Package: uasimg (via r-universe)

March 14, 2025

Type Package

Title Drone Image Utilities

Description This package manages images from drone mapping projects, including data management, metadata creation, creating data catalogs, converting file formats, and utilities for single-image analysis.

Date 2025-01-13

Version 1.9.4

BugReports https://github.com/ucanr-igis/uasimg/issues

URL https://ucanr-igis.github.io/uasimg/

License GPL-3

Encoding UTF-8

Language en-US

LazyData true

Depends R (>= 3.6.0), magrittr

Imports base64enc, crayon, digest, dplyr, exiftoolr, htmltools, imager, leaflet, leaflet.extras, lifecycle, methods, readr, rmarkdown, sf, stringr, tibble, tidyr, tools, xml2

Suggests ggmap, ggplot2, magick, kableExtra, knitr, DiagrammeR, DiagrammeRsvg, terra

RoxygenNote 7.3.2

VignetteBuilder knitr

Config/pak/sysreqs libfftw3-dev libgdal-dev gdal-bin libgeos-dev libglpk-dev make libicu-dev libjpeg-dev libpng-dev libtiff-dev libxml2-dev libssl-dev perl libproj-dev libsqlite3-dev libudunits2-dev libx11-dev

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

RemoteUrl https://github.com/ucanr-igis/uasimg

RemoteRef HEAD

RemoteSha ad578a7746d1780dc532c5ea24224c5c5d0107eb

Contents

`findonpath` *findonpath*

Description

Searches for a file on the system environment path

Usage

```
findonpath(fn, status = TRUE, quote = TRUE)
```

Arguments

`fn`	The filename to search for (without a path)
`status`	Show messages
`quote`	Enclose returned filename in quotes

Value

The full path and name of the found file, or NULL if not found

Note

This will return the first found occurence of file `fn`, searching 1) the current working directory, 2) the user's R 'home' directory, then 3) the directories on the operating system environment 'path'

`geo2utm` *Look up UTM zone*

Description

Finds the UTM zone for a geographic coordinate

Usage

```
geo2utm(x, lat = NULL)
```

Arguments

`x`	Longitude in decimal degrees. Can also be a numeric vector of length 2 containing longitude and latitude values.
`lat`	Latitude in decimal degrees (omit if x is a vector of long and lat)

Details

This will return the EPSG number of the UTM zone in which `x` falls

Value

The EPSG number of the correct UTM zone

`m2ft` *Unit conversion functions*

Description

Unit conversion functions

Usage

```
m2ft(x)

ft2m(x)

cm2in(x)

msq2acres(x)

msq2ha(x)
```

Arguments

x	A numeric vector to convert

Details

These unit conversion functions are vectorized.

Value

A vector of converted values

Functions

- `ft2m()`: Feet to meters
- `cm2in()`: Centimeters to inches
- `msq2acres()`: Meters squared to acres
- `msq2ha()`: Meters squared to hectacres

`print.uas_grp` *Prints a UAS Image Collection Groups object*

Description

Prints a UAS Image Collection Groups object

Usage

```
## S3 method for class 'uas_grp'
print(x, ...)
```

Arguments

x	A UAS Image Collection Groups object
...	unused

print.uas_info *Print a summary of UAS metadata*

Description

Print a summary of metadata from a UAS image collection

Usage

```
## S3 method for class 'uas_info'
print(x, metadata = TRUE, ...)
```

Arguments

x	A UAS Image Collection object
metadata	Show additional meta data info
...	Other arguments (unused)

Details

Prints a summary of a drone images metadata collection object

See Also

uas_info

uas_cameras *Cameras*

Description

Returns the names of known cameras

Read a cameras.csv file

Usage

```
uas_cameras(names_only = TRUE)

uas_readcameras(cameras_fn)
```

Arguments

names_only	Return only camera names
cameras_fn	The file name (including path) of a cameras.csv file

Details

In order to estimate the GSD and image footprint, certain properties of the camera are required, including the focal length and physical dimension of the CCD sensor. Properties of several commonly used cameras have been collected and are bundled with the package as a csv file.

If your drone camera is not in the database, you can still index your images `uas_info`, however generic camera settings (EXIF fields) will be used. In most cases, these settings will work fine to get the GPS coordinates and timestamps, however it won't be able to model the on-the-ground image footprint.

To add your camera to the database, you can start an issue on GitHub, or contact the package author by email. Alternately, you can find the cameras.csv file in the package folder (run `system.file("cameras/cameras.csv", package = "uasimg")`), and add a row for your camera. Or copy the csv file and pass the file name as the value of `cameras` in `uas_info`.

This utility function will read a cameras.csv file. Note certain columns are required. To see an example of a valid CSV file, run `uas_cameras(names_only = FALSE)`

Value

A character vector containing the names of cameras known to the uasimg package

Functions

- `uas_readcameras()`: Read cameras csv file

See Also

`uas_info`

Examples

```
## Not run:
uas_cameras()

## End(Not run)

## Not run:
uas_readcameras(system.file("cameras/cameras.csv", package="uasimg"))

## End(Not run)
```

`uas_convert` *Convert file format*

Description

Convert image file formats while preserving EXIF data

Usage

```
uas_convert(
  x,
  dir_out = NULL,
  idx = NULL,
  format_out = c("jpg", "tif")[1],
  quality = 90,
  copy_exif = TRUE,
  overwrite = FALSE,
  quiet = FALSE
)
```

Arguments

x	A vector of filenames (with path), or an object of class 'uas_info'
dir_out	The output directory(s)
idx	Row numbers of the images (optional)
format_out	The output format
quality	The quality level for jpg compression (0..100), larger numbers produce better image quality
copy_exif	Whether to copy the original EXIF info to the new file, logical
overwrite	Overwrite existing files, logical
quiet	Suppress messages, logical

Details

This function converts image formats. Images are converted using the 'magick' package (which calls ImageMagick libraries), while EXIF data is copied using exiftool. Supported output formats include JPG and TIFF.

x can be a vector of image file names (including the path), or a image metadata object (created by uas_info). If idx (row numbers) is passed, it will be used to select images to convert.

dir_out is the name of the directory where converted images will be output. If NULL, images will be placed in the same directory as the input images. If x is an object of class 'uas_info', dir_out can be a vector of directories equal in length to the number of folders indexed in x (in which case the converted images will be placed in the corresponding output directory).

quality sets the compression ratio for jpg images. To preserve a high level of fidelity and improve the odds of successful stitching, it is recommended you set quality to 90 or higher. If format_out = "tif", the output files will be saved as uncompressed tif files.

