GeoELAN v2.7.0 2024-10-15

Important:

> **GoPro Hero 12 Black is not supported** since it does not have a GPS module.
>
> **GoPro Hero 13 Black will be supported** as soon as I get hold of some raw/unedited sample footage.
>
> **Garmin VIRB Ultra 30 is discontinued**. GeoELAN will continue to support these and the FIT-format.

Introduction

GeoELAN is a command-line tool that geo-references time-aligned text-annotations of observed phenomena in audiovisual recordings, captured with a recent GoPro or Garmin VIRB action camera, see Larsson et al 2021. In other words, GeoELAN is used for annotating action camera GPS logs with the help of the free annotation tool ELAN.

Requirements:

- GoPro Hero 5 Black - GoPro Hero 11 Black, Hero 13 Black (Hero 12 Black is not supported)
- Garmin VIRB
- FFmpeg (in `PATH` preferred, but custom path can also be set when running GeoELAN)

> GeoELAN is multi-functional command-line tool that can
>
> - **geo-reference** ELAN-annotations of GoPro and VIRB footage (i.e. annotate GPS logs) and **generate annotated points, lines, or circles**.
> - **inspect** the raw content of your GoPro GPMF data, or Garmin FIT-files.
> - **locate and match** all relevant files belonging to the same recording session irrespective of file name (clips, telemetry-files).
> - automatically **join clips** (requires FFmpeg) for a specific recording session, and **generate an ELAN-file** with linked media.

Any ELAN annotation - be it an on-site utterance, or a plant in view - can be geo-referenced as long as the GPS logged coordinates within the annotation's timespan. The nature of the workflow also means consultants not physically present at the the time of recording can evaluate and annotate sections to be geo-referenced post-collection. As the name implies, the annotation tool ELAN plays a central role and is required to annotate events. The output can be points, polylines, or polygons (circles), as KML and GeoJSON-files. "GoPro" refers to a GoPro Hero 5 Black - Hero11 Black, and "VIRB" to the Garmin VIRB Ultra 30.

> While GeoELAN functionality differs slightly between Garmin and GoPro due to differences in formats and file structure, the general workflow and the final output are the same.

Acknowledgments

GeoELAN was developed with support from the Bank of Sweden Tercentenary Foundation (Grant nos NHS14-1665:1 and IN17-0183:1).

We would also like to acknowledge the The Language Archive, Max Planck Institute for Psycholinguistics in Nijmegen for their tireless efforts in developing ELAN, and making it available for free.

References

ELAN (Version 6.8) [Computer software]. 2024. Nijmegen: Max Planck Institute for Psycholinguistics, The Language Archive. Retrieved from https://archive.mpi.nl/tla/elan

Larsson, J. 2024. GeoELAN Manual. Lund: Lund University Humanities Lab. https://github.com/jenslar/geoelan

Larsson et al, 2021. Integrating behavioral and geospatial data on the timeline: towards new dimensions of analysis, *International Journal of Social Research Methodology*, 24:1, 1-13, DOI: 10.1080/13645579.2020.1763705

Requirements

- An action camera with a built-in GPS. Supported devices are:
  - GoPro Hero Black 5 - Hero 11 Black, Hero 13 Black. (Hero 12 Black is not supported. Max, and Fusion cameras have not been tested)
  - Garmin VIRB Ultra 30 (documentation) (**discontinued**)
- ELAN (documentation)
- FFmpeg (for concatenating video)
- Rust toolchain (optional, only required for compiling GeoELAN from source)

Installation

- Download the zip-file from https://github.com/jenslar/geoelan or use `git clone https://github.com/jenslar/geoelan.git`.
- See the `bin` directory for pre-compiled executables for Linux, macOS, and Windows.

Compile and install from source

You can also compile GeoELAN from source. Depending on your operating system, this may require installing additional software, and some understanding of working in a terminal. The basic steps are:

1. Install the Rust programming language
2. Get the GeoELAN source from https://github.com/jenslar/geoelan
3. `cd geoelan` (you should be in the folder containing `Cargo.toml`)
4. `cargo build --release`
5. `cargo install --path .` (optional, makes `geoelan` a global command)

Before you start

Practical advice

- **Keep the original files**. GeoELAN can only use unedited files, since these contain all telemetry (GPS log, sensor data etc), and identifiers used for synchronisation.
  - Pre-maturely **converting videos will discard both telemetry and identifiers**, which means there is no data for GeoELAN to work with.

    - Low-resolution clips have a `.LRV` extension for GoPro, and `.GLV` for VIRB (these are normal MP4-files).
    - GeoELAN will automaticall use low-resolution clips for linking in ELAN. Run `geoelan cam2eaf --link-high-res` to link high-resolution video instead.
- **You can rename files**. GeoELAN does not use file name when matching files as long as the file extension for video is either `.MP4`, `.LRV`, or `.GLV` (case ignored).
- **Data locations**:
    - **GoPro**: All telemetry is embedded inside the MP4-files.
    - **VIRB**: All telemetry, such as the GPS-log, is stored as a separate FIT-file.
    - **Keep all files on the microSD** unless you are absolutely certain which files are relevant.

Running GeoELAN

- GeoELAN is a command line tool and has no graphical user interface.
- FFmpeg is required to concatenate clips. (`cam2eaf`)
- If you use macOS and GeoELAN does not run, see https://support.apple.com/en-us/HT202491.

Device compatibility

- GoPro: Only "main line" Hero cameras with GPS have been tested, but Max and Fusion cameras may still work.
- Garmin: Only VIRB Ultra 30 has been tested extensively, but earlier VIRB models may still work.

GPS

Make sure the GPS is turned on and has acquired a satellite lock. This may take a couple of minutes or longer, especially if you have not used the camera for a while or have traveled far between uses.

Verifying a satellite lock:

- For **VIRB**, the GPS-icon should be steady, not blinking (it may log coordinates while the icon is still blinking, but do not rely on this being the norm).
- For **GoPro**, the GPS-icon should be white, not gray. The icon only shows under settings, not on the main screen.

