Package ‘viewshed3d’

July 3, 2020

Title Compute Viewshed in 3D Point Clouds of Ecosystems

Version 3.2.0

Description A set of tools to compute viewshed in 3D from Terrestrial Laser Scanner data and prepare the data prior to visibility estimation.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

Imports data.table,raster,lidR,rgl,pracma,viridis,utils,nabor,sp

NeedsCompilation no

Author Bastien Lecigne [aut, cre] (<https://orcid.org/0000-0002-1496-202X>),
Jan Eitel [aut]

Maintainer Bastien Lecigne <lecignebastien@gmail.com>

Repository CRAN

Date/Publication 2020-07-03 17:10:03 UTC

R topics documented:

denoise_scene *Filters isolated points from a point cloud*

Description

Filters isolated points from a point cloud

Usage

```
denoise_scene(data, method, filter, k, store_noise)
```

Arguments

data	LAS file of a 3D point cloud.
method	character string. Defines the method to use for noise filtering. Can be "quantile", "sd" or "voxel". See details. Default = "sd".
filter	numeric. The intensity of the filter that depends on the method. See details.
k	numeric. The number of nearest neighbours to use to compute the mean nearest neighbour distance. Required only if method = "quantile" or "sd". Default = 5.
store_noise	logical. If TRUE, the surveyed points considered as noise are not removed from the data and a column "Noise" is added with a value of 1 indicating non-noisy points and a value of 2 indicating noisy points. Default = FALSE.

Details

`method = "quantile"`: the quantile-based method computes the distance of the k nearest neighbours for each surveyed point and considers points that fall in the last user defined quantile as noise. If quantile is used as the filtering method, the default is set to = 0.999.

`method = "sd"`: the standard deviation-based method computes the average distance of the k nearest neighbours of each surveyed point and considers points as noise if they are more than the average distance plus a number of times the standard deviation away from other surveyed points. The filter parameter sets the standard deviation multiplier. Default = 4. This filter is similar to the "SOR filter" available in CloudCompare.

`method = "voxel"`: the voxel-based method considers surveyed points as noise if they are the only surveyed point within a user defined voxel volume. The `filter` parameter sets the voxel size (i.e., voxel side length). Default = 0.5.

Value

The filtered data (if `store_noise = FALSE`) or the classified data (if `store_noise = TRUE`) with noisy points labeled as 2.

Examples

```
#- import the tree_line_plot dataset
file <- system.file("extdata", "tree_line_plot.laz", package="viewshed3d")
tls <- lidR::readLAS(file,select="xyz")

#- remove duplicated points
tls <- lidR::filter_duplicates(tls)

#- filter noise with the quantile base method
data <- viewshed3d::denoise_scene(tls,
                                  method="quantile",
                                  filter=0.999,
                                  k=5,
                                  store_noise = TRUE)

lidR::plot(data,color="Noise",colorPalette=c("white","red")) # plot

#- filter noise with the standard deviation based method
data <- viewshed3d::denoise_scene(tls,
                                  method="sd",
                                  filter=4,
                                  k=5,
                                  store_noise = TRUE)

lidR::plot(data,color="Noise",colorPalette=c("white","red")) # plot

#- filter noise with the voxel based method
data <- viewshed3d::denoise_scene(tls,
                                  method="voxel",
                                  filter=0.5,
                                  store_noise = TRUE)
lidR::plot(data,color="Noise",colorPalette=c("white","red")) # plot
```

---

downsample_scene *Reduces the point cloud density*

Description

Reduces the point cloud density

Usage

```
downsample_scene(data, method, filter)
```

Arguments

data	LAS file of a 3D point cloud.

method	character string. Defines the method to use for downsampling. Can be "space" or "random". See details. Default = "space".
filter	numeric. The intensity of the filter that depends on the method. See details.

Details

method = "space": a single point is conserved within a voxel of filter size.

method = "random": randomly select a user defined proportion of the point cloud. Here, filter is the proportion of points to keep in the point cloud.

Value

The downsampled data.

Examples

```
#- import the tree_line_plot dataset
file <- system.file("extdata", "tree_line_plot.laz", package="viewshed3d")
tls <- lidR::readLAS(file,select="xyz")

#- reduce the point cloud density: keep one point in a voxel of 4cm.
sub = viewshed3d::downsample_scene(tls,filter=0.04)

#- plot the downsampled point cloud
lidR::plot(sub)
```

---

reconstruct_ground *Optimal ground reconstruction for visibility computation*

Description

Reconstructs the ground surface with a grid resolution defined by the user and adds a second grid around the animal position with an optimal resolution so that no sightline can pass through the ground when computing visibility with the visibility function.