Examples

```
## Not run:
## Make a list of DNG files to convert
files_dng <- list.files("D:/uas/mavic_pro/raw",
                        pattern = ".DNG$",
                        full.names = TRUE,
```

```
                        ignore.case = TRUE)

uas_convert(files_dng,
            dir_out = "D:/uas/mavic_pro/jpg",
            format_out = "jpg")

## End(Not run)
```

---

uas_cropctr *Crop out the center of images*

Description

Crop out the center of images

Usage

```
uas_cropctr(
  x,
  crp_h = 10,
  crp_w = 10,
  dir_out = ".",
  out_prefix = "",
  out_suffix = "_crp",
  overwrite = FALSE,
  quiet = FALSE
)
```

Arguments

x	A list of class 'uas_info'
crp_h	Height of the crop box (meters)
crp_w	Width of the crop box (meters)
dir_out	The output directory for the cropped images
out_prefix	A character object that will be pre-pended to output file names
out_suffix	A character object that will be suffixed to output file names (before the extension)
overwrite	Overwrite existing files, T/F
quiet	Suppress messages and printing of the pandoc command line, T/F

Details

This function will crop around the center of a set of images. This can be used to create a photo-mosaic. If the photos were taken at nadir (i.e., straight down), the center region of each photo is often the least distorted. If the images have world files created for them (see uas_worldfile), the cropped images will have the world files also. If the images have been cropped to minimize the amount of overlap between then, and they have world files, viewing them in GIS software should results in a basic photo mosaic (albeit without any stitching or blending).

This function requires the command line version of gdal to be installed. For instructions see http://gdal.org. Windows users can download a gdal installation file (*.msi) from http://www.gisinternals.com/release.php, install the software, add the installation directory to the system path (using the Windows Control Panel » System » Advanced), and restart RStudio. To see if it worked, run `Sys.which("gdal_translate")`.

If the images were collected in a grid pattern, you can use the average distance between centers as the value for 'crp_h', and the average distance between flight lines for 'crp_w'. This will result in cropped images where the edges are just touching. 'crp_h' and 'crp_w' should be expressed in meters.

Value

A vector of the image files created

See Also

uas_worldfile

---

uas_dirs_find *Search directories for images*

Description

Find sub-directories which contain images

Usage

```
uas_dirs_find(x, ext = c("jpg", "tif"), min_images = 3, exclude_tb = TRUE)
```

Arguments

x	The base directory
ext	A character vector of the file extensions to search for (excluding the '.')
min_images	The minimum number of image files a directory must contain to be included in the results
exclude_tb	Exclude folders called 'tb' (thumbnails)

Details

This function recurses through sub-directories of ‘x‘ and returns those that contain image files. This can be used to help identify folders that need to be cataloged.

Only directories that have at least `min_images` image files will be returned. If `exclude_tb = TRUE`, directories named ‘tb‘ (which is where `uas_thumbnails_make` saves thumbnail images) are excluded. Hidden directories and directories that start with ‘.‘ are excluded.

Value

A tibble with 3 columns: `path`, `ext`, and `num_files`.

`uas_exp_geotiff` *Export individual images as a pseudo-georectified GeoTIFF*

Description

Export individual images as a pseudo-georectified GeoTIFF

Usage

```
uas_exp_geotiff(
  x,
  flt = NULL,
  idx = NULL,
  method = c("near", "binlinear", "cubic")[1],
  crs = NULL,
  dir_out = ".",
  overwrite = FALSE,
  use_tmpdir = FALSE,
  quiet = FALSE
)
```

Arguments

`x`	A list of class ’uas_info’
`flt`	Which flight(s) in x to process (default is all)
`idx`	Which row numbers (i.e., images) in the selected flights to process (default is all)
`method`	Resampling method to use for the rotation
`crs`	coordinate reference system of the output file
`dir_out`	The output directory (default is the same folder as the source images)
`overwrite`	Overwrite existing files
`use_tmpdir`	Use the temp directory
`quiet`	Show messages.

Details

This function will export individual image(s) to a pseudo-georectified GeoTIFF, using the estimated ground footprint modeled from the EXIF data. It should be self-evident that footprints should be considered approximate at best, as they modeled from a) the above-launch-point elevation, b) GPS location, c) yaw, and d) camera parameters (all of which have error).

Note: if your goal is simply to view individual images in their approximate location in desktop GIS software, this function is probably overkill (use `uas_worldfile` instead).

A prerequisite for using this function is that you computed footprints when you first created the image collection object (in other words, be sure to pass pass `fp = TRUE` when you run `uas_info`). This function also requires that you have 'terra' installed.

`method` is the name of a resampling method used to create the pseudo-georectified image. The default is `near` which is fastest and should give reasonably good results. See also `terra::resample()`.

The crs of the pseudo-georectified image will be the crs of the image collection object (i.e., UTM). You can override this with the `crs` argument. If provided, `crs` should be provided as text as a <authority:number> code (e.g., `"epsg:4326"`) or WKT syntax. For details see `terra::project()`.

Note rectifying (unrotating) images can take a long time and result in much larger image files (because GeoTiffs are uncompressed). You can use the `flt` and `idx` arguments to specify which flight(s) and images(s) within selected the flights respectively to export.