> It may be difficult to acquire a satellite lock and/or reliably log position in areas with heavy overhead vegetation or dense cities with very tall buildings. Using a headstrap, instead of a cheststrap, sometimes helps.

GPS logging behaviour:

- **GoPro** logs dummy coordinates if no lock has been acquired. GeoELAN will not use these.
  - Verify lock by running: `geoelan inspect --gpmf PATH/TO/GOPRO.MP4 --gps` which will list number of bad points.
- **VIRB** seems not to log position at all until a satellite lock has been acquired.

Annotating in ELAN

- It is best to limit each kind of observed phenomena you wish to geo-reference to a single ELAN-tier, so…
- …to keep e.g. place names and plant sightings within the same ELAN-file, make a separate tier for each (see the example walkthrough in the next section). Then you can just re-run GeoELAN on the same ELAN-file and select another tier to geo-reference along with changing other output options as required.

Example walkthrough

This section describes how GeoELAN can be used to geo-reference ELAN-annotations. Please refer to the detailed sections if you get stuck. Remember that all input video clips must be the unprocessed, original MP4 (GoPro + VIRB) and FIT-files (VIRB). The so-called FIT-files mentioned throughout this manual are where the VIRB logs GPS-data and other kinds of telemetry during a recording session. These need to be matched to the corresponding video recording. GeoELAN will help with all of this, with the exception of annotating your data.

Note that some commands differ slightly between GoPro and VIRB.

The basic steps are:

1. Record video with a recent GoPro or VIRB.

2. Use GeoELAN to concatenate the video clips and generate an ELAN-file.
3. Annotate spatially interesting sections in ELAN.
4. Use GeoELAN to geo-reference the annotations, resulting in annotated KML and GeoJSON files.

Input files (example file names, naming convention may differ sligtly depending on model):

- **GoPro**:
  - `GH010026.MP4`, any clip in a recording session (remaining clips located automatically)
- **VIRB**:
  - `VIRB0001-1.MP4`, any clip in a recording session (remaining clips located automatically)
  - FIT-file with corresponding GPS-data (located automatically)

Output files:

- **GoPro + VIRB**:
  - KML and GeoJSON files with ELAN annotation content synchronised and mapped to the corresponding points as descriptions.

Step 1/3: Generate an ELAN-file with linked media files

In step 1 we will locate all video clips (GoPro + VIRB) and FIT-files (VIRB) that belong to a specific recording session. Video clips are then joined, and linked in the resulting ELAN-file.

Command

Command

```
geoelan cam2eaf --video INDIR/VIRB_OR_GOPRO_CLIP.MP4 --indir INDIR/ --outdir OUTDIR/
```

Output files GoPro

```
OUTDIR/GH010026/
├── GH010026.mp4            High-resolution video (concatenated)
├── GH010026_LO.mp4         Low-resolution video for ELAN (concatenated)
├── GH010026.wav            Extracted audio for ELAN (concatenated)
├── GH010026.eaf            ELAN-file with pre-linked media files
├── GH010026.kml            Overview KML-file with all points logged during the recording session
```

```
├── GH010026.json           Overview GeoJSON-file with all points logged during the recording
session
└── GH010026.txt            FFmpeg concatenation file, paths to input clips
```

Output files VIRB

```
OUTDIR/VIRB0001-1/
├── 2017-05-29-13-05-42.fit  FIT-file with corresponding telemetry
├── VIRB0001-1.mp4           High-resolution video (concatenated)
├── VIRB0001-1_LO.mp4        Low-resolution video for ELAN (concatenated)
├── VIRB0001-1.wav           Extracted audio for ELAN (concatenated)
├── VIRB0001-1.eaf           ELAN-file with pre-linked media files
├── VIRB0001-1.kml           Overview KML-file with all points logged during the recording session
├── VIRB0001-1.json          Overview GeoJSON-file with all points logged during the recording
session
└── VIRB0001-1.txt           FFmpeg concatenation file, paths to input clips
```

Explanation of the command

The relevant sub-command is `cam2eaf`. Run `geoelan cam2eaf --help` for an overview.

By specifying any clip in the recording session via `--video`, the remaining clips (GoPro + VIRB), including the corresponding FIT-file (VIRB), will be automatically located and joined, if they exist in the input directory `INDIR/`, including sub-directories. The result, including an ELAN-file with linked media files, will be saved to the output directory `OUTDIR/`.

If low-resolution clips (`.GLV`/`.LRV`) are located, these will be linked in the ELAN-file. If not, the high-resolution video will be linked instead.

GeoELAN defaults to *not* insert a tier with geo-data in the ELAN-file due to the effect this may have on performance. To do so, use the `--geotier` flag (see *Geo-data in ELAN*).

> **TIP:** For longer recording sessions or when batching, resulting in many video clips, step 1 is usually much faster if `--indir` and `--outdir` is not on the same physical hard drive. Those with an SSD (standard on most modern laptops) should be fine running step 1. on a single drive however.

Step 2/3: Annotate events in ELAN

Next, use ELAN with the ELAN-file from step 1 to annotate events that should be geo-referenced in step 3. Feel free to create any tier structure you may need. Tokenized tiers can not be geo-referenced, but otherwise any tier is fine, including deeply nested, referred tiers.

GeoELAN will geo-reference annotations from a single tier (selectable in step 3). Thus, if you want to generate a KML/GeoJSON-file with e.g. indigenous place names mentioned on-site during the recording, those place names must be limited to a single tier. If there are other spatial categories or groupings you wish to explore, simply create a new tier for each. In step 3 you can then re-run GeoELAN as many times as required, then select a different tier and/or options on each run.

When the annotations are geo-referenced in step 3, the annotation values in the selected tier will be used as descriptions for the synchronized, corresponding points in the KML and GeoJSON-files. Points corresponding to unannotated sections of the ELAN-file will either be discarded or have no description, depending on which options you use in step 3.

