Package: gaplightr (via r-universe)

May 30, 2026

Title Gap Light Analysis of Virtual and Real Fisheye Photography

Version 0.0.0.9000

Description Analyzes forest canopy gap light transmission using LiDAR point cloud data and hemispherical photography. Provides tools for processing LiDAR data, computing horizon angles, and calculating gap light metrics.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 4.2.0)

Imports lidR, plotrix, imager, terra, sf, future.apply

Suggests testthat (>= 3.0.0), withr, devtools, future, knitr, rmarkdown

VignetteBuilder knitr

Config/testthat/edition 3

URL https://hakaiinstitute.github.io/gaplightr/, https://github.com/HakaiInstitute/gaplightr

BugReports https://github.com/HakaiInstitute/gaplightr/issues

Config/pak/sysreqs libabsl-dev cmake libfftw3-dev libfreetype6-dev libgdal-dev gdal-bin libgeos-dev libglpk-dev libglu1-mesa-dev make texlive libicu-dev libjpeg-dev libpng-dev libtiff-dev libuv1-dev libxml2-dev libgl1-mesa-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev libx11-dev zlib1g-dev

Repository https://hakaiinstitute.r-universe.dev

Date/Publication 2026-05-28 21:31:56 UTC

RemoteUrl https://github.com/HakaiInstitute/gaplightr

RemoteRef HEAD

RemoteSha 275a40391a18fa14e10574d12fd7da7084163df5

Contents

`gla_create_fisheye_photos`

Create fisheye photos for multiple points

Description

Batch process LiDAR data to create synthetic hemispherical (fisheye) photographs for multiple spatial points. Supports parallel processing and can resume from previous runs.

Usage

```
gla_create_fisheye_photos(
  points,
  output_dir,
  camera_height_m = 1.37,
  min_dist = 1,
  img_res = 2800,
  max_cex = 1.5,
  min_cex = 0.04,
  pointsize = 20,
  dpi = 1200,
  parallel = TRUE,
  resume = TRUE,
  radial_distortion = "equidistant"
)
```

Arguments

`points`	An sf object containing spatial points with required columns: `las_files`, `lat`, `lon`, `elevation`, and `horizon_mask`. Use `gla_extract_horizons()` to add the `horizon_mask` column.
`output_dir`	Directory path where fisheye photo BMP files will be saved
`camera_height_m`	Camera height above ground in meters. Default is 1.37m

`min_dist` Minimum distance from camera to include LiDAR points (meters). Points closer than this distance are excluded. Must be greater than 0 - a value of 0 would allow rho = 0, causing division by zero in point size scaling. Because points with rho < min_dist are filtered out in `gla_transform_lidar()`, symbol size (cex) can only exceed max_cex when min_dist is set to a value less than 1 (or if that filtering behavior changes), leading to very large dots near the camera. Choose min_dist in concert with max_cex. Default is 1m.

`img_res` Image resolution in pixels (width and height). Default is 2800.

`max_cex` Maximum symbol size for plotting points (CEX value). Controls the size of points at rho = 1 (1 metre from camera) under the inverse-distance formula. Default is 1.5.

`min_cex` Minimum symbol size for plotting points (CEX value). The asymptotic lower bound approached as distance increases. Default is 0.04.

`pointsize` Point size parameter for bitmap graphics device. Default is 20.

`dpi` Resolution in dots per inch for output image. Default is 1200.

`parallel` Logical. If TRUE (default), use parallel processing via `future.apply`. Set up parallel plan with `future::plan()` before calling this function

`resume` Logical. If TRUE (default), skip points that already have fisheye photos in the output directory

`radial_distortion`
Lens projection method. Use "equidistant" (default) for standard equidistant polar projection, or provide custom lens calibration data (see `gla_lens_sigma_8mm()` for format).

Details

This function processes multiple points in batch, transforming LiDAR data into synthetic hemispherical photographs. For each point, it:

1. Reads and transforms the LiDAR point cloud
2. Projects points onto a hemispherical image plane
3. Creates a bitmap image with distance-weighted point sizes
4. Saves the result as a BMP file

Graphical parameter calibration: The default values for `img_res`, `pointsize`, `min_cex`, `max_cex`, and `dpi` were derived from calibration against real hemispherical photographs collected in a coastal temperate rainforest. Because optimal values depend on forest structure, canopy density, and LiDAR point density, users working in other forest types or with different LiDAR acquisitions should expect to calibrate these parameters for their study region. Calibration can be done formally (e.g., optimizing against co-located real photos) or informally through visual trial and error.

Parallel processing can significantly speed up processing for many points. Set up a parallel plan before calling this function: `future::plan(future::multisession, workers = 4)`

The resume feature allows you to interrupt and restart processing without re-creating existing photos.