This function has been tested with JPG files from DJI cameras. It has not yet been fully tested for TIF files from multispectral cameras, and may not work with those formats (contact the package author if you want to try).

Un-rotating the images requires write permission for the directory where the input images are saved (to write a temporary worldfile). If you don't have write permission where the images reside, pass `use_tmpdir = TRUE`. This will make a temporary copy of the image in the temp directory.

Value

A list of filenames generated.

See Also

`uas_info`, `uas_worldfile`

`uas_exp_kml` *Export flight area and camera locations for GIS*

Description

Export geometry(s) from a flight to KML and Shapefile

Usage

```
uas_exp_kml(
  x,
  flt = NULL,
  ctr = FALSE,
  fp = FALSE,
  mcp = FALSE,
  combine_feats = FALSE,
  combine_fn = NULL,
  output_dir = NULL,
  out_fnbase = NULL,
  create_dir = TRUE,
  overwrite = FALSE,
  quiet = FALSE,
  flt_idx = deprecated()
)

uas_exp_shp(
  x,
  flt = NULL,
  ctr = FALSE,
  fp = FALSE,
  mcp = FALSE,
  combine_feats = FALSE,
  combine_fn = NULL,
  output_dir = NULL,
  out_fnbase = NULL,
  create_dir = TRUE,
  overwrite = FALSE,
  quiet = FALSE,
  flt_idx = deprecated()
)
```

Arguments

x	A list of class 'uas_info'
flt	Flight(s) in x to process (character or numeric vector, default is all)
ctr	Export the image centroids, Logical
fp	Export the image footprints, Logical
mcp	Export the minimum convex polygon of the image footprints, logical
combine_feats	Combine features into a single Shapefile / KML, logical
combine_fn	File name (minus path and extension) for the combined features layer
output_dir	The output directory. If NULL, the layers will be saved in a 'map' sub-directory of the image folder
out_fnbase	The base of output filenames
create_dir	Create the output directory if it doesn't exist, logical

overwrite	Overwrite existing files, Logical
quiet	Suppress messages, Logical
flt_idx	‘r lifecycle::badge("deprecated")‘ Use ‘flt‘ instead

Details

flt allows you to specify a subset of image folders in x to process. You can pass a vector of flight names (use names(x) to see what those are) or integers.

ctr, fp, and mcp (all TRUE/FALSE) specify which geometry(s) to export. Output file names will be generated from the flight metadata saved in the uas_info object. Alternately, you can pass the base of a file name using out_fnbase. You can specify the output directory with output_dir. The default is to save them in a sub-directory of the images directory called 'map', which will be created if needed.

If combine_feats = TRUE, the geometries from the folders in x will be combined into a single KML or Shapefile. For this to happen, you must also pass a value for combine_fn (a base for the file name of the combined features).

Value

A vector of filenames

Functions

- uas_exp_shp(): Export flight info to Shapefile

See Also

uas_info, uas_report

---

uas_getcache *Manage cache directory*

Description

View and set the directory where extracted EXIF data are cached

Usage

```
uas_getcache(default = TRUE, quiet = FALSE)

uas_setcache(dir = "~/.R/uasimg", write = FALSE, quiet = FALSE)

uas_clearcache(quiet = FALSE)
```

Arguments

default	Use a default cache directory if no other has been set
quiet	Suppress messages
dir	The directory for cached EXIF data (must exist)
write	Write directory location to .Renviron

Details

Extracting exif data from a large number of images can take awhile. To avoid having to do this more than once for the same folder of images, the results can be saved (or cached) to a local directory of your choice. When cache = TRUE in uas_info, R will first look to see if EXIF data has already been extracted for the image folder, and if so use it instead of running exiftool.

Cached results are saved as native R objects. Cache files store the EXIF data for a folder of images, however if images are removed or added from a directory, any cached results will be nullified and exiftool will run again. Cached EXIF data does not include supplemental flight metadata that you provide with metadata text files (see uas_metadata_make, such as the pilot's name or location name.

uas_getcache() retrieves the current cache directory, and uas_setcache() sets it. The default location is (*~/.R/uasimg*). When uas_setcache() is run with write = TRUE, the setting will be persistent across R sessions (generally recommended). uas_clearcache() will delete cached EXIF data, which is sometimes called for after a package update.

Functions

- uas_setcache(): Set cache directory
- uas_clearcache(): Clear cache directory

See Also

uas_info

---

uas_grp_flt *Parse an image collection into individual flights*

Description

Parse an image collection into individual flights

Usage

```
uas_grp_flt(
  x,
  interflt_val = 10,
  interflt_units = c("med_int", "secs")[1],
  init_fltnum = 1,
```

```
  min_images = 5,
  cross_dirs = TRUE,
  options = NULL,
  quiet = FALSE
)
```

Arguments

x	A list of class 'uas_info'
interflt_val	Time value between flights
interflt_units	Time units between flights, see details "med_int" or "secs"
init_fltnum	Initial flight number
min_images	Minimum number of images to be considered a flight
cross_dirs	Parse the images in directories collectively
options	Options and overrides for creating groups
quiet	Suppress messages

Details

cross_dirs means images in all directories will be examined when look for flights. This would be appropriate if the images from one flight were spread across multiple folders.

