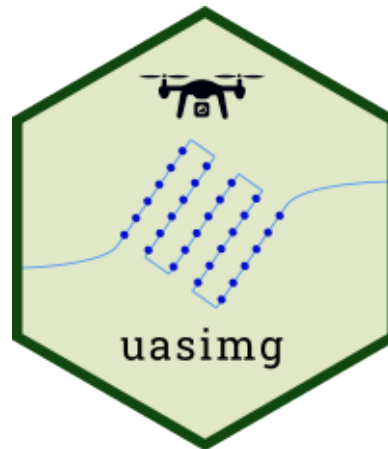


Drone Image Utilities for R

Andy Lyons

June 2024



<https://ucanr-igis.github.io/uasimg>



UNIVERSITY OF CALIFORNIA
Agriculture and Natural Resources

Informatics and GIS Program

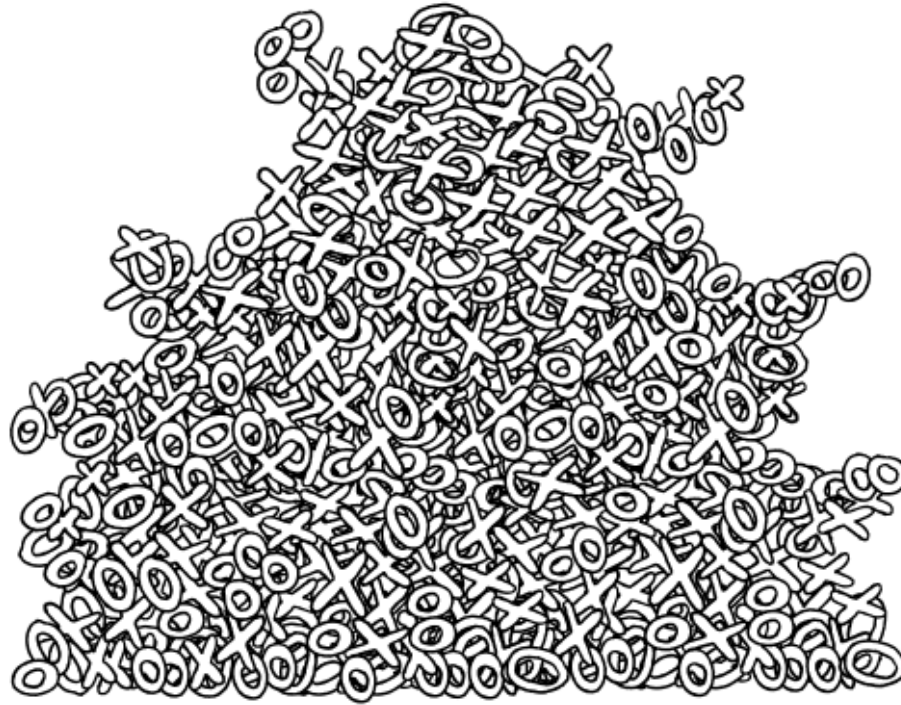
Motivation

Drone mapping takes a lots of images...



...but it's not just images that you have to keep track of!

flight logs
study area
outline
GCP
coordinates
telemetry files
field data
other photos
video files
basemap layers



“ I know it's here somewhere. ”



“Yeah, I remember hearing someone
from UCSB flew that site.”

“Did you ask Brandon?”

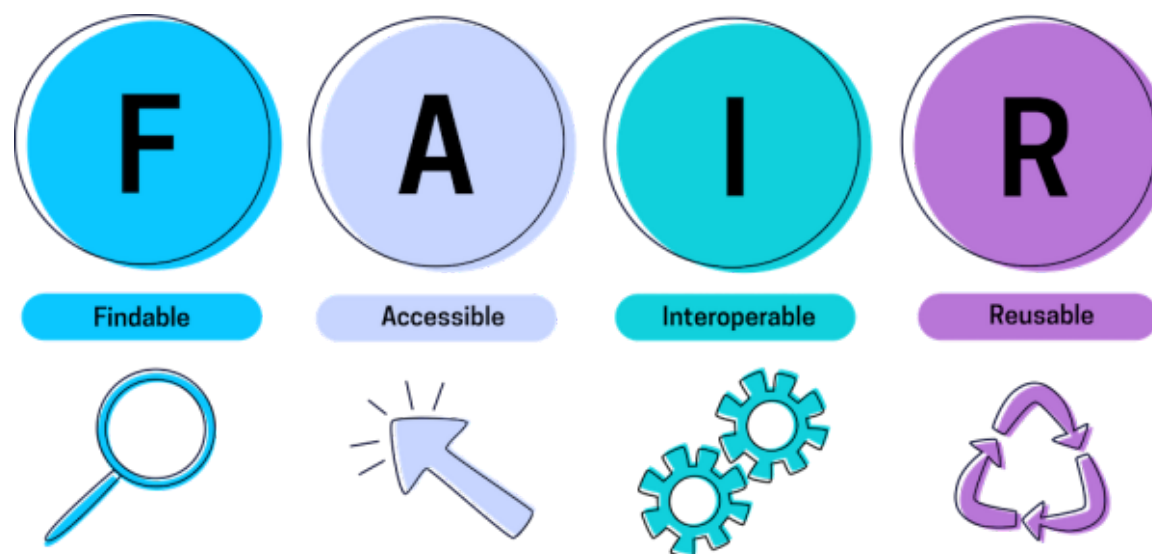
“I found the images, but I can't reproduce the stitching. It looks funny.”