An annotated event can relate to anything observed in the recording and can be represented as either points or polylines in the output KML-file. If you are unsure which best applies to what you have in mind for your data, or how this may affect how you annotate, here are a few ideas for each kind.

> **Points** could concern documenting:
>
> - **the location of a plant or a geographical feature**, e.g. annotate the timespan either is visible in the video.
> - **an uttered place name or an animal cry**, e.g. annotate the timespan of the on-site utterance or cry.
>
> For these specific cases, the exact time spans of the annotations are not that important. It should be enough to ensure the annotation lasts for the duration of the place name being uttered, or for as long as the plant is visible. If unsure, add a another second to the annotation timespan. An average coordinate will be calculated for those that were logged within each annotation's time span, so as long as the camera wearer does not stray too far from the observation point, the result should be accurate enough.

Lines could concern documenting:

- various **types of movement through the landscape**. To annotate the movement of "walking up-hill" as it is observed visually in the recording, set the annotation's start time at the bottom of the hill and its end at the top, or for as long as the motion can be observed.
- a **narrative reflecting on the immediate surroundings** as they change over time. E.g. comments on visible landscape features, or perhaps the re-construction of an historical event as it unfolded over space and time.

Step 3/3: Generate a KML-file from geo-referenced ELAN annotations

Now that we have a few annotations, GeoELAN will geo-referenence these by determining which points were logged within each annotation's timespan. Note the different commands between GoPro and VIRB.

This is where you choose the approriate geographical representations for your annotated phenomena. Here are suggestions for the examples in step 2.

Points:

- the location of a plant or a geographical feature
- an uttered place name or an animal cry

To get a single, average coordinate for each annotation, use the `--geoshape point-single` option.

Lines:

- types of movement through the landscape
- narrative reflecting on the immediate surroundings

Two line options may apply to the above. To get a continuous polyline alternating between marked (annotated) and unmarked (un-annotated) events, use the option `--geoshape line-`

> `all`. To get a broken-up polyline representing marked events only, use the option `--geoshape line-multi`.

There are other options, such as *circle* output. It is the same as point output with the difference that radius and height can be specified (all circles will have the same size). For a more detailed overview of the possibilities, see the `--geoshape` option for the command *eaf2geo*. Experiment! If you realise one representation is not appropriate after all, re-run GeoELAN with a different option.

VIRB

Command

```
geoelan eaf2geo --eaf VIRB0001-1.eaf --fit 2003-01-02-12-00-00.fit --geoshape point-single
```

Output files

```
OUTDIR/VIRB0001-1/
├── ...                              Existing files
├── VIRB0001-1_point-single.kml      New KML-file, one point per annotation in the selected tier
└── VIRB0001-1_point-single.geojson  New GeoJSON-file, one point per annotation in the selected tier
```

GoPro

Command

```
geoelan eaf2geo --eaf GH010026.eaf --gpmf INDIR/GH010026.MP4  --geoshape point-single
```

> **Important:** `GH010026.MP4` **must be an unedited GoPro clip from the recording session**, as it was generated by the camera, **not** the video linked in your ELAN file. E.g. the same one specified in step 1.

Output files

```
OUTDIR/GH010026/
├── ...                           Existing files
├── GH010026_point-single.kml     New KML-file, one point per annotation in the selected tier
```

```
└── GH010026_point-single.geojson  New GeoJSON-file, one point per annotation in the selected tier
```

Explanation of the command

The relevant sub-command is `eaf2geo`. Run `geoelan eaf2geo --help` for an overview.

GeoELAN geo-references all annotations in a single tier (you will be prompted to select tier from a list) for the specified ELAN-file, then generates annotated KML and GeoJSON files where each point represents a single annotation.

By specifying an ELAN-file (`--eaf`) and an original, unedited GoPro MP4-clip (`--gpmf`) or VIRB FIT-file (`--fit`), GeoELAN will synchronise the annotations with the coordinates contained within the MP4/FIT-file. Similar to step 1, all files will be automatically located.

`--geoshape point-single` tells GeoELAN to distill each annotation into a single point (an average of all points withing the annotation timespan). The generated KML/GeoJSON-file will contain as many points as there are annotations in the selected tier. Each point inherits the corresponding annotation value as its description. The KML-file is named according to the selected `--geoshape` option, in this case `GH010026_point-single.kml`/`VIRB0001-1_point-single.kml`.

For the example command for VIRB, the user will be presented with a list of recording sessions present in the FIT-file (see *The FIT-format and the Garmin VIRB*). For GoPro, specifying an original clip, e.g. the same one specified in step 1, is enough.

Commands

Command	Alias	Description
`cam2eaf`	`g2e`	Generate an ELAN-file, and link concatenated media files
`eaf2geo`	`e2g`	Geo-reference ELAN-annotations and generate annotated KML/GeoJSON
`locate`	`l`	Locate and match video clips and/or FIT-files

Command	Alias	Description
`inspect`	`i`	Inspect the telemetry of a GoPro MP4-file or any Garmin FIT-file
`plot`	`p`	Plot the telemetry of a GoPro MP4-file or any Garmin FIT-file
`manual`	`m`	View or save this manual to disk

Run `geoelan --help` for a general overview, or `geoelan <COMMAND> --help`, for an overview of a specific command.

The most relevant commands are probably `cam2eaf` and `eaf2geo`. `locate` is there to help with locating and matching video clips and/or FIT-files that belong to the same recording session, but this functionality partly exists in `cam2eaf` as well. `inspect` can be used to print various kinds of data in a GoPro MP4/Garmin FIT-file, but will do so in an unprocessed form. It is intended more as a technical aid for troubleshooting or to verify the contents of MP4/FIT-files. `plot` is used to plot sensor data and some of the GPS data, such as altitude over time. `manual` is for viewing or saving the full manual.