---

uas_info *Extract metadata info from UAS images*

Description

Extracts info from geotagged images taken from a drone

Usage

```
uas_info(
  img_dirs,
  ext = c("jpg", "jpeg", "tif", "tiff", "raw", "dng")[1:4],
  pattern = NULL,
  alt_agl = NULL,
  fp = FALSE,
  fwd_overlap = fp,
  cameras = NULL,
  metadata = "metadata.*\\.txt",
  path2name_fun = NULL,
  use_exiftoolr = TRUE,
  exiftool = NULL,
  exif_csv = NULL,
  cache = TRUE,
  update_cache = FALSE,
  quiet = FALSE
)
```

Arguments

img_dirs	Directory(s) where the image files reside
ext	File name extension(s)
pattern	A regex expression for files
alt_agl	The elevation above ground level in meters (optional for images with the relevative altitude saved)
fp	Compute image foot prints, T/F
fwd_overlap	Whether or not to compute the amount of overlap between one image and the next, T/F
cameras	Location of the cameras.csv file. Is NULL the package csv file will be used.
metadata	A filename pattern for a metadata file, or a metadata list object (see Details)
path2name_fun	A function to generate the default short name (see Details)
use_exiftoolr	Whether to use the exiftoolr package, logical
exiftool	The path to the exiftool command line tool (omit if on the OS path). Ignored if use_exiftoolr = TRUE.
exif_csv	The file name of a new csv file where the exif data will be saved (omit to make a temp one)
cache	Logical or a directory where EXIF data should be cached (see Details)
update_cache	Whether to update the cache
quiet	Don't show messages

Details

This will read the EXIF data from one or more directory(s) of image files and return a list of Image Collection Metadata' object(s) for each directory. The metadata objects will include the centroids of each image, and auto-generated flight metadata. Depending on the arguments used, the metadata objects may also include the estimated ground-footprints of individual images, and custom flight metadata fields (such as the name of the pilot). Extracting image locations requires that the images have embedded coordinates, also called geostamps. This is common with drone images, but some drone platforms require an extra processing step to geostamp the images.

An Image Collection Metadata object requires that all images be from the same sensor (camera). It is ok to have collections with different sensors in a single 'uas_info' object, but each individual collection should be from the same sensor. If you have a directory with mixed images from multiple sensors, there are two arguments you can use to specify which ones to include in the Metadata object. `ext` should be used to specify the image filename extensions (e.g., 'jpg' or 'tiff'). This will also prevent non-image files, such as text files or videos, from being read. `pattern` can be used to specify a regex expression that image filenames must match as an *additional* requirement. Neither `ext` nor `pattern` are case sensitive.

To include estimated footprints in the metadata object, pass `fp = TRUE`. This further requires that 1) the camera parameters are known (see below), and 2) the flight altitude above ground level is either recorded in the EXIF info or provided by `alt_agl` (in meters). If `alt_agl` is passed, it will override any altitude data in the EXIF info. Ground-footprints are estimates only. The modeled footprints assume the camera was at nadir (common in mapping work but there are exceptions), and that the

above ground altitude is the same as the above launch point altitude (generally true in flat areas, but not hilly areas). Estimated footprints are also only as accurate as the above-launch point altitude recorded in the EXIF data (which is typically the least accurate of the xyz coordinates).

Camera parameters are saved in a csv file called *cameras.csv*. The package ships with a CSV file containing the parameters for many popular drone cameras. If your drone camera is not in the database, you can create your own *cameras.csv* file (see `uas_cameras` for details) and pass the file name as the `cameras` argument. Or contact the package author to add your camera to the database.

`uas_info` uses a free command line tool called EXIFtool to read the EXIF data. You can install this by running `install_exiftool`. Alternately you can download exiftool from `http://www.sno.phy.queensu.ca/~phil/exiftool/`. If you're on Windows, after you download it you should rename the executable file from *exiftool(-k).exe* to *exiftool.exe*, and save it somewhere on your system's PATH (e.g., c:\Windows).

If you installed EXIFtool using the exiftoolr package on Windows, and the output includes strange `-- press ENTER --` statements, you can ignore them. These messages are generated by the exiftool executable. To get rid of them, find the exiftool executable (try `tools::R_user_dir("exiftoolr")`) and rename the exiftool(-k).exe' file to 'exiftool.exe' (if you don't see the '.exe.' extensions in File Explorer, you probably have file extensions hidden).

`metadata` is an optional argument to pass supplemental flight metadata that can not be extracted from the images (e.g., location name, pilot). For details, see the Vignette on Flight Metadata `vignette("flight_metadata", package = "uasimg")`.

`metadata` can also be a named list containing metadata fields / values. For supported field names, see `vignette("flight_metadata", package = "uasimg")`.

`metadata` can also be a filename regular expression (pattern) for a metadata text file saved in the same directory (for details on how to write a pattern expression, see `list.files`). This is the recommended way to enter metadata, because the little metadata text files move with the images.

If multiple metadata text files match the pattern expression, they will all be read. This allows you for example to have a boilerplate file called *metadata_org.txt* with organizational info (such as a contact person), and another called *metadata.txt* with info about that specific flight (e.g., the pilot or wind conditions). Metadata text files should be plain text in YAML format (the easiest way to create new metadata text files is using `uas_metadata_make`.) Each line should contain a key:value pair (with no quotes or delimiters). Lines starting with a hash or forward slash will be ignored. Example:

```
name_short: hrec_wtrshd2_flt03

name_long: Hopland Research and Extension Center, Watershed II Flight 3

data_url: https://ucdavis.box.com/s/dp0sdfssxxxxxsdf

pilot: Andy Lyons

description: These data were collected as part of a restoration monitoring program.

notes: We had to pause the mission about half way through as a hawk was getting close, hence there is a ti
of about 45 seconds. Pix4Dcapture was used as the mission planning software with an overlap of 75
```

path2name_fun can be a function to generate a default short name for the flight. The function should be written to accept one and only one argument - a directory path. This can be useful if the default short names should be constructed from pieces of the image directory path. See also uas_path2name_fun.

cache can be a logical value or the name of a directory where EXIF data gets cached. If cache = TRUE, the default cache directory will be used (~/.R/uasimg). The only information that gets cached is image metadata. Flight metadata is never cached (see the Vignette on Flight Metadata for a discussion of image and flight metadata). Cached EXIF data is linked to a directory of images based on the directory name and total size of all image files. So if images are added or removed from the directory, the cache will be automatically rebuilt the next time the function is run. update_cache is a logical value which forces an update of cached data when TRUE.