“Maybe your overlap was too low?”

“ We have to report how many acres we’ve flown with each type of camera for the USDA report.

”
Whatdyathink?

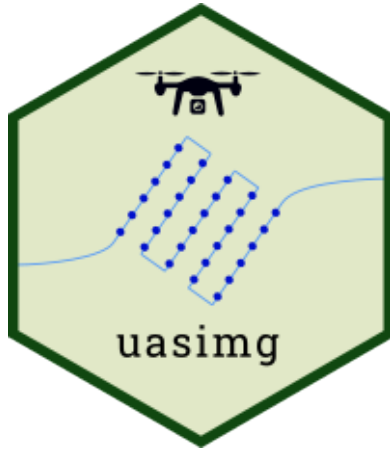
How do we get our drone data to be FAIR compliant?



uasimg Overview

R package with functions to help you:

- create catalogs of your images
- record your flight metadata
- create GIS layers of your data collections
- store your images in a directory structure that will allow you to find them again
- share your metadata and data with others
- prepare individual images for visualization and analysis in other applications
- deal with annoying data management tasks



Sample Data Catalogs

UC Merced Vernal Pool and Grassland Reserve

Point Pinole Regional Shortline Eucalyptus Grove

Hopland River Fire Revegetation Study

Origins

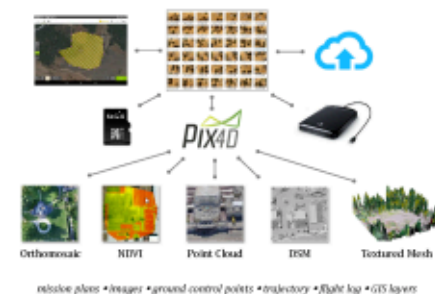
Open Source Tools for Drone Data Management

Andy Lyons, Jacob Flanagan, Sean Hogan, and Maggi Kelly
University of California Division of Ag and Natural Resources



Introduction

Modern photogrammetry software has dramatically improved the quality and speed of generating 2D and 3D data products from drone imagery, however a significant amount of data management remains before and after the stitching process. To address the increasing demands of managing data for a busy drone program within a team environment, we developed a series of tools using open source software.



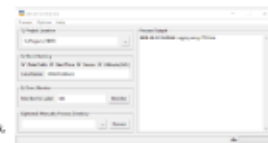
Foundations: Standard Directory Structure

The initial step was to develop a project directory structure and set of naming conventions that were flexible, extensible, and portable. Then trial and error, we developed a generic directory tree that accommodated nearly all drone projects.



IGIS Drive Monitor

The *IGIS Drive Monitor* is designed for field use. It copies drone data from SD cards to a backup location in an autonomous way. It will search for drives with a user specified label and automatically find all images, consolidate, copy and rename them according to a block naming scheme.



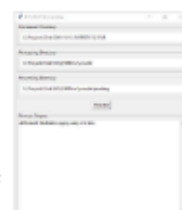
Through this, it creates a standardized data structure that is portable and can be easily processed on a remote computer (i.e., server).

IGIS Drive Monitor is written completely in Python 3 and uses open source libraries for reading image Exif and monitoring for external drive insertion. Optionally, it can also interact with external software via command scripts and create processing instructions for a remote processing computer for data verification and processing (e.g. Pix4D), respectively.

IGIS | University of California
Agriculture and Natural Resources | Informatics and GIS Program
<http://igs.ucanr.edu>

IGIS Pix4D Controller

Intended to be installed on a server, the *IGIS Pix4D Processing Controller* software ingests newly added drone data from a specified directory given a processing flag and instructs an instance of Pix4D (it is assumed a working copy of Pix4D is installed on the same machine) to automatically process project data. A data structure in the form given by the *IGIS Drive Monitor* allows the software to automatically find all the images belonging to flight block and a build in customizable sensor library relays which processing template Pix4D should use for processing. In addition, if human interaction is necessary during the processing (i.e., applying Ground Control Points), the software will place these data sets into a "Pending Directory" and wait until instructed to continue through processing.

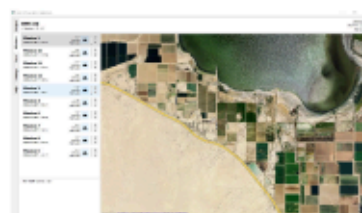


IGIS Pix4D Processing Controller was written in Python 3 with open source libraries for reading exif information from images. In addition, it uses command scripts to interact with Pix4D (a working copy of Pix4D should be installed on the same machine as this scripting software).

