

L
LatLongCoord (built-in class), 4

P
PathFinder (built-in class), 6

R
Rover (built-in class), 5

S
setCoordinates(), 6
setDuration(), 6
setMaxSlope(), 3

setObstacle(), 3

t
time(), 5

U
UTMCoord (built-in class), 4

v
velocity(), 5

11