Value

An Image Collection Metadata object. This is a named list with elements for image centroids (as a sf data frame), footprints, total area, minimum convex polygon, total directory size, the data flown, and flight metadata.

See Also

uas_getcache, uas_report, uas_path2name_fun

---

uas_metadata_make *Create flight metadata text files*

Description

Create and/or edit flight metadata text files

Usage

```
uas_metadata_make(
  x,
  md_file = "metadata.txt",
  md_suffix = NULL,
  make_new = FALSE,
  overwrite = FALSE,
  open = TRUE,
  md_flds = uas_getflds(),
  md_template = NULL,
  read_uasinfo = FALSE,
  use_system_editor = FALSE,
  quiet = FALSE
)
```

Arguments

x	A character vector of directories, or a uas_info object
md_file	Name of the file to create
md_suffix	A suffix to append to the file names
make_new	Create new metadata file(s), Logical
overwrite	Overwrite existing metadata files, Logical
open	Open the file for editing
md_flds	A character vector of field names
md_template	A template metadata file
read_uasinfo	Incorporate existing flight metadata already saved in x (assuming x is a uas_info object)
use_system_editor	Whether to open metadata files in the operating system text editor (as opposed to RStudio), Logical
quiet	Suppress messages

Details

This creates, and/or opens for editing, supplemental metadata text file(s) in image folders.

If `make_new = TRUE`, metadata files will be created in each directory of `x`. `x` can be either a character vector or a `uas_info` object. The files will be named based on the value of `md_file`, with an option to append a suffix `md_suffix`. Both `md_file` and `md_suffix` should be of length 1 or equal to the number of `x`.

If `open = TRUE`, the text file(s) will be opened. To open existing metadata text files, let `open = TRUE` and `make_new = FALSE`. By default, files will be opened with whichever text editor is in use by R (see `file.edit`). To use the system text editor, let `use_system_editor = TRUE`.

To customize the fields that are added to flight metadata files, run `uas_setflds` first. To pre-populate some of the fields, pass the path to an existing metadata file as the value for `md_template`. `md_template` can either be a local file or online (example).

Value

A character vector of the external metadata text file(s) created.

See Also

`uas_getflds`, `uas_setflds`

uas_move *Move UAS images into sub-directories*

Description

Move UAS images into sub-directories

Usage

```
uas_move(
  x,
  flt = NULL,
  tree,
  outdir_base,
  req_all_fltmdflds = TRUE,
  create_dirs = "ask",
  imgs_action = c("copy", "move", "none")[1],
  imgs_prepend_fn = FALSE,
  write_metadata = TRUE,
  preview_only = FALSE,
  tb_action = imgs_action,
  map_action = c("copy", "move", "none")[3],
  quiet = FALSE,
  flt_idx = deprecated()
)
```

Arguments

x	A list of class 'uas_info'
flt	Flight(s) in x to process (character or numeric vector, default is all)
tree	Directory tree template filename or character vector, see Details
outdir_base	Output directory root
req_all_fltmdflds	Require all flight metadata fields in the directory tree template to be defined
create_dirs	Create the output directory tree
imgs_action	The action to take with images
imgs_prepend_fn	Whether th prepend image file names with datestamp (to ensure they'll be unique)
write_metadata	Write a metadtata.txt file in the output image folder
preview_only	Preview the directory tree only
tb_action	The action to take with thumbnail images saved in the default location
map_action	The action to take with the contents of the map folder
quiet	Suppress messages
flt_idx	‘r lifecycle::badge("deprecated")‘ Use ‘flt‘ instead

Details

req_all_fltmdflds means don't move anything unless all uas_info objects have all tokens in the directory tree

Value

A vector of directory locations where images were copied / moved

See Also

uas_info

---

uas_path2name_fun *Creates a function that converts a directory path to a flight name*

Description

Creates a function that converts a directory path to a flight name

Usage

```
uas_path2name_fun(idx_from_last = 1)
```

Arguments

idx_from_last	Indices of the directory path, see Details

Details

This will return a function that will parse a directory path into its constituent pieces and construct a name from one or more of those pieces. This can be useful for automatically naming flights based on the directory path in which the images are saved when running uas_info.

idx_from_last are indices of the subdirectories in the path, starting from the end. Hence if you wanted to construct names for a flight based on the very subdirctory (only), let idx_from_last = 1. If you want the names to be constructed based on the 2nd to last subdirectory, let idx_from_last = 2, and so on. If more than one number is passed for idx_from_last, the name returned will concatenate the pieces separated by underscores.

Note that this function does not actually convert a directory path to a flight name, but returns a function that will do the conversion.

Value

A function that returns a flight name

See Also

uas_info

Examples

```
## Not run:
data_dir <- "D:/Drone_Data/HREC/PostRiverFire2020/imgs/ebX_Flt02_postproc/img/msp"
flight_name_fun <- uas_path2name_fun(c(5,3,1))
flight_name_fun(data_dir)
ebee_flt2_info <- uas_info(data_dir, fp = FALSE, path2name_fun = flight_name_fun)

## End(Not run)
```

`uas_rename` *Rename UAS images*

Description

Rename UAS images

Usage

```
uas_rename(
  x,
  flt = NULL,
  name_template = "img{i}_{alt_agl}m_{camera_abbrev}_{Y}_{m}_{d}_{H}_{M}_{S}",
  all_lower = TRUE,
  preview = TRUE,
  confirm = TRUE,
  flt_idx = deprecated()
)
```

Arguments

`x`	A list of class 'uas_info'
`flt`	Flight(s) in x to process (character or numeric vector, default is all)
`name_template`	A template for generating the file names (see Details)
`all_lower`	Make file names all lowercase, Logical
`preview`	Preview the new names only
`confirm`	Confirm continue before changing file names
`flt_idx`	‘r lifecycle::badge("deprecated")‘ Use ‘flt‘ instead

Details

This function will rename image files on disk based on a naming template which can include placeholders (tokens) from image and/or flight metadata. This can be useful when you want to rename your image files based on some formula of date-time parts, AGL altitude, flight metadata fields, camera type, etc. Renaming image files can be helpful when you're doing analysis of individual images, but is generally not needed when you're doing photogrammetry whereby you're more likely to be using directory names to help you identify groups of images for stitching.