Usage

```
reconstruct_ground(
  data,
  ground_res,
  position,
  angular_res,
  method,
  full_raster,
  ...
)
```

Arguments

data	LAS class object containing a 3d point cloud + a Classification field that classes points as ground and non-ground, as provided by the classify_ground function from the lidR-package.
ground_res	numeric. The grid resolution to reconstruct the ground on the entire 3D scene. Default = 0.05. NOTE: a if needed, second grid may be added with smaller (internally computed) resolution.
position	vector of length 3 containing the xyz coordinates of the animal position when computing the visibility with the visibility function. Default = c(0,0,0).
angular_res	numeric. The angular resolution of sightlines when computing the visibility with the visibility function. Default = 1.
method	which algorithm to use for spatial interpolation. Can be "knnidw", "tin" or "kriging". See documentation from the lidR-package for knnidw, tin and kriging.
full_raster	should the entire raster be interpolated for the ground portion around the animal position? Parameter passed to the grid_terrain function available in the lidR-package.
...	other arguments to pass to the spatial interpolation algorithm. See documentation from knnidw, tin and kriging

Value

A LAS class object containing the 3D point cloud coordinates with the ground reconstructed to be passed directly to the visibility function. Note: the Classification field is preserved.

Examples

```
#- import the tree_line_plot dataset
file <- system.file("extdata", "tree_line_plot.laz", package="viewshed3d")
tls <- lidR::readLAS(file,select="xyz")

#- class ground and vegetation points
class <- lidR::classify_ground(tls, lidR::csf(rigidness = 1L,
                                        class_threshold = 0.2,
                                        sloop_smooth = FALSE))

#- reconstruct the ground. Here the ground is reconstructed with the user
#- defined resolution only.
recons <- viewshed3d::reconstruct_ground(data=class,position = c(0,0,3),
                                         ground_res = 0.05,
                                         angular_res = 2,
                                         method="knnidw")

lidR::plot(recons,color="Classification",
           colorPalette = c("darkgreen","chocolate4"))

#- when the position is closer to the ground, the user defined resolution is
#- not sufficient and a second grid is added with the optimal resolution so
```

```
#- that no sightline can pass trough the ground when computing visibility.
#- In this example, full_raster = TRUE was used as a portion of the ground
#- near the animal location is not reconstructed because of a data
#- gap around a TLS scan position when using full_raster = FALSE.
recons <- viewshed3d::reconstruct_ground(data=class,position = c(0,0,1),
                                        ground_res = 0.05,
                                        angular_res = 2,
                                        method="knnidw",
                                        full_raster = TRUE)

lidR::plot(recons,color="Classification",
           colorPalette = c("darkgreen","chocolate4"))
```

`sample_scene` *Recenters and subsets a 3D scene for visibility estimates*

Description

Recenters and if needed subsets a 3D scan image for use in the `visibility` function. Keeps the points that fall within a user defined distance from the animal location and recenters the scene so that the animal location in the output point has 0,0,0 coordinates. The animal location can be defined by providing xyz coordinates or can be manually selected within the scene. The scene shape can be spherical or circular (see details for more information).

Usage

```
sample_scene(data, scene_radius, scene_shape, center, downsample, messages)
```

Arguments

`data`	LAS class object containing the xyx coordinates of a 3D point cloud.
`scene_radius`	numerical. The radius of the final scene. Can refer to the radius of a sphere if `scene_shape = "sph"` or of a circle if `scene_shape = "circ"`.
`scene_shape`	character string. Defines the shape of the scene: "sph" and "circ" are accepted (see details for more informations). Default = "circ".
`center`	(optional) vector of length 3 providing the xyz coordinates of the user defined animal location. If not provided, the user can manually select the animal location in the 3D point cloud. The average coordinates of the selected region will be set as the animal location (see details).
`downsample`	numeric. Enables the user to downsample the point cloud before visualizing it for scene center manual selection (if no `center` is provided). Defines the voxel resolution within which a single point of the input scene will be kept, see `tlsSample` for more details. `downsample = 0` desable downsampling. Default is 0 if the scene contains less than 5e6 surveyed points or 0.1 if the scene contains more than 5e6 surveyed points.
`messages`	logical. Disables the messages and message box when manually selecting the scene center.

Details

Scene shape: if scene_shape = "circ" the distance to scene center is computed in the xy dimension of the original scene only, resulting in a circular scene. If scene_shape = "sph" the distance to the scene center is similar to arguments scene.radius in visibility function and cut_off in viewsheds.

Manual selection of scene center: if no center is provided, a 3D plot automatically opens. The user can navigate around the surveyed points (rotate = left click, pan = right click) and select points (in a rectangular region) with the middle click. Once the points are selected, a message box opens (if not disabled). If the user clicks "yes", the plot window closes and the average point coordinates are defined as the animal location. If the user clicks "no", he/she can revise the previous selection.

Value

A LAS class object containing the coordinates of the reshaped scene.