Value

The input `points` sf object with an added column `fisheye_photo_path` containing the file paths to the generated fisheye photos

See Also

`gla_extract_horizons()` for extracting horizon masks

Examples

```
## Not run:
  # Assuming you have points with horizon masks already extracted
  future::plan(future::multisession, workers = 4)

  points_with_photos <- gla_create_fisheye_photos(
    points = stream_points,
    output_dir = "output/fisheye_photos",
    camera_height_m = 1.37,
    parallel = TRUE,
    resume = TRUE
  )

## End(Not run)
```

---

gla_create_virtual_plots

Create virtual plots from LiDAR data

Description

Clips circular plots from a LiDAR catalog at specified point locations. Processes points in batches to avoid memory exhaustion with large datasets.

Usage

```
gla_create_virtual_plots(
  points,
  folder,
  output_dir,
  plot_radius,
  filter = "-keep_class 1 2 9",
  chunk_size = 0,
  batch_size = 1000,
  resume = TRUE,
  check_las_catalog = FALSE,
  ...
)
```

Arguments

points	sf object with point locations
folder	Directory containing LAS/LAZ files
output_dir	Directory to save clipped plot files
plot_radius	Radius of circular plots in meters
filter	LAS filter string (default: "-keep_class 1 2 9")
chunk_size	Chunk size for LAScatalog processing (default: 0)
batch_size	Number of plots to process per batch (default: 1000). Reduce if encountering memory errors with large datasets.
resume	Skip points with existing output files (default: TRUE)
check_las_catalog	Run las_check on catalog (default: FALSE)
...	Additional arguments passed to readLAScatalog

Details

Processes plots in batches to prevent memory exhaustion. With large datasets (e.g., 5000+ plots), processing all at once can fail. The batch_size parameter controls how many plots are clipped simultaneously. Default of 1000 works well for most systems. Reduce to 500 or lower if still encountering memory issues.

Value

sf object with added las_files column containing paths to clipped plots

Examples

```
points_path <- system.file("extdata", "points.geojson", package = "gaplightr")
dem_path <- system.file("extdata", "dem.tif", package = "gaplightr")
las_folder <- system.file("extdata", "lidar", package = "gaplightr")

points <- gla_load_points(points_path, dem_path)
points <- gla_create_virtual_plots(
  points = points,
  folder = las_folder,
  output_dir = tempdir(),
  plot_radius = 50
)
```

`gla_extract_gap_fraction` *Extract gap fraction from fisheye image*

Description

Processes a fisheye image and computes gap fraction (proportion of sky visible) in elevation and azimuth bins.

Usage

```
gla_extract_gap_fraction(
  img_file,
  elev_res = 5,
  azi_res = 5,
  rotation_deg = 0,
  radial_distortion = "equidistant",
  threshold = 0
)
```

Arguments

`img_file`	Path to fisheye image file
`elev_res`	Elevation resolution in degrees (default 5)
`azi_res`	Azimuth resolution in degrees (default 5)
`rotation_deg`	Rotation angle in degrees to align image with true north (default 0)
`radial_distortion`	Lens projection method. Use "equidistant" (default) for standard equidistant polar projection, or provide custom lens calibration data (see `gla_lens_sigma_8mm()` for format).
`threshold`	Threshold value for converting image to binary. Can be: • Numeric value (0-1): pixels below threshold become 0, above become 1. Default is 0 (matches original behavior). • "auto": automatic threshold detection using Otsu's method • "XX%": percentile-based threshold (e.g., "95%")

Value

A list with gap fraction matrix and metadata

Examples

```
photo <- system.file("extdata", "example-photo.jpg", package = "gaplightr")
gla_extract_gap_fraction(photo)
```

```
# Use lens calibration
sigma_cal <- gla_lens_sigma_8mm()
gla_extract_gap_fraction(photo, radial_distortion = sigma_cal)
```

`gla_extract_horizons` *Extract horizon masks for multiple points*

Description

Extracts terrain horizon elevation angles from a DEM for multiple observation points and converts them to horizon masks suitable for fisheye photo creation. Supports caching to disk for efficient resume of interrupted workflows.