Caution is advised when using this function, because it will actually **rename your files!** Use `preview = TRUE` to test your naming template. When `preview = TRUE`, the function will return a tibble with the 'old' and 'new' names, but not actually change any file names.

`flt` allows you to specify a subset of image folders in `x` to process. You can pass a vector of flight names (use names(x) to see what those are) or integers.

When you're ready, set `preview = FALSE`. Aafter renaming files, you'll need to rerun `uas_info` on the directory(s) to update the info.

Value

A tibble showing the old and new names for each image.

File Name Template

`name_template` should be a pattern containing placeholders (or 'tokens') in curly brackets. When you run the function, the tokens will be swapped out for actual values.

For example, if a filename is *DJI_0213.JPG* and the name template was `"img_{Y}-{m}-{d}_{H}-{M}-{S}"`, the file would be renamed something like *img_2018-06-17_13-04-46.jpg*, where the numbers for date and time are extracted from the image EXIF data. Supported tokens you can use include:

- `{i}` an integer starting from 1 and going up, padded with enough leading zeros to accommodate all values
- `{Y}` year (4-digit)
- `{y}` year (2-digit)
- `{m}` month (2-digit)
- `{d}` day (2-digit)
- `{j}` Julian day
- `{H}` hour (2-digits)
- `{M}` minutes (2-digits)
- `{S}` seconds (2-digits)
- `{camera_abbrev}` an abbreviated name of the camera
- `{alt_agl}` altitude above the launch point (usually in meters). You can indicate rounding by adding a comma and the number of decimal places. For example `{alt_agl,0}` would round the AGL value to the nearest whole number. This option is only available for images where the relative altitude is saved in the EXIF data (which excludes most multi-spectral images).

In addition, name templates can include any flight metadata field that has been defined. For example if the flight metadata information includes fields for `proj` (project abbreviation) and `loc` (location), the name template could include `{proj}` and `{loc}`.

When creating your name template, remember:

1) All file names must come out unique. An easy way to ensure this is to include `{i}` in your name template, which will be replaced with a sequence of integers.

2) File names should not include characters that aren't allowed for file names. These include `< > : "/ \ | ? *`. If any of these characters are found in name_template, it will be rejected.

3) `name_template` should not contain an extension

See Also

uas_info, uas_metadata_make

---

uas_report *Flight summaries*

Description

Creates image collection summaries for individual flights (folders)

Usage

```
uas_report(
  x,
  flt = NULL,
  group_img = FALSE,
  thumbnails = FALSE,
  show_local_dir = TRUE,
  units = c("imperial", "metric")[1],
  report_title = "Flight Summary",
  attachments = c("mcp_kml", "ctr_kml")[0],
  pts_col = NULL,
  output_dir = NULL,
  create_dir = TRUE,
  output_file = NULL,
  overwrite_html = FALSE,
  open_report = FALSE,
  self_contained = TRUE,
  tbm_use = FALSE,
  tbm_overwrite = FALSE,
  tbm_size = 480,
  tbm_src = c("Google", "Stadia")[1],
  tbm_api_key = NULL,
  tbm_exp = 0.2,
  report_rmd = NULL,
  header_html = NULL,
  footer_html = NULL,
  use_tmpdir = FALSE,
  quiet = FALSE,
  show_gps_coord = lifecycle::deprecated(),
  png_map = lifecycle::deprecated(),
  overwrite_png = lifecycle::deprecated(),
  png_exp = lifecycle::deprecated(),
  google_api = lifecycle::deprecated(),
  col = lifecycle::deprecated()
)
```

Arguments

x	A list of class 'uas_info'
flt	Flight(s) in x to process (character or numeric vector, default is all)
group_img	Group images within ~1m of each other into 1 point
thumbnails	Display thumbnail images, logical
show_local_dir	Show the local image directory, logical
units	imperial or metric, character
report_title	Title to appear at the top of the summary
attachments	Supplementary files to create and link to the flight summary, see Details.
pts_col	Color value(s) of the centroids and/or footprints
output_dir	If NULL, then will be placed in a 'map' sub-directory of the images
create_dir	Create the output directory if it doesn't exist
output_file	Name of the HTML file. If NULL a default based on the name of the input directory is chosen.
overwrite_html	Overwrite existing HTML files without warning, logical
open_report	Open the HTML file in a browser
self_contained	Make the output HTML file self-contained
tbm_use	Whether to create a PNG version of the map. May be logical, or dimensions of the output image in pixels (see Details)
tbm_overwrite	Overwrite existing PNG files without warning, logical
tbm_size	The size of a square flight thumbnail image in pixels, number
tbm_src	The API service to use to get the flight thumbnail background image, see Details.
tbm_api_key	API key for Google Static Maps or Stadia, see Details.
tbm_exp	A proportion to expand the bounding box of the PNG map, see Details.
report_rmd	Rmd template used to generate the HTML file. See Details.
header_html	A HTML file name or URL to use as the header
footer_html	A HTML file name or URL to use as the footer
use_tmpdir	Use the temp dir for processing
quiet	TRUE to supress printing of the pandoc command line
show_gps_coord	‘r lifecycle::badge("deprecated")‘ Does nothing
png_map	‘r lifecycle::badge("deprecated")‘ Use tbm_use
overwrite_png	‘r lifecycle::badge("deprecated")‘ Use tbm_overwrite
png_exp	‘r lifecycle::badge("deprecated")‘ Use tbm_exp
google_api	‘r lifecycle::badge("deprecated")‘ Use tbm_api_key
col	‘r lifecycle::badge("deprecated")‘ Use pts_col

Details

This will generate HTML report(s) of the images in the UAS metadata object based.