> Note that some parameters in the following sections may only be valid for e.g. GoPro cameras, not VIRB, and vice versa. The description column will be prefixed [GoPro] or [VIRB] to denote this.

Set GoPro satellite lock (`--gpsfix`) and dilution of position (`--gpsdop`) thresholds

GoPro cameras log how well they can see satellites. If none is in line of sight, dummy coordinates will be logged. GeoELAN will ignore these by default, and for `cam2eaf` a '3D lock' (altitude is included) is the default. In cases where only 2D lock could be achieved, one can manually set minimum "lock level" via `--gpsfix`. Valid values are `0` (no lock), `2` (2D lock), and `3` (3D lock). Setting to `0` will result in unusable data for `eaf2geo` if most coordinates are bad.

Time adjustment with `--time-offset`

If the action camera has not adjusted for the current time zone, several commands have a `--time-offset` option. It takes a /- value in hours that will be applied to all timestamps in the output, e.g. `--time-offset 7+` will add seven hours to all timestamps.

Reducing the number of coordinates with `--downsample`

The command `eaf2geo` outputs coordinates as KML and GeoJSON files. Since supported cameras log at either 10 or 18Hz, a 2 hour recording may contain more than 70 000 logged points. The `--downsample` parameter can be used to reduce the number of coordinates exported. Google Earth does not cope well with a large amount of points, whereas dedicated GIS software such as QGIS, usually will.

`--downsample` takes a positive numerical value that is effectively a divisor: `--downsample 10` means an average coordinate will be calculated for every cluster of 10 points. For 70 000 logged points, a value of 100 means the output will contain 700 averaged points and so on. If the user sets `--downsample` to a value that exceeds the total number of points logged by the GPS, it will be changed to the largest applicable value (resulting in a single point for the entire recording as opposed to none at all).

> Extreme values may affect the result in unexpected ways, depending on gaps in and/or quality of the GPS-data.

VIRB Ultra 30 logs at 10Hz, and GoPro logs at 10 or 18Hz depending on model. Only VIRB Ultra 30 and GoPro Hero 11 (10Hz) and later timestamp each individual point, whereas earlier models only timestamp a cluster of points. In the latter case, GeoELAN average each cluster to a single, timestamped point, resulting in roughly 1 point/second.

If 'cam2eaf' or 'eaf2geo' return errors

Try the `inspect` command on problematic MP4/FIT-files. This way you can verify whether points were actually logged or not. If the file is corrupt the error message will also be printed.

FFmpeg

The command `cam2eaf` requires FFmpeg. See the appendix under *FFmpeg* on how to install. If you intend to use the *static build*, point to it using `--ffmpeg PATH/TO/FFMPEG/ffmpeg` (`ffmpeg.exe` on Windows). If the `--ffmpeg` option is not used, `geoelan` will assume `ffmpeg` is available as a global command and complain accordingly if it is not.

> **TIP:** GeoELAN will never overwrite existing files without permission. Should you accidentally delete the generated ELAN-file with the output media files intact, just re-run the `cam2eaf`

command. It will automatically skip concatenating videos, but still generate a new ELAN-file.

TIP: In the tables for the respective command sections, arguments listed under 'Flags' do not take a value, whereas those listed under 'Options' do. If a `default` value is listed, it will be automatically set, unless the user specifies otherwise.

cam2eaf

- *Command/alias:* `cam2eaf` / `c2e`
- *Help:* `geoelan cam2eaf --help`
- *Basic usage:* `geoelan cam2eaf --indir INDIR/ --video GH010006.MP4 --outdir OUTDIR/`

`cam2eaf` generates an ELAN-file with pre-linked media files. All clips in the specified recording session will be automatically located, grouped, and concatenated. A WAV-file from the full video is also extracted. By default the low-resolution footage is used (if found), use the `--link-high-res` flag to link the high-resolution footage. The corresponding coordinates can optionally be added a tier.

Flags

Short	Long	Description
	`--dryrun`	Show results but do not process or copy files
	`--fullgps`	Use the full-res GPS log for the ELAN geotier
	`--geotier`	Insert tier with synchronised coordinates in ELAN-file
	`--link-high-res`	Link high-resolution video in ELAN-file
`-l`	`--low-res-only`	Only concatenate low-res clips (`.LRV`/`.GLV`), ignores high-res clips
	`--single`	Only use the specified clip, ignore remaining clips in session

Short	Long	Description
	`--verify`	[GoPro] Verify GPMF data, ignore corrupt clips

Options

Short	Long	Description	Default	Required
	`--ffmpeg`	Custom path to FFmpeg	`ffmpeg`	
`-i`	`--indir`	Input path for locating files		yes
`-o`	`--outdir`	Output path for resulting files	`geoelan`	
`-t`	`--time-offset`	Time offset in +/- hours	`0`	
`-v`	`--video`	Clip in the relevant session		unless `-f` or `-u`
	`--gpsfix`	[GoPro] Minimum satellite lock	`3`	
`-f`	`--fit`	[VIRB] FIT-file		unless `-u` or `-v`
`-u`	`--uuid`	[VIRB] UUID for a clip in the relevant session		unless `-f` or `-v`

Example GoPro

GoPro example

`geoelan`	`cam2eaf`	`-v GH010026.MP4`	`-i INDIR/`	`-o OUTDIR/`	`--geotier`
	command	clip in session	input directory	output directory	insert coordinate tier

Result: Locates all clips for the recording session containing the clip `GH010026.MP4` (`-g`) in the input

directory `INDIR/` (`-i`). These will be concatenated, and the audio track exported as a WAV for use in ELAN. The resulting files are then copied to the output directory `OUTDIR/` (`-o`). The generated ELAN-file will also have synchronised coordinates inserted as a tier (`--geotier`).

Examples VIRB

> Recording session can be specified using one of `--fit`, `--uuid`, `--video`. These options are mutually exclusive. `--fit` returns a list of sessions present in the FIT-file, from which the user can select the relevant one. `--uuid` and `--video` require no further user input. UUID is the unique VIRB clip identifier and can be retreived by running `geoelan inspect --video VIRB0001-1.MP4`.