Usage

```
gla_extract_horizons(
  points,
  dem_path,
  output_dir,
  step = 5,
  max_search_distance = NULL,
  distance_step = NULL,
  camera_height_m = 1.37,
  parallel = TRUE,
  resume = TRUE,
  verbose = FALSE
)
```

Arguments

`points`	sf object containing observation points with point geometry. Must have x_meters and y_meters columns. Coordinates will be extracted from the geometry and transformed to WGS84 if necessary.
`dem_path`	Character path to the DEM raster file (GeoTIFF)
`output_dir`	Character path to directory for caching horizon CSV files. Each point's horizon data is saved as x_y_horizon.csv. Required parameter.
`step`	Numeric azimuth step size in degrees for horizon calculation (default: 5)
`max_search_distance`	Maximum search distance in meters for horizon detection (default: NULL, uses full DEM extent)
`distance_step`	Distance step size in meters for sampling along line of sight (default: NULL, uses raster resolution)
`camera_height_m`	Camera height above ground in meters. The observer elevation is calculated as ground elevation (from DEM) plus this height.

parallel	Logical. If TRUE (default), use parallel processing via future.apply. Set up parallel plan with future::plan() before calling this function
resume	Logical indicating whether to skip points that already have cached horizon files (default: TRUE). When FALSE, recomputes all horizons.
verbose	Logical indicating whether to print progress messages (default: FALSE)

Details

When parallel = TRUE, each worker loads the DEM independently from disk. Set up a parallel plan before calling this function: future::plan(future::multisession, workers = 3)

Value

The input sf object with an added horizon_mask list-column. Each element is a list containing:

x_msk Numeric vector of x-coordinates for horizon mask polygon

y_msk Numeric vector of y-coordinates for horizon mask polygon

See Also

gla_create_fisheye_photos() for using extracted horizons

Examples

```
## Not run:
  points <- gla_load_points("points.gpkg", "dem.tif")
  points <- gla_extract_horizons(
    points = points,
    dem_path = "dem.tif",
    output_dir = "output/horizons"
  )

## End(Not run)
```

---

gla_lens_sigma_8mm *Get Sigma 8mm f/3.5 EX DG lens radial calibration*

Description

Returns the radial distortion calibration data for a Sigma 8mm f/3.5 EX DG circular fisheye lens. This calibration maps normalized image radius to elevation angle.

Usage

```
gla_lens_sigma_8mm()
```

Value

A list with two components:

radius	Normalized radial distance from image center (0 to 1)
elevation	Elevation angle in radians (0 at horizon, pi/2 at zenith)

A list with components radius (normalized image radius), elevation (elevation angle in radians), and name (lens identifier for this calibration; currently "sigma8mm").

---

gla_load_points *Load and validate observation points against a DEM*

Description

Loads point locations from a spatial file or sf object, validates them against a DEM, and enriches each point with elevation, projected coordinates, and WGS84 lat/lon. The result is the starting point for all downstream gaplightr workflows.

Usage

```
gla_load_points(x, dem, drop_na_dem = FALSE, ...)
```

Arguments

x	Either a file path to any file that sf::read_sf can read, or an sf object containing point geometries. If x contains a point_id column of unique positive integers, those values are used as-is for all downstream file naming; otherwise sequential IDs are assigned automatically.
dem	Either a file path to a DEM raster file, or a SpatRaster object
drop_na_dem	Logical. If FALSE (default), points falling on NoData cells in the DEM cause an error. If TRUE, those points are dropped with a warning and processing continues with the remaining points.
...	Additional arguments passed to sf::read_sf() when x is a file path

Details

This function performs strict validation:

- Geometry type must be POINT
- CRS must be defined and projected (not geographic lat/lon)
- Point and DEM CRS must match exactly
- All points must fall within DEM spatial extent
- Points on NoData cells error by default; set drop_na_dem = TRUE to drop them with a warning instead

Point IDs:

Every point is assigned a `point_id`, a positive integer used to name all downstream output files (LAS clips, horizon CSVs, fisheye photos). If `x` does not contain a `point_id` column, sequential IDs are assigned automatically (1, 2, 3, ...). To use your own IDs (for example to preserve cached outputs across re-runs, or to match an existing site numbering scheme), include a `point_id` column containing unique positive integers before calling this function.

Examples

```
points_path <- system.file("extdata", "points.geojson", package = "gaplightr")
dem_path <- system.file("extdata", "dem.tif", package = "gaplightr")
points <- gla_load_points(points_path, dem_path)



# Supply your own point_id values to keep cached outputs stable across
# re-runs or to match an existing site numbering scheme.
dem_path <- system.file("extdata", "dem.tif", package = "gaplightr")
my_points <- sf::st_read(
  system.file("extdata", "points.geojson", package = "gaplightr"),
  quiet = TRUE
)
my_points$point_id <- c(101L, 202L, 303L)  # user-defined IDs
points <- gla_load_points(my_points, dem_path)
```

`gla_process_fisheye_photos` *Process fisheye photos for solar radiation analysis*

Description

Batch processes fisheye photos for multiple points, computing gap fractions and transmitted solar radiation. This is the main workflow function for analyzing hemispherical fisheye photos.