IGIS Drone Data Management Logbook

The *IGIS Drone Data Management Logbook*, written in C++, is an interactive data manager that aids a drone pilot and field crew in syncing data from SD cards, creating standardized data structures, visualizing image positions and mission footprints and automatically create processing instructions (i.e., "pix4d"). The data structure allows for larger projects that are made up of multiple missions to follow an ideal workflow of partial processing of individual missions and then combining them for a larger result.

A sensor pairing feature allows multiple sensors to be flown at the same time - Additional sensor imagery are trimmed to the same footprint and time frame as the main sensor. Additionally, field notes can be recorded for each mission and kept together within the data structure.



The *IGIS Drone Data Management Logbook* is a work in progress and written in C++ using libraries from Qt, libtiff, libexif and WPD Application Programming Interface.

R-package: uasimg

The *uasimg* package for R contains functions to:

- create HTML catalogs for a folder of images
- estimate actual overlap, GSD, & mission area
- compute and export image centroids, footprints, and the mission MCP
- move images into folders by flight
- launch functions from the context menu in Windows Explorer



https://is.gd/uas_data_aag2019

Why R?



- full featured scripting language
- wrappers for open source libraries (exiftool, gdal, geos)
- good reporting engine
- good deployment system
- good tools to create GUIs

Image Metadata vs Flight Metadata

Image metadata

- camera make and model, dimensions, GPS coordinates, yaw, focal length, aperture, etc.
- stored in the images themselves
- essential for photogrammetry