> Using `--fullgps` (together with `--geotier`) may slow down ELAN considerably.

VIRB example 1

`geoelan`	`cam2eaf`	`-v VIRB0001-1.MP4`	`-i INDIR/`	`-o OUTDIR/`	`--geotier`
	command	clip in session	input directory	output directory	insert coordinate tier

Result: Locates all clips for the recording session containing the clip `VIRB0001-1.MP4` (`-v`) in the input directory `INDIR/` (`-i`). These will be concatenated, and the audio track exported as a WAV for use in ELAN. The resulting files are then copied together with the corresponding FIT-file to the output directory `OUTDIR/` (`-o`). The generated ELAN-file will also have synchronised coordinates inserted as a tier (`--geotier`).

VIRB example 2

`geoelan`	`cam2eaf`	`-f 2017-01-28-05-16-40.FIT`	`-i INDIR/`	`-o OUTDIR/`	`-l`
	command	FIT-file	input directory	output directory	ignore hi-res MP4

Result: Recording session is specified via the FIT-file `2017-01-28-05-16-40.fit` (`-f`). The user will be

prompted to select session from a list, allowing GeoELAN to locate the corresponding clips in the input directory `INDIR/` (`-i`). Only the low-resolution clips (`--low-res-only`) will be concatenated. All resulting files are then copied together with the corresponding FIT-file to the output directory `OUTDIR/` (`-o`).

> If you are unsure of the whereabouts of the FIT-file, make the search wider. Specifying the root of an external hard drive as input directory (`--indir`) will make the search process take slightly longer, but should work well. Otherwise, just specify the FIT-file separately (`--fit`), which can be useful if it is located outside of the input directory.

eaf2geo

> - *Command/alias:* `eaf2geo` / `e2g`
> - *Help:* `geoelan eaf2geo --help`
> - *Basic usage:* `geoelan eaf2geo --eaf VIRB0001-1.eaf --fit 2017-01-28-05-16-40.fit`

`eaf2geo` generates KML and GeoJSON files by geo-referencing all annotations in the specified tier. The user is presented with a list of all tiers in the ELAN-file to select from. Referred tiers are fine, but tokenized tiers can not be used, since these lack meaningful time stamps. Several output options exist via the `--geoshape` option, such as points or polylines (see below). In the resulting KML and GeoJSON files, any point that intersects with an annotation's timespan will inherit the annotation value as a description.

Flags

Short	Long	Description
	`--cdata`	KML-option, added visuals in Google Earth

Options

Short	Long	Description	Default	Possible	Required
`-d`	`--downsample`	Downsample factor for coordinates	`1`		

Short	Long	Description	Default	Possible	Required
`-e`	`--eaf`	ELAN-file			yes
`-f`	`--fit`	[VIRB] FIT-file			unless `-g`
`-g`	`--gpmf`	[GoPro] MP4-file			unless `-f`
	`--geoshape`	Output options for KML-file	`point-all`	`point-all`, `point-multi`, `point-single`, `line-all`, `line-multi`, `circle-2d`, `circle-3d`	
	`--height`	Circle height (`circle-3d`)	`10.0`		
	`--radius`	Circle radius (`circle-2d`, `circle-3d`)	`2.0`		
`-t`	`--time-offset`	Time offset, +/- hours	`0`		
	`--vertices`	Circle vertices/roundness ('circle-2d', 'circle-3d')	`40`		

GoPro example

`geoelan`	`eaf2geo`	`-g GH010026.MP4`	`-e GH010026.eaf`	`--geoshape line-all`
	command	original GoPro MP4-file	ELAN-file	output option

Result: Geo-references annotations in the ELAN-file `GH010026.eaf` (`-e`) and generates KML and GeoJSON files with a continous poly-line, alternating between marked (annotated) and unmarked (un-annotated) sections (`--geoshape line-all`).

VIRB example

`geoelan`	`eaf2geo`	`-f 2017-01-28-05-16-40.fit`	`-e VIRB0001-1.eaf`	`--geoshape point-single`
	command	FIT-file	ELAN-file	output option

Result: Geo-references annotations in the ELAN-file `VIRB0001-1.eaf` (`-e`) and generates KML and GeoJSON files with a single point per annotation (`--geoshape point-single`). Since no original VIRB clip is specified, the user will be presented with a list of clip UUIDs in the specified FIT-file `2017-01-28-05-16-40.fit` (`-f`) to choose from. It should be fairly straight forward to guess which session is relevant.

The *geoshape* option

Different geographical representations can be generated, including points and lines. Six possible `--geoshape` values are accepted:

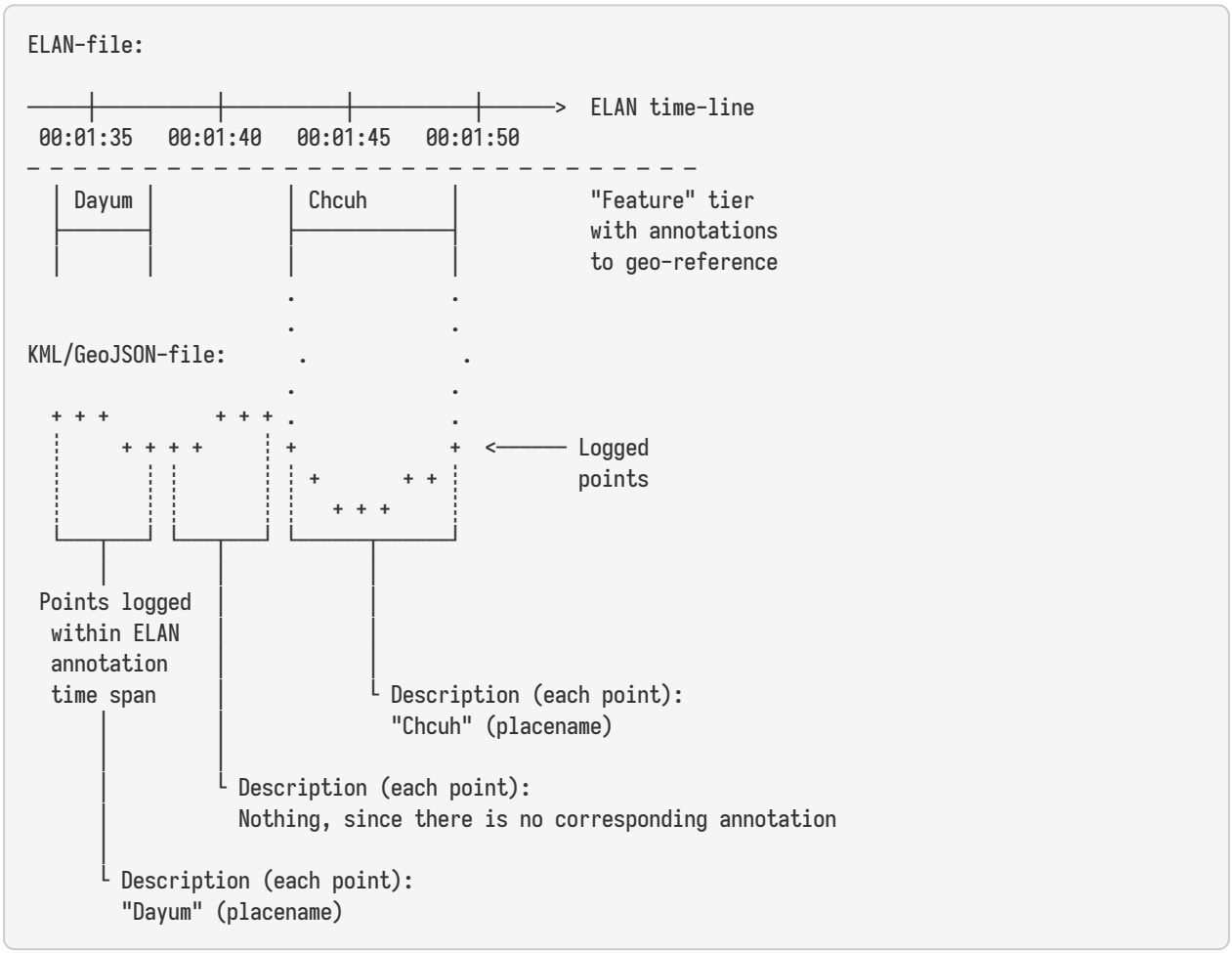
Option	Description
`point-all`	All logged points exported (default if no option passed)
`point-multi`	Exported points correspond to marked/annotated events only
`point-single`	A single, averaged point for each annotation
`line-all`	Polyline from all logged points
`line-multi`	Polyline, corresponds to marked/annotated events only
`circle-2d`	2D polygon, corresponds to marked/annotated events only
`circle-3d`	3D polygon, corresponds to marked/annotated events only

`--downsample` can be used with all these options, but will be ignored for `point-single`, `circle-2d` /`circle-3d` since these will only ever result in a single point per annotation. `circle-2d` and `circle-3d` allow for further customisation, such as radius and height (`circle-3d`, KML-only). The circle options

work in the same way as `point-single` and are currently only a visual flair, since radius and height are not yet derived from ELAN annotation values.

`point-all`

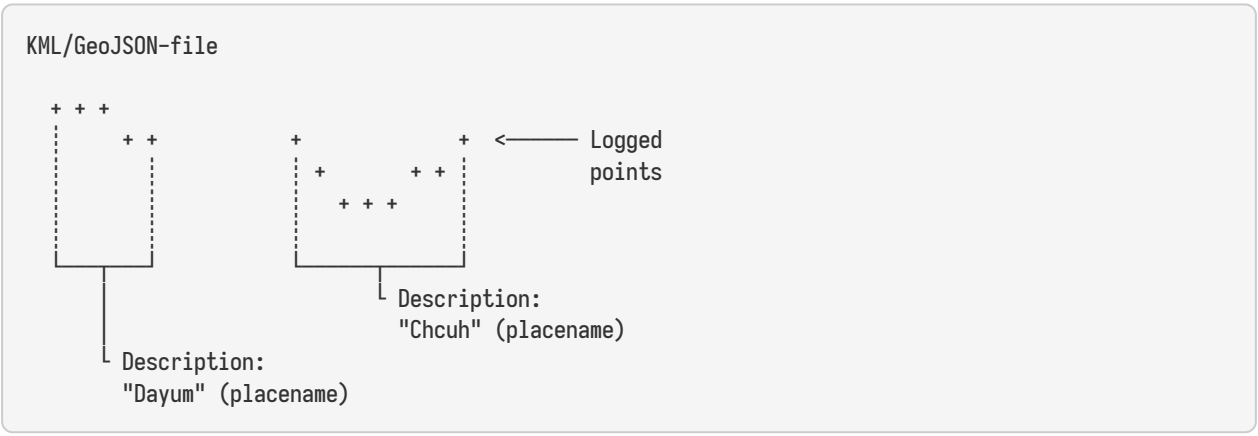
All points logged during the recording session will be exported. Only points that intersect with the time span of an annotation will inherit the annotation value as the coordinate description. Points that do not, will have no description.



`point-multi`

Only points that intersect with the time span of an annotation will be exported. Each point will inherit the annotation text as the coordinate description. Points that have no corresponding annotation will be discarded.