`group_img` determinies whether images at the same location are represented by a single point on the map. This is common with multi-spectral sensors that take generate multiple images per location. 'Same location' is determined by looking at the 5th decimal place of the x and y geographic coordinates (~1m).

`units` defines whether the units for the flight area, above ground altitude, and GSD are reported in imperial or metric units.

If no value for `output_dir` is passed, the report will be saved in a sub-directory of the image directory called 'map'. This sub-directory will be created if `create_dir = TRUE`.

`self_contained` determines whether the HTML file(s) created will have all the JavaScript and CSS files embedded in the HTML file itself, or saved in a subdirectory called 'libs'. If saving several reports to one output directory, If saving multiple HTML reports to the same output directory, passing `self_contained = FALSE` is more efficient

The HTML report is generated from a RMarkdown file. If you know how to edit RMarkdown, you can modify the default template and pass the filename of your preferred template using the `report_rmd` argument.

`tbm_use` determines whether a thumbnail image of the flight will be downloaded from Google or StadiaMaps, and saved in `output_dir`. Although this thumbnail image of the flight is not be displayed in the flight summary report, it is used when generated a Table-of-Contents for a series of flights, and can be useful as a standalone quick glance of the flight in Windows Explorer, GitHub repos, etc.

Note that both Google Maps and StadiaMaps require an API key (that you pass using `tbm_api_key`). If you don't have an API key for one of these services, then you can't download a thumbnail image of the flight. To get an API key for the Google Static Maps service, see https://developers.google.com/maps/documentation/maps-static/ (there are also a number of tutorials available). Note that Google does require a credit card or their APIs, but the monthly quota before you get charged should be more than enough if all you're doing is downloading thumbnail images for your drone flights. To get an API key for StadiaMaps (no credit card required), start here. See the ggmap package for details.

The flight thumbnail image will be created at the dimensions given by `tbm_size`. `tbm_exp` is a percentage of the flight bounding box that will be used as a buffer around the background map. If the map seems too cropped, or you get a warning message about rows removed, try increasing it. `pts_col` can be used to pass a single color or vector of custom colors for the image locations (the default is a rainbow color ramp).

`attachment` specifies which supplementary files to create and link to the flight summary. Choices are `ctr_kml` and `mcp_kml` for KML versions of the camera locations and MCP (minimum convex polygon around all images). These KML files will be created in the same output directory as the flight summary.

Value

The HTML file name(s) of the flight summaries generated

See Also

uas_info, uas_exp_kml,

---

uas_setflds *Manage supplemental metadata fields*

Description

Set and view a default set of supplemental metadata fields

Usage

```
uas_setflds(flds = NULL, reset = FALSE, quiet = FALSE)

uas_getflds()

uas_flds_oem()
```

Arguments

flds	Character vector of metadata fields
reset	Use a default set of metadata fields
quiet	Suppress messages

Details

These functions allow you to define a default set of supplemental metadata fields that can be assigned image collections either as a function argument, or a small text file placed in the directory with the images. Note in this context 'supplemental metadata' refers to bits of information that can not be automatically extracted from the images themselves (e.g., the name of the project or pilot.)

Creating a default set of supplemental metadata fields does not make them mandatory, nor does it limit the metadata fields you can use. The default set serves as a template when a new metadata.txt file is created.

Functions

- uas_setflds(): Set default fields for image collection metadata.
- uas_getflds(): Get default supplemental metadata field names.
- uas_flds_oem(): Get a standard set of image collection metedata fields

See Also

uas_getcache, uas_info

uas_thumbnails_make *Create image thumbnails*

Description

Move UAS images into sub-directories by group

Usage

```
uas_thumbnails_make(
  x,
  flt = NULL,
  output_dir = NULL,
  tb_width = 400,
  rotate = FALSE,
  overwrite = FALSE,
  use_magick = FALSE,
  stats = FALSE,
  quiet = FALSE,
  flt_idx = deprecated()
)
```

Arguments

x	A list of class 'uas_info'
flt	Flight(s) in x to process (character or numeric vector, default is all)
output_dir	Output directory
tb_width	Thumbnail width
rotate	Rotate the thumbnails by the camera yaw, Logical
overwrite	Overwrite existing files
use_magick	Use functions from the magick package
stats	Report the amount of time it takes to create each thumbnail, logical
quiet	Suppress messages
flt_idx	‘r lifecycle::badge("deprecated")‘ Use ‘flt‘ instead

Details

This will create thumbnail images for the drone images in x. The default output folder is a sub-directory of each image folder called *map/tb*, which will be created if needed. This location can be overridden with output_dir. The dimensions of the thumbnails is determined by tb_width, from which the height is set automatically.

flt allows you to specify a subset of image folders in x to process. You can pass a vector of flight names (use names(x) to see what those are) or integers.

`rotate` will rotate the thumbnails by the camera yaw. This can make it easier to match up ground features when viewing the thumbnails in a flight report. Note when thumbnails are rotated the `tb_width` parameter sets the width of the image *before* the rotation. The width and height of the rotated thumbnail will vary according to the angle of rotation. If your rotations look off feel free to contact the package author, as this feature is still experimental (i.e., some drones record the yaw of the drone and the yaw of the gimbal camera separately).

Thumbnail files will be given an 8-character suffix that looks random but is actually generated from the image contents. This is to prevent clashes when thumnbail files from different flights are 'gathered' into a single folder attached to a Table of Contents folder (see `uas_toc`).

If `use_magick = TRUE`, it will use resizing functions from the magick package. This is slower than the equivalent functions from the `imager` package (the default), but may be necessary if you are processing TIFs and don't have ImageMagick installed on your computer (which `imager` requires to read TIFs).