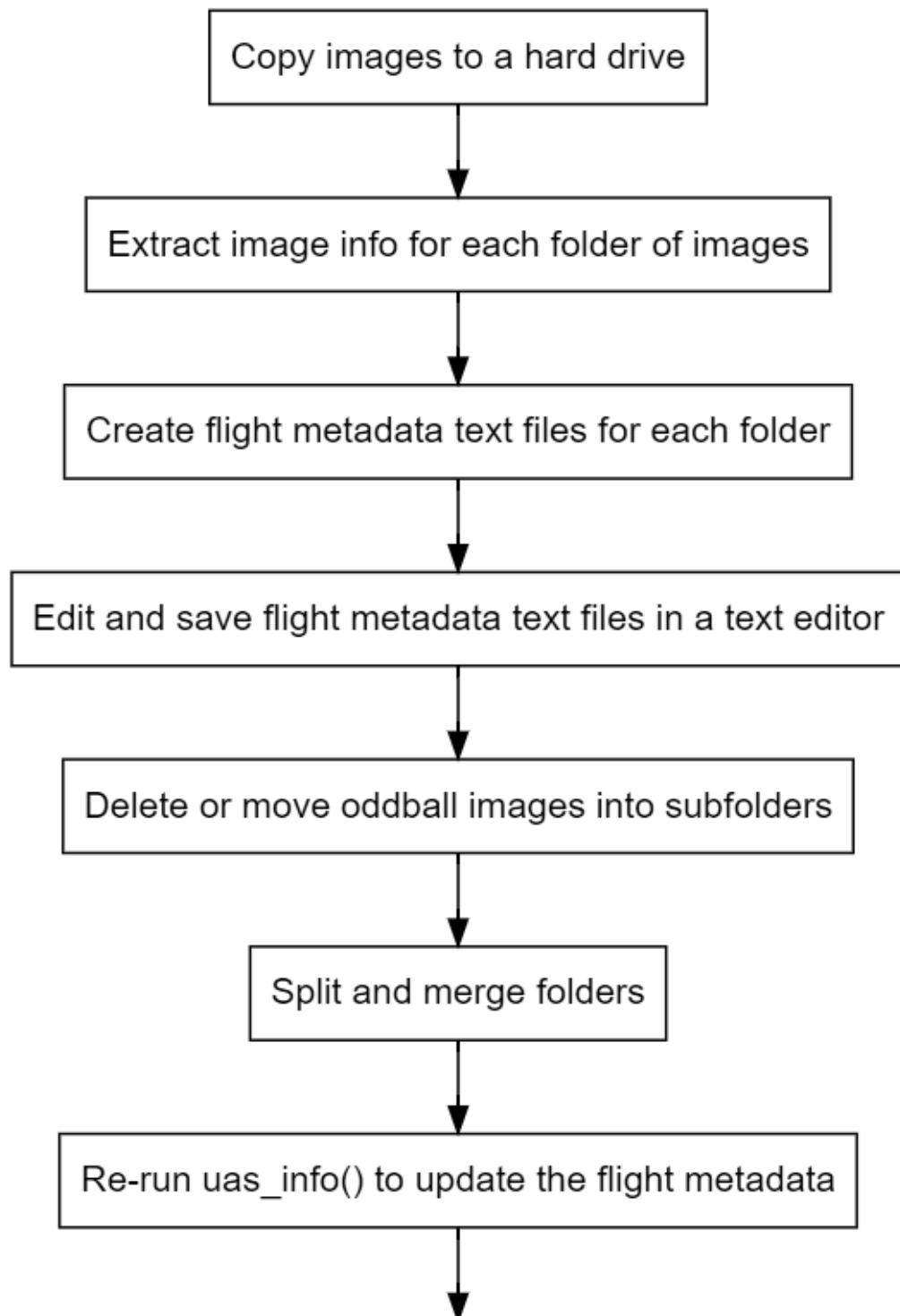
Flight metadata

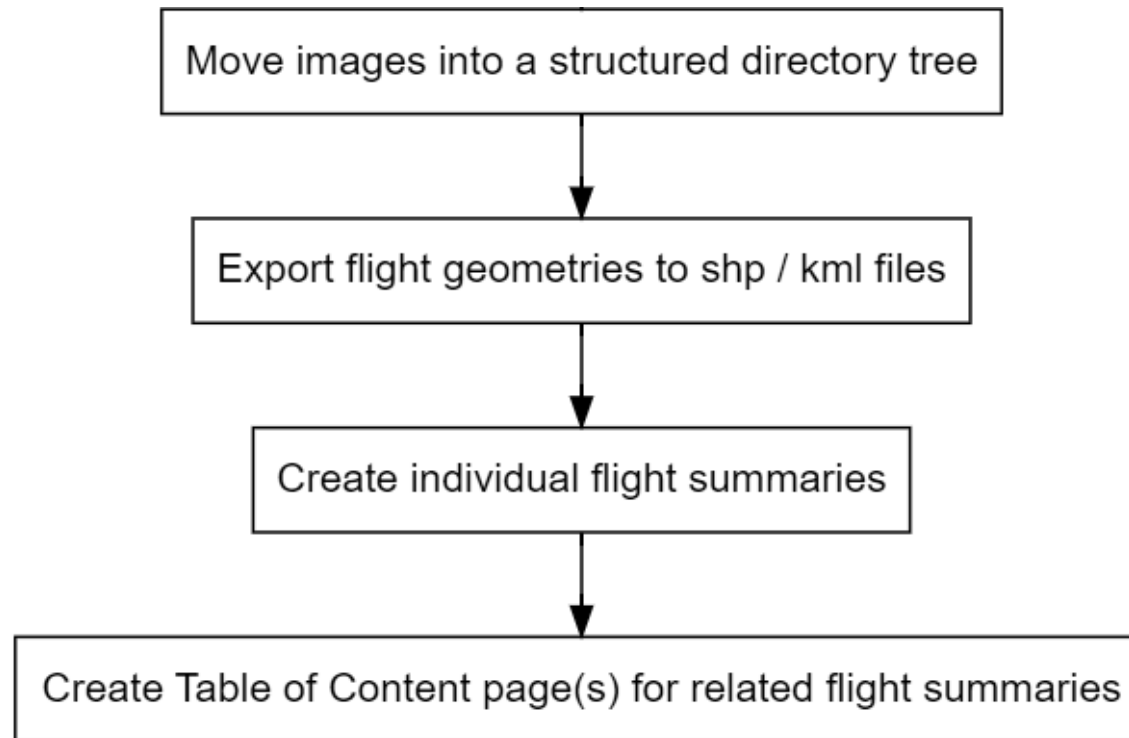
- depends on the project / organization
- examples: name of the site, project, pilot, aircraft, flight notes
- not critical for photogrammetry, but essential for good data management

- uasimg uses flight metadata for i) **cataloging**, ii) **searching**, and iii) **creating directory trees**
- see also: [Flight Metadata Vignette](#).

Cataloging Workflow

https://ucanr-igis.github.io/uasimg/articles/catalog_wrkflw.html





Create Flight Info Objects

Step 1. Extract Image Metadata

```
ptpinole_info = uas_info("d:/uas/ptpinole/west/march2020/dcim")
```

This will:

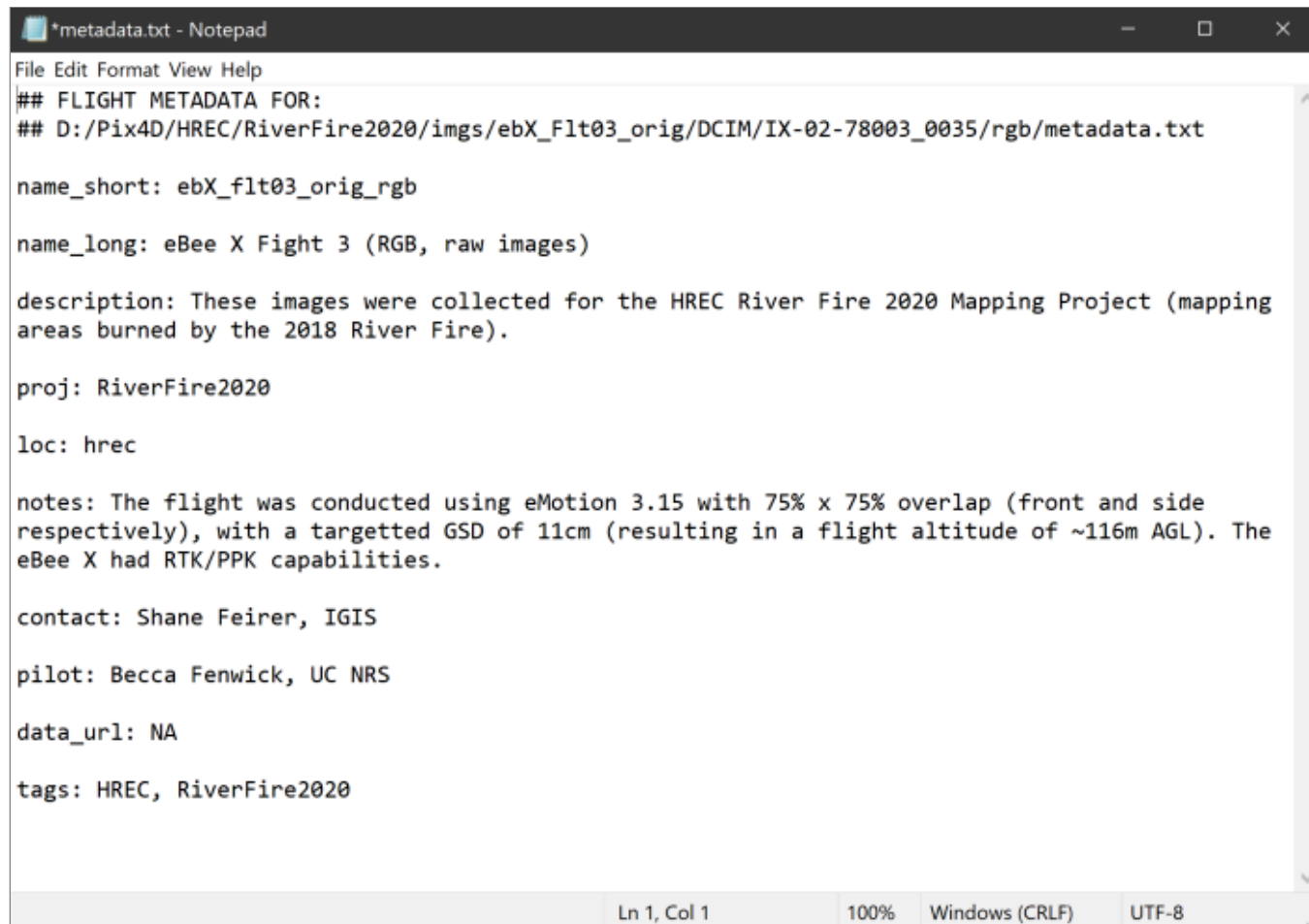
1. extract the image coordinates
2. read the sensor type, location, yaw, focal length, etc.
3. compute image footprints
4. create an minimum convex polygon (MCP) for the entire flight
5. read flight metadata
6. cache all the results
7. return a “Flight Info” object

You can process multiple directories at once

A “Flight Info” object by itself is not very useful!

Step 2. Entering Flight Metadata

Recommended way to record flight metadata - text editor



```
*metadata.txt - Notepad
File Edit Format View Help
## FLIGHT METADATA FOR:
## D:/Pix4D/HREC/RiverFire2020/imgs/ebX_Flt03_orig/DCIM/IX-02-78003_0035/rgb/metadata.txt

name_short: ebX_flt03_orig_rgb

name_long: eBee X Fight 3 (RGB, raw images)

description: These images were collected for the HREC River Fire 2020 Mapping Project (mapping
areas burned by the 2018 River Fire).

proj: RiverFire2020

loc: hrec

notes: The flight was conducted using eMotion 3.15 with 75% x 75% overlap (front and side
respectively), with a targetted GSD of 11cm (resulting in a flight altitude of ~116m AGL). The
eBee X had RTK/PPK capabilities.

contact: Shane Feirer, IGIS

pilot: Becca Fenwick, UC NRS

data_url: NA

tags: HREC, RiverFire2020

Ln 1, Col 1    100%    Windows (CRLF)    UTF-8
```

- Any field name allowed. Defaults:

```
uas_flds_oem()
```

```
## [1] "name_short" "name_long" "description" "proj"
"loc"
## [6] "pilot" "contact" "uav" "data_url"
```



```
"tags"  
## [11] "notes"
```

Flight Metadata Helper Function

You don't have to start from scratch!

```
uas_metadata_make(ptpinole_info)
```

- create a *metadata.txt* file for every image directory in x
- you can customize field names with `uas_setflds()`
- can pre-populate values from a template ([example](#))
- open them in text editor

TIP: Re-run `uas_info()` after you edit flight metadata text files!