Examples

```
#- import the tree_line_plot dataset
file <- system.file("extdata", "tree_line_plot.laz", package="viewshed3d")
tls  <-  lidR::readLAS(file,select="xyz")

#- define the animal location
center <- c(0,-6,1)

#- reshape the TLS scene with scene_shape="circ" and the calculated center
reshaped <- viewshed3d::sample_scene(tls,scene_radius = 4,
                                     center=center,
                                     scene_shape = "circ")

lidR::plot(reshaped)

#- reshape the TLS scene with scene_shape="sph" and the calculated center
reshaped <- viewshed3d::sample_scene(tls,scene_radius = 4,
                                     center=center,
                                     scene_shape = "sph")

lidR::plot(reshaped)


#- manual selection of the center
reshaped <- viewshed3d::sample_scene(tls,scene_radius = 4,
                                     scene_shape = "circ")

lidR::plot(reshaped)
```

viewshed3d **viewshed3d**: *tools to compute visibility in 3D point clouds of ecosystems*

Description

For many animals, the ability to visually assess the environment and detect approaching predators is an important part of anti-predator strategies. Because this can occur across spatial scales, estimation of the viewshed can help to quantify visibility as a continuous variable around animal locations and facilitate studies of habitat selection and predator-prey interactions.

Visibility and cumulated viewsheds: visibility within a single viewshed is calculated using the `visibility` function. This function is designed to sample the point cloud in every direction of the 3D space from a single user-defined location and to record the distance to the nearest point in each direction. Each direction is thus considered as a sightline - of a user defined angle - that is assumed to end when an object is encountered. The `viewsheds` function computes the overlap between viewsheds calculated from different locations and returns a voxel cloud quantifying for each voxel (i.e. each portion of the 3D scene) the number of times it was visible from any location.

Ground reconstruction: in the point clouds, some portions of the ground is frequently not sampled by the sensor (especially in the case of a TLS). That would result in infinite sightlines that continue below the ground surface. To correct for this effect, the `reconstruct_ground` function can be used to reconstruct the ground before using the `visibility` function. The `reconstruct_ground` function computes the optimal resolution to reconstruct the ground based on user-defined parameters for visibility calculation.

3D scene reshaping: because 3D scenes might cover a large area, but the visibility analyses might be computed for smaller areas, the `sample_scene` function can be used at the beginning of the data preparation process to segment a scene, with the appropriate properties in terms of size and shape for visibility calculation. This might be usefull to reduce computation time during the ground reconstruction process.

Noise filters: the `denoise_scene` function provides three different methods to filter isolated points from 3D point clouds.

Details

Dataset: the `viewshed3d` package provides a TLS scene of a circular forest plot located at northern treeline sites in Alaska (`tree_line_plot.laz`). This dataset has the following specifications :

- Format: LAS
- 2513044 points
- radius: 17 m
- center coordinates: 0,0,0
- duplicated points removed
- dowsampled by keeping one point within a 2 cm voxel

viewsheds *Computes cumulated viewsheds*

Description

Computes cumulated viewsheds within a 3D point cloud and return a voxel cloud accounting for the number of times each voxel was visible.

Usage

```
viewsheds(data, positions, angular_res, vox_res, cut_off, pb)
```

Arguments

`data`	LAS class object containing the xyx coordinates of a 3D point cloud.
`positions`	data.frame or data.table with 3 columns containing the xyz coordinates of the animal locations from which the viewsheds will be computed.
`angular_res`	numeric. The angular resolution of a single sightline. Default = 1.
`vox_res`	numeric. The resolution of the output voxel cloud. Default = 0.2.
`cut_off`	(optional) numeric. Defines a cut-off distance for each individual viewshed. Speeds up the process when `viewsheds` is applied to big datasets.
`pb`	logical. If `FALSE`, desables the progress bar.

Details

Sightline directions in each viewshed are computed from the method described by Malkin (2016). This ensures that every sightline explores a similar portion of the 3d space.

Value

A LAS class object containing the coordinates of the voxel cloud (X, Y, Z), and the number of times each voxel was visible from any position (N_visible).

Note

In most cases, a ground reconstruction should be performed before viewsheds computation. This can be done with the `classify_ground` and `grid_terrain` functions from the `lidR-package`.

References

Malkin, Z. (2016). A new method to subdivide a spherical surface into equal-area cells. arXiv:1612.03467.