Value

A named list (one element for each directory processed) of thumbnail files created in the output directory

See Also

uas_report

`uas_toc` *Generate a Table of Contents of flight summaries*

Description

Generate a Table of Contents of flight summaries

Usage

```
uas_toc(
  html_reports,
  output_dir = ".",
  output_fn = "index.html",
  gather_dir = NULL,
  toc_title = "UAS Flight Summaries",
  toc_desc = NULL,
  fltmap_show = TRUE,
  fltmap_kml = FALSE,
  fltmap_base = NULL,
  header_html = NULL,
  footer_html = NULL,
  overwrite_toc = FALSE,
  overwrite_gather = FALSE,
  open_toc = FALSE,
```

```
  quiet = FALSE
)
```

Arguments

html_reports	HTML file names of flight summaries
output_dir	Output directory
output_fn	Output file name
gather_dir	Subdirectory of output_dir where HTML files will be copied
toc_title	Title to show at the top of the Table of Contents
toc_desc	A short description to appear under the title
fltmap_show	Show a map of all flight areas, logical
fltmap_kml	Create a KML of all flight areas, logical
fltmap_base	A list object containing of background KML files and their symbology for the flight map, see Details.
header_html	HTML file name or URL to use as a page header
footer_html	HTML file name or URL to use as a page footer
overwrite_toc	Overwrite existing file, logical
overwrite_gather	Subdirectory of output_dir where HTML files will be copied
open_toc	Open the table of contents in a browser when done, logical
quiet	Suppress messages, logical

Details

This function generates a master Table of Contents HTML page for a set of individual flight summaries created by uas_report, saving the TOC in output_dir. The TOC page can either point to the flight summary reports where they are, or copy them to a single directory.

htmls_reports should be a vector of HTML filenames including the full path (i.e., the object returned by uas_report). Reports will be appear in the Table of Contents in the same order.

The default behavior is to create links to the flight summary reports where they currently are. The function will attempt to create relative paths from the output_dir to the locations of the htmls_reports. At a minimum, this requires output_dir and htmls_reports to be on the same volume. Preferably output_dir is be a parent directory of htmls_reports. If your HTML reports are scattered across many directories, consider using gather_dir which will put them all in one place.

To copy the flight summary reports to a common location, pass a value for gather_dir. gather_dir should be a *sub-directory* of output_dir where the HTML files in htmls_reports (and any associated files such as thumbnails) will be copied. gather_dir should be relative to output_dir (not an absolute path). To copy HTML files in output_dir itself (i.e., not a sub-directory), set gather_dir = '.' If gather_dir does not already exist, R will attempt to create it.

If fltmap_show = TRUE, the Table of Contents will include an interactive map showing the flight areas of all the flight summaries on the page. fltmap_base is an optional list of lists that you can

use to have additional layers appear on the map. Currently only polygon and polyline KML files are supported as additional base layers. Each list element should be a named list with three elements: *kml_fn* (a KML file name), *color* (a named color or hex code for the outline), and *weight* (outline thickness in pixels).

`header_html` and `footer_html` allow you to specify filenames that contain HTML text for a page header and footer (i.e., for branding elements).

See Also

`uas_report`

`uas_worldfile` *Create world and projection files for UAS images*

Description

Create world files and projection files for UAS images

Usage

```
uas_worldfile(
  x,
  flt = NULL,
  idx = NULL,
  aux.xml = TRUE,
  wld = FALSE,
  wldext = "auto",
  prj = FALSE,
  rotated = TRUE,
  quiet = FALSE
)
```

Arguments

`x`	A list of class 'uas_info'
`flt`	Which flight(s) in x to process (character or numeric vector, default is all)
`idx`	Which rows in x to process (default is all)
`aux.xml`	Create an aux.xml file, logical. See details.
`wld`	Create a world file, logical. See details.
`wldext`	Extension for the world file, character. Ignored if `wld = FALSE`.
`prj`	Create a prj file, logical. See details.
`rotated`	Compute parameters that replicate the camera Yaw.
`quiet`	Show messages.

Details

This function creates sidecard world files and/or projection files that are required to view raw images in GIS software such as ArcGIS and QGIS.

The parameters this function uses to generate world file are taken exclusively from the image metadata, including the altitude above the launch point, the focal length, and so on. They should be considered **approximate at best**. For a more accurate image placement, process the images with photogrammetry software.

To generate world files, you must tell it to compute footprints when generating the flight metadata object. In other words, when you run `uas_info`, pass `fp = TRUE`. If you've already generated a metadata object for the flight, you may also need to pass `update_cache = TRUE`. Computing footprints requires an altitude for each image, which not all drones support (especially multispectral cameras). For details see `uas_info`.

This function has been tested with JPG files from several DJI cameras. It has not yet been adapted for TIF files from multispectral cameras, and may not work with those formats (contact the package author if you want to try).

`aux.xml` files are ESRI auxillary files for raster layers. If your objective is to open the images in ArcGIS or QGIS, then generating the `aux.xml` files (the default setting) should be all you need. To be read by GIS software, they should have the same name as the image file with 'aux.xml' added on. aux.xml files are capable of storing a lot of info about a raster layer, including statistics, the rotation info, coordinate reference system, and other stuff. The files generated by this function will only contain projection and rotation info only. Note also **any existing aux.xml files will be overwritten**. aux.xml files are the only sidecar file that ArcGIS software seems to support for reading the projection info.

World files are small text files with extensions `jpw` and `tfw` for JPG and TIF files respectively. To be read by GIS software, they must have the same basename as the image and be saved in the same directory. Both ArcGIS and QGIS read world files, however they are not needed if the same info is available in an aux.xml file.

`prj` files contain just the Coordinate Reference System info. They do not seem to be recognized for rasters by ArcGIS, however QGIS picks them up.

Value

A list of vectors of filenames generated.

See Also

`uas_info`

Index