Organize Your Files

Parse an image collection into individual flights

```
uas_grp_flt()
```

Create separate Flight Info objects if you have:

- Multiple flights in one folder
- One flight spread across multiple folders

Rename Files

```
uas_rename()
```

Rename files based on a template, such as:

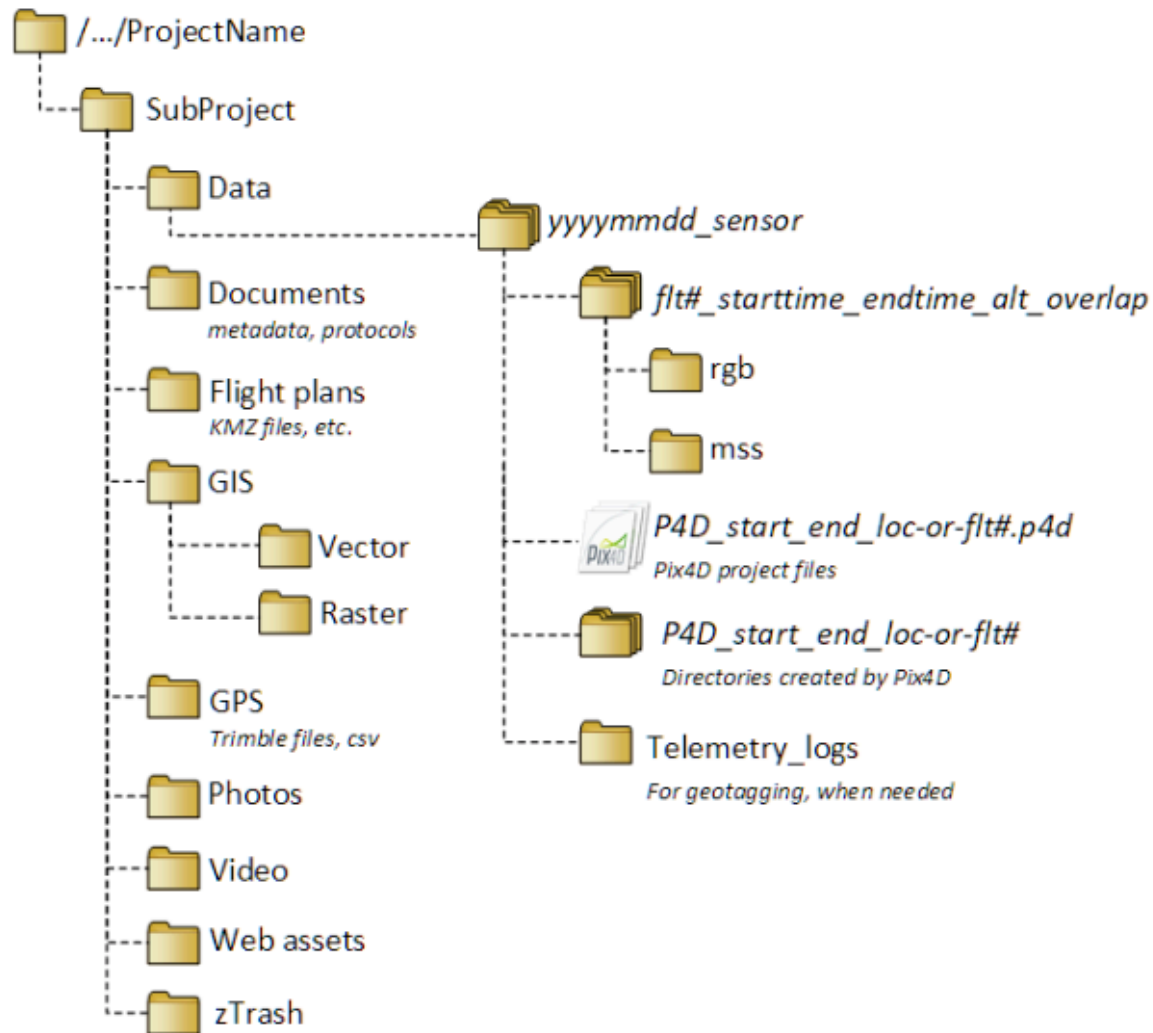
{proj}-{loc}_{camera_abbrev}_{Y}{m}{d}_{H}{M}{S}

Fields available:

- `{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

- + all your flight metadata

Move Your Data into Standard Directory Trees



You can move your data by hand or with a function:

```
uas_move()
```

Options:

- move or copy images
- create the directory tree from a template text file:

```
{proj}/{loc}/{subloc}/imgs/{flt_date}/{flt_start}-{flt_end}_{camera_abbrev}  
{proj}/{loc}/{subloc}/gis  
{proj}/{loc}/{subloc}/p4d  
{proj}/{loc}/{subloc}/documents
```

- placeholders in {} will be replaced with flight and/or image metadata
- you can use *any* flight metadata field and a subset of image metadata fields
- the first directory should be where you want the images to go

- you can create additional directories for other parts of your workflow

Export Flight Geometry as GIS files

```
uas_exp_kml(ptpinole_infox)  
uas_exp_shp(ptpinole_info)
```

Geometries you can export:

- image center points
- modelled footprints
- flight area (minimum convex polygon of all the images)