Examples

```
#- import the tree_line_plot dataset
file = system.file("extdata", "tree_line_plot.laz", package="viewshed3d")
tls = lidR::readLAS(file,select="xyz")

#- remove noise to avoid visibility estimates error
tls_clean <- viewshed3d::denoise_scene(tls,method="sd",
                                       filter=6)

#- RECONSTRUCT THE GROUND
#- classify ground points
class=lidR::classify_ground(tls_clean, lidR::csf(rigidness = 1L,
                                           class_threshold = 0.1,
                                           sloop_smooth = TRUE), FALSE)

#- reconstruct the ground. No need for a very fine ground reconstruction.
ground = lidR::grid_terrain(class, 0.05, lidR::knnidw())

#- build the final scene
reconstructed = na.omit(raster::as.data.frame(ground, xy = TRUE))
names(reconstructed)=c("X","Y","Z")
recons=rbind(lidR::LAS(na.omit(reconstructed)),tls_clean)

#- CREATE THE POSITIONS WITH RANDOM POINTS
N_positions = 10 #- how many points ?
height = 2 #- points height relative to the ground
positions=data.table::data.table(reconstructed[runif(N_positions,
                                               1,nrow(reconstructed)),])
positions[,Z:=Z+height]

#- compute the cumulated viewsheds from the positions
cumulated=viewshed3d::viewsheds(data=recons,
                                positions = positions ,
                                angular_res = 1,
                                vox_res = 0.2)

#- plot the result
x=lidR::plot(cumulated,color="N_visible",size=3,
             colorPalette=viridis::cividis(nrow(positions)+1),trim=6)

#- add the positions
lidR::add_treetops3d(x,sp::SpatialPointsDataFrame(positions,positions),
                     radius=0.5,col="red",add=TRUE)
```

`visibility` *Computes the visibility from a single location in a 3D point cloud*

Description

Computes visibility from a user-defined location with user-defined sightline angles and returns the visibility as function of distance and, optionally, the 3D point cloud classified as visible and non-visible points.

Usage

```
visibility(data, position, angular_res, scene_radius, store_points)
```

Arguments

`data`	LAS class object containing the xyz coordinates of a 3D point cloud
`position`	vector of length 3 containing the xyz coordinates of the animal location. Default = c(0,0,0).
`angular_res`	numeric. The angular resolution of a single sightline. Default = 1.
`scene_radius`	(optional) numeric. Defines the radius of the scene relative to the animal position. Can be used to apply a cut-off distance to visibility analyses.
`store_points`	logical. If `TRUE`, the 3D point cloud is returned with visible and not visible points classified (see details).

Details

Sightline directions are computed from the method described by Malkin (2016). This ensures that every sightline explores a similar portion of the 3d space.

Value

If `store_points` = `FALSE`, a data.table of the visibility (Visibility) as a function of distance to the animal location (r) is returned. If `store_points` = `TRUE`, a list containing two objects is returned. The first object is similar to the data.table returned when `store_points` = `FALSE`. The second object is a LAS class object containing the coordinates of the point cloud (X, Y, Z), the distance of each point to the animal position (r) and the class of each point: visible or not visible from the animal position (Visibility = 2 or 1, respectively).

Note

In most cases, a ground reconstruction should be performed before visibility computation to avoid sightlines passing through the ground. This can be done with the `reconstruct_ground` function.

References

Malkin, Z. (2016). A new method to subdivide a spherical surface into equal-area cells. arXiv:1612.03467.

Examples

```
#- import the tree_line_plot dataset
file <- system.file("extdata", "tree_line_plot.laz", package="viewshed3d")
tls <- lidR::readLAS(file)

center <- c(0,0,2) # defines the scene center for the entire process
angle <- 1 # defines the angular resolution for the entire process

#- remove noise to avoid visibility estimates error
tls_clean <- viewshed3d::denoise_scene(tls,method="sd",
                                       filter=6)


#- class ground and vegetation points
class <- lidR::classify_ground(tls_clean, lidR::csf(rigidness = 1L,
                                              class_threshold = 0.2,
                                              sloop_smooth = FALSE))

#- reconstruct the ground with the optimal resolution
recons <- viewshed3d::reconstruct_ground(data=class,
                                         position = center,
                                         ground_res = 0.05,
                                         angular_res = angle,
                                         method="knnidw",
                                         full_raster = TRUE)

#- compute the visibility and store the output point cloud.
#- As the input file is a LAS object, the output
#- point cloud is also stored in a LAS file.
view.data <- viewshed3d::visibility(data = recons,
                                    position = center,
                                    angular_res = angle,
                                    scene_radius = 17, # apply cut_oof distance
                                    store_points = TRUE)

#- 3D plot with visible points in white and non-visible points in darkgrey
x=lidR::plot(view.data$points,color="Visibility",colorPalette = c("grey24","white"))

#- add animal position to the plot
position=data.frame(X=center[1],Y=center[2],Z=center[3])
lidR::add_treetops3d(x,sp::SpatialPointsDataFrame(position,position),
                     radius=0.2,col="red")

#- plot the visibility as function of distance
plot(view.data$visibility$r,view.data$visibility$visibility,
     type="l",ylim=c(0,100),lwd=4)
```

Index