Usage

```
gla_process_fisheye_photos(
  points,
  clearsky_coef = 0.65,
  time_step_min = 2,
  day_start = 1,
  day_end = 365,
  day_res = 1,
  elev_res = 5,
  azi_res = 5,
  Kt = 0.45,
```

```
  rotation_deg = 0,
  parallel = TRUE,
  keep_gap_fraction_data = FALSE,
  radial_distortion = "equidistant",
  threshold = 0,
  solar_constant = 1367
)
```

Arguments

points	An sf object with point locations. Must contain columns: fisheye_photo_path, lat, lon, and elevation.
clearsky_coef	Clear-sky transmission coefficient (default 0.65). Proportion of extraterrestrial radiation reaching the surface under clear skies.
time_step_min	Time step in minutes for computing solar positions (default 2).
day_start	Start day of year (default 1). Use day-of-year format (1-365).
day_end	End day of year (default 365).
day_res	Day resolution - compute every N days (default 1 for daily).
elev_res	Elevation resolution in degrees (default 5)
azi_res	Azimuth resolution in degrees (default 5)
Kt	Mean cloudiness index (Kt = H/Ho) for the period of interest (default 0.45). Ratio of measured to extraterrestrial radiation.
rotation_deg	Rotation angle in degrees to align image with true north (default 0)
parallel	Logical. If TRUE (default), use parallel processing via future.apply. Set up parallel plan with future::plan() before calling this function
keep_gap_fraction_data	Include gap fraction matrix in output (default FALSE). If TRUE, adds gap_fraction_data column to output.
radial_distortion	Lens projection method. Use "equidistant" (default) for standard equidistant polar projection, or provide custom lens calibration data (see gla_lens_sigma_8mm() for format).
threshold	Threshold value for converting image to binary. Can be: • Numeric value (0-1): pixels below threshold become 0, above become 1. Default is 0 (matches original behavior). • "auto": automatic threshold detection using Otsu's method • "XX%": percentile-based threshold (e.g., "95%")
solar_constant	Solar constant in W/m² (default 1367). The total solar electromagnetic radiation per unit area at the top of Earth's atmosphere.

Value

An sf object with computed solar radiation metrics:

`canopy_openness_pct`
: Canopy openness percentage derived from the fisheye photo gap fraction

`mean_daily_extraterrestrial_irradiance_Wm2`
: Mean daily irradiance above the atmosphere (W/m²). Computed from astronomical parameters only - no atmospheric, canopy, or topographic effects.

`mean_daily_direct_irradiation_MJm2d`
: Mean daily direct irradiation on a flat open surface (MJ/m²/day). Computed from extraterrestrial irradiance and `Kt` only - no canopy or topographic shading.

`mean_daily_diffuse_irradiation_MJm2d`
: Mean daily diffuse irradiation on a flat open surface (MJ/m²/day). Computed from extraterrestrial irradiance and `Kt` only - no canopy or topographic shading.

`mean_daily_global_irradiation_MJm2d`
: Mean daily global irradiation on a flat open surface (MJ/m²/day). Computed from extraterrestrial irradiance and `Kt` only - no canopy or topographic shading.

`transmitted_direct_irradiation_MJm2d`
: Direct irradiation reaching the sensor (MJ/m²/day), accounting for both canopy and topographic shading encoded in the fisheye photo.

`transmitted_diffuse_irradiation_MJm2d`
: Diffuse irradiation reaching the sensor (MJ/m²/day), accounting for both canopy and topographic shading encoded in the fisheye photo.

`transmitted_global_irradiation_MJm2d`
: Global irradiation reaching the sensor (MJ/m²/day), accounting for both canopy and topographic shading encoded in the fisheye photo.

`transmitted_direct_irradiation_pct`
: Transmitted direct irradiation as a percentage of the flat open-sky reference (`mean_daily_direct_irradiation_MJm2d`). The denominator does not account for topographic shading, so this value reflects the combined effect of canopy and terrain.

`transmitted_diffuse_irradiation_pct`
: Transmitted diffuse irradiation as a percentage of the flat open-sky reference (`mean_daily_diffuse_irradiation_MJm2d`). The denominator does not account for topographic shading, so this value reflects the combined effect of canopy and terrain.

`transmitted_global_irradiation_pct`
: Transmitted global irradiation as a percentage of the flat open-sky reference (`mean_daily_global_irradiation_MJm2d`). The denominator does not account for topographic shading, so this value reflects the combined effect of canopy and terrain.

Examples

```
## Not run:
# Process fisheye photos for August (days 213-243)
results <- gla_process_fisheye_photos(
  points = stream_points,
  day_start = 213,
  day_end = 243,
```

```
  Kt = 0.45
)

# Use automatic thresholding for real photos
results <- gla_process_fisheye_photos(
  points = stream_points,
  threshold = "auto"
)

# Keep gap fraction data for further analysis
results <- gla_process_fisheye_photos(
  points = stream_points,
  keep_gap_fraction_data = TRUE
)

## End(Not run)
```

Index