Catalog Your Data

Create Individual Flight Summary Reports

Create HTML flight summary for one or more Flight Info object(s):

```
uas_report(ptpinole_info)
```

Report options:

- report title
- map of image locations
- show image thumbnails
- group image locations into 'clusters'
- data URL
- custom header / footer
- directory where the images are saved on your local machine

- hyperlinks to download the KML files
- create an image thumbnail of the entire flight area (requires a Google Maps or Stadia API key)



Group Flight Summaries into Catalogs

Feed a list of Flight Summary HTML files.

```
uas_toc(c("d:/uas/ptpinole/west/march2020/dcim/map/ptpinole_mar2020.html",  
          "d:/uas/ptpinole/west/jul2020/dcim/map/ptpinole_jul2020.html",  
          "d:/uas/ptpinole/west/oct2020/dcim/map/ptpinole_oct2020.html"))
```

It will extract flight metadata fields from the HTML file for the table-of-contents.

Function options:

- title & HTML header/footer
- copy all the individual Flight Summary HTML files to one place
- show a map of all the flight areas
- include hyperlinks to download all flight areas as a single KML

Example: <http://uas.igis-data.click/hrec/watershed2/>

Sample Script

```
library(uasimg)

## Define the subdirectories that have the images
hrec_dirs <- c("D:/Pix4D/HREC/Watershed1/Data/2017-01-16_X5/Flight01_1514_1526_400ft",
              "D:/Pix4D/HREC/Watershed1/Data/2017-01-16_X5/Flight02_1532_1540_400ft")
file.exists(hrec_dirs)

## Create/edit metadata text files
# uas_metadata_make(hrec_dirs, make_new = TRUE, overwrite = FALSE, open = TRUE,
use_system_editor = TRUE)

## Extract image and flight metadata
hrec_info <- uas_info(hrec_dirs, fp = TRUE)

## Create image thumbnails (optional)
uas_thumbnails_make(hrec_info, rotate = TRUE, stats = TRUE, overwrite = FALSE)

## Create flight summary reports
hrec_rpts <- uas_report(hrec_info,
                      thumbnails = TRUE,
                      attachments = c("mcp_kml", "ctr_kml"),
                      footer_html = "D:/Pix4D/catalog/headers_footers/footer_igis.html",
                      tbm_use = TRUE,
                      tbm_src = "Google",
                      tbm_api_key = getOption("ANDYS_GOOGLE_STATIC_MAPS_KEY"),
                      tbm_overwrite = FALSE,
                      group_img = FALSE,
                      show_local_dir = TRUE,
                      overwrite_html = TRUE,
                      open_report = TRUE)

## Create a TOC
hrec_toc <- uas_toc(hrec_rpts,
                   toc_title = "HREC Watershed 1, January 2017",
```

```
output_dir = "D:/Data/uas.igis-data/hrec/watershed1",  
gather = ".",  
fltmap_base = list(list(kml_fn = "D:/Pix4D/HREC/Baselayers/hopland.kml",  
                        color = "yellow",  
                        weight = 2)),  
  
fltmap_kml = TRUE,  
footer_html = "D:/Pix4D/catalog/headers_footers/footer_igis.html",  
open_toc = TRUE,  
overwrite_toc = TRUE)
```

Output: <http://uas.igis-data.click/hrec/watershed1/>

Utilities for Individual Images

World Files

‘World Files’ are plain-text files (also called sidecar files) with location and projection info.

GIS software like ArcGIS Pro and QGIS use these files to display images in the proper place and size.