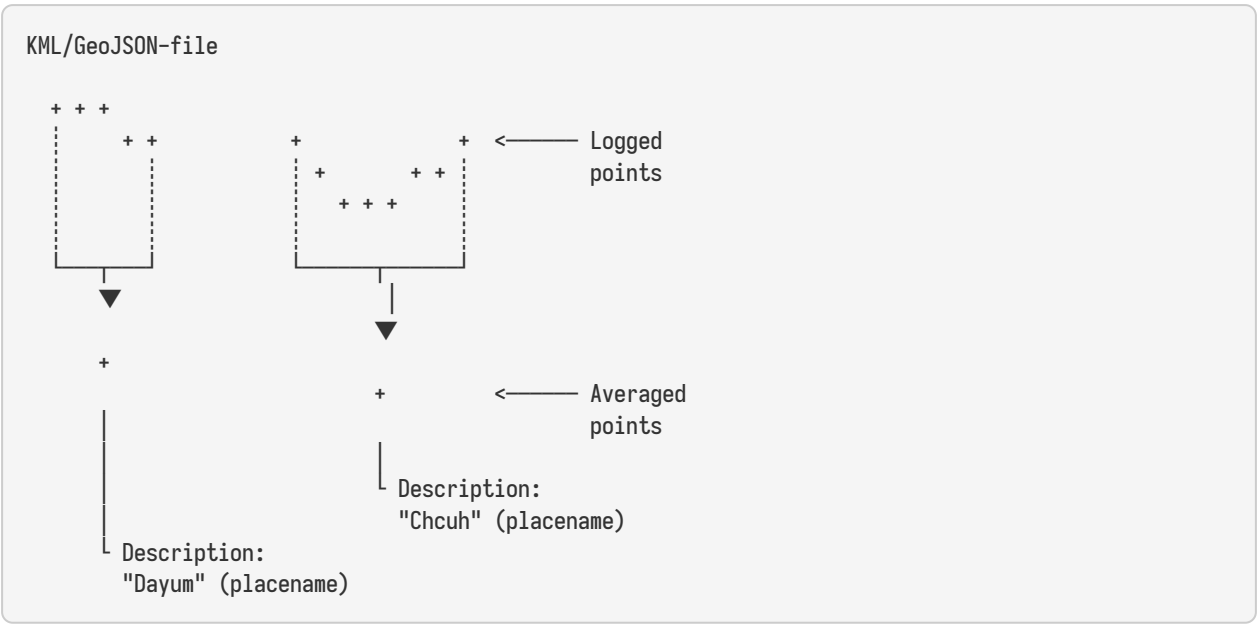
> Useful for including points corresponding to marked events only.



`point-single`

Only points that intersect with the time span of an annotation will be considered for export. The difference to `point-multi` is that each annotation will only generate a single point: an average of those logged within the annotation's time span. Note that a custom `--downsample` value will be ignored for `point-single` since it may affect the result negatively. `--downsample` also has little use here, since the number of points in the output will not be affected and will be quite low compared to the other options.

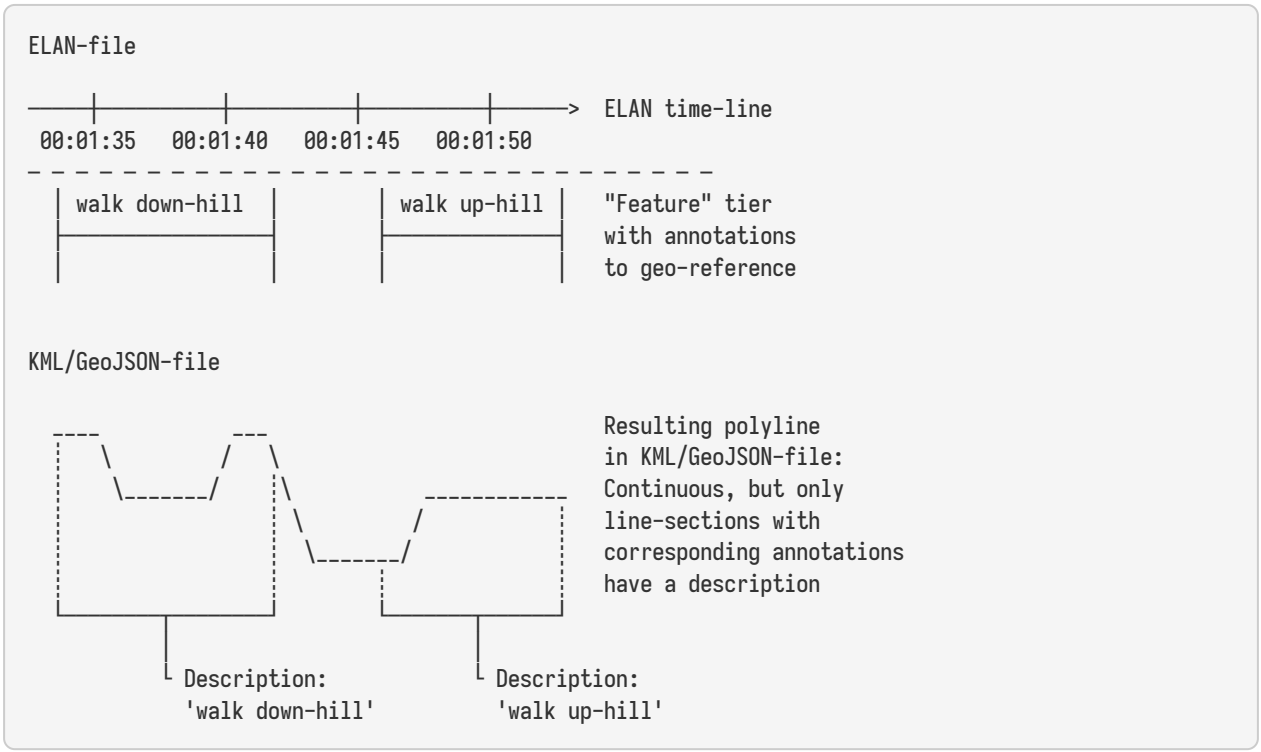
> Useful for distilling marked events, such as place names, to a single point for each event.