Generate with:

```
uas_worldfile(ptpinole_info)
```

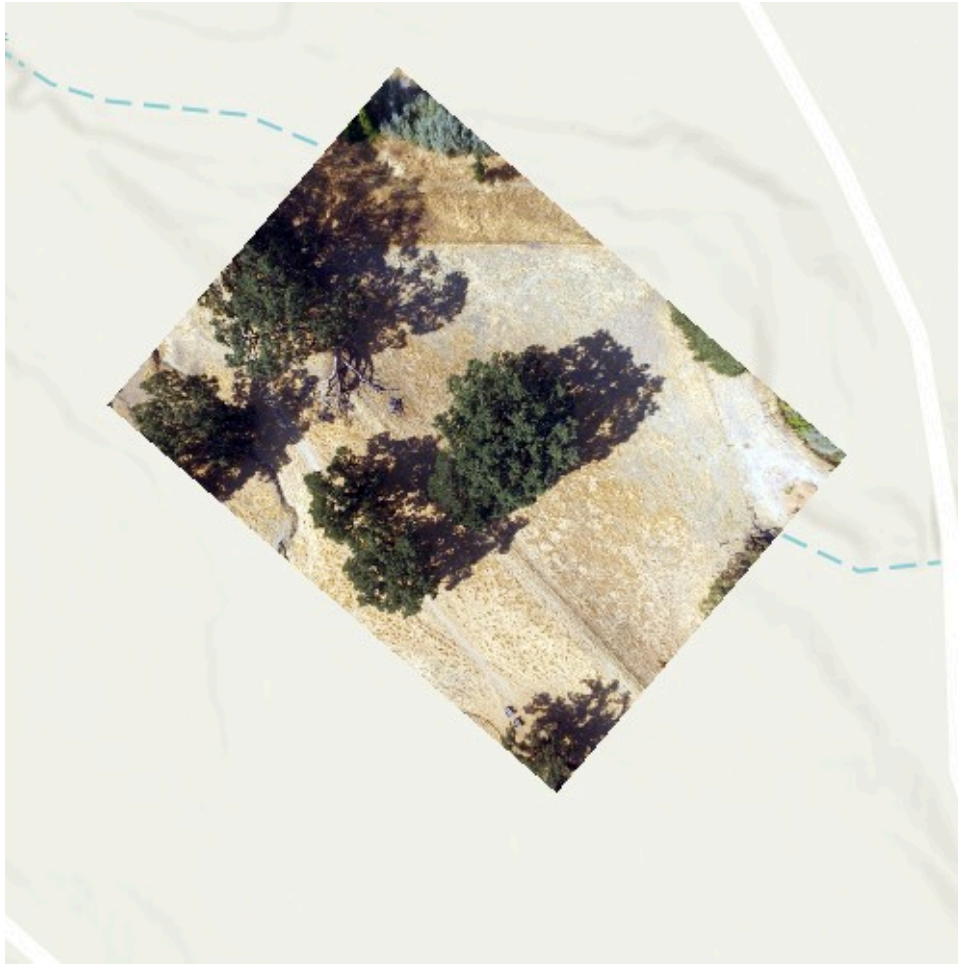
Options:

- World File format (aux.xml, wld, or prj)
- include yaw

Output:

Name	Date	Type	Size
 DJI_0001.JPG	12/10/2017 11:31 AM	JPG File	5,059 KB
 DJI_0001.JPG.aux.xml	5/13/2021 1:06 PM	XML Document	1 KB
 DJI_0002.JPG	12/10/2017 11:31 AM	JPG File	5,065 KB
 DJI_0002.JPG.aux.xml	5/13/2021 1:06 PM	XML Document	1 KB
 DJI_0003.JPG	12/10/2017 11:31 AM	JPG File	5,019 KB
 DJI_0003.JPG.aux.xml	5/13/2021 1:06 PM	XML Document	1 KB
 DJI_0004.JPG	12/10/2017 11:31 AM	JPG File	4,973 KB
 DJI_0004.JPG.aux.xml	5/13/2021 1:06 PM	XML Document	1 KB
 DJI_0005.JPG	12/10/2017 11:31 AM	JPG File	4,963 KB
 DJI_0005.JPG.aux.xml	5/13/2021 1:06 PM	XML Document	1 KB

With World files, your images will just “pop” into place (approximately) in your GIS software:



Crop Out Overlap

If your images can't be stitched, life is not over!

This function will crop each image, keeping the center part (least distorted)

```
uas_cropctr(ptpinole_info)
```

Function options:

- specify the crop dimensions in the image horizontal and vertical axis in meters

Example Output:



Not great, but...

- If the ground is flat and the features are short, might be good enough.

- If you're counting things in individual images, might be good enough.
- If you know you won't be able to stitch images (i.e., open water), consider flying at “zero percent” overlap and use this technique to verify you're not double-counting.

Convert between file formats

Convert between JPG, TIF, and DNG **without** losing all the image metadata (EXIF info):

```
uas_convert()
```

Save Images as Pseudo-Georeferenced GeoTIFFs

```
uas_exp_geotiff()
```

Open images with remote sensing tools for:

- spatial joins and overlays
- pixel classification
- compute vegetation indices
- zonal stats
- object detection with AI

In the pipeline...

Scaling for Production

- version 2.0 - more pipe friendly
- wrapper functions / code recipes
- automate production with an Excel based control panel
- standalone flight metadata editor

Making your data discoverable

- generate XML files for Google Dataset Search
- publish your data catalogs as AGOL web maps

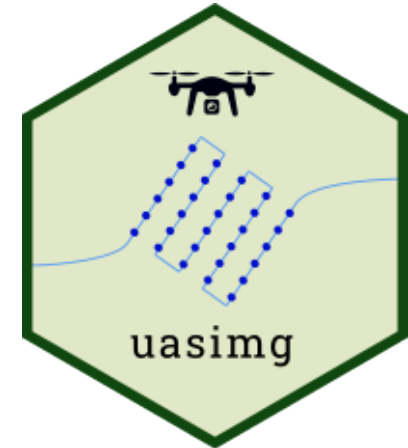
Adding Related Data to the Catalog

- flight logs, uncorrected images, calibration files, etc.
- processed data
- quality reports
- processing settings

Summary

The need for data management tools and workflows increases as you:

- accumulate more and more data
- work in a team
- are mandated to archive your data for the long term
- are mandated to make the existence of your data known to others



<https://ucanr-igis.github.io/uasimg/>

Data management and creating metadata will never be fun, but

We all stand to gain if we get better at this.