`line-all`

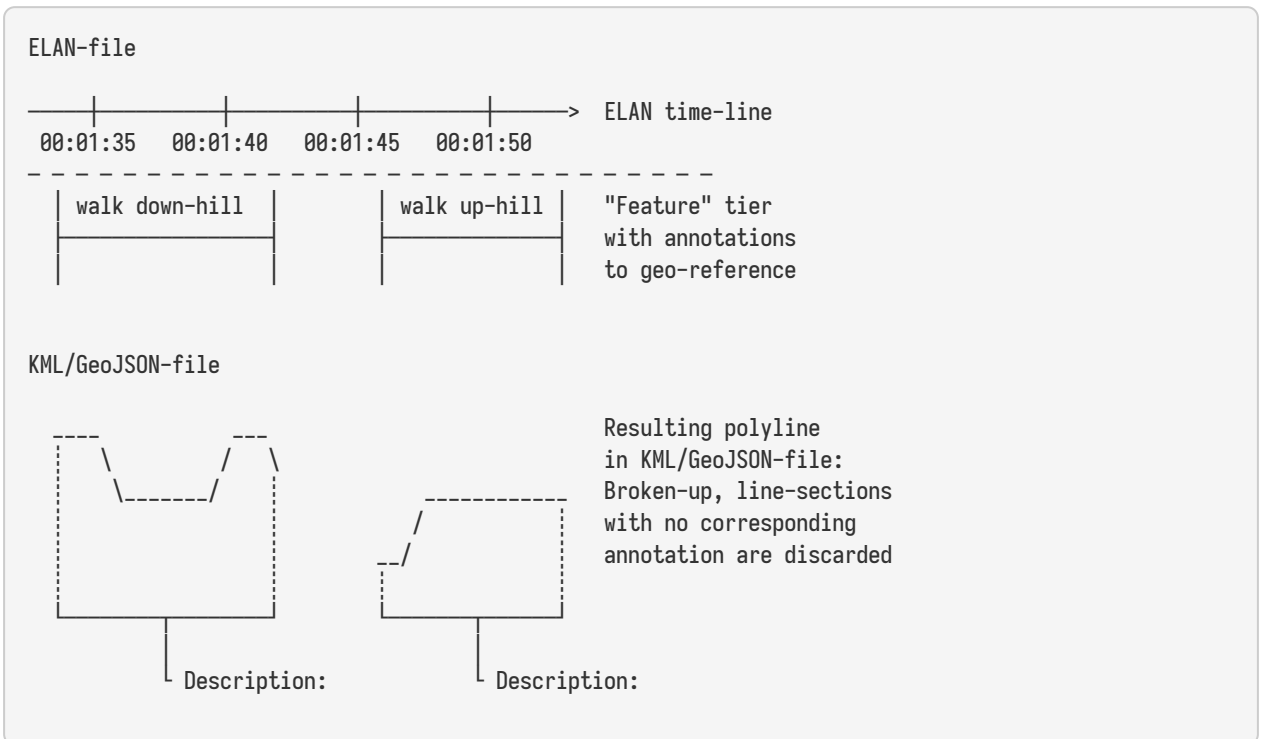
Similar to `point-all`. All points logged during the recording session will be exported, resulting in a continuous polyline. Only line-sections that intersect with an annotation span inherit the annotation

value as a description. Those that do not will have no description.



line-multi

Only points that intersect with the time span of an annotation will be exported, resulting in a broken-up line. Each sub-section inherits the value of the annotation it intersects with. *Useful for representing paths corresponding to marked events only*.



```
            'walk down-hill'          'walk up-hill'
```

`circle-2d`, `circle-3d`

`circle-2d`, and `circle-3d` work almost exactly like `point-single` with the difference that a circle is generated around the calculated average point. It is mostly a visual flair and its shape is currently not affected by annotation values. `circle-2d` is flat against the ground, whereas `circle-3d` can take a height value to become a cylindrical 3D shape (only applies to KML, not GeoJSON). If circle output is specified, three more options become available:

Option	Description	Default
`--height`	Height relative to ground in meters (`circle-3d`)	
`--radius`	Radius in meters (`circle-2d`, `circle-3d`)	`2.0`
`--vertices`	Roundness, valid range 3 - 255 (3 will literally be triangle)	`40`

The 'cdata' option

The `--cdata` option only affects KML-files. It will insert extra information into the KML-file in the form of HTML inside the `<description>` element for each point (see the CDATA section in Google's KML documentation). In Google Earth this results in an information bubble to pop up when a point is clicked on, as a visual flair for e.g. presentations.

locate

> - *Command/alias:* `locate` / `l`
> - *Help:* `geoelan locate --help`
> - *Basic usage:* `geoelan locate --indir INDIR/ --kind gopro`

`locate` will locate and match original GoPro and VIRB clips in the input folder. For VIRB, corresponding FIT-file/s will also be located. By optionally specifying a UUID (`--uuid`, `--fit`) or a clip (`--video`) in a specific session, only the files in that recording session will be returned. If you are unsure of the

location of all relevant files, use an input path closer to the root, such as the root of an external hard drive. If duplicate files are found, the last one encountered will be returned.

Flags

Short	Long	Description
	`--quiet`	Do not print file-by-file search progress

Options

Short	Long	Description	Possible	Required
`-i`	`--indir`	Input path for locating files		yes
`-k`	`--kind`	Camera brand	`virb`, `gopro`	unless `-v`, `-u`, `-f`
`-v`	`--video`	Clip in relevant session		
	`--verify`	[GoPro] Verify GPMF data, ignore corrupt files		
`-f`	`--fit`	[VIRB] FIT-file for selecting session		
`-u`	`--uuid`	[VIRB] UUID for clip in session		

Example 1

`geoelan`	`locate`	`-i INDIR/`	`--kind gopro`
	sub-command	input directory	consider GoPro files

Result: Locates all GoPro clips in `INDIR/` (`-i`) and groups them in recording sessions.

Example 2

`geoelan`	`locate`	`-i INDIR/`	`-v VIRB0001-1.MP4`
	sub-command	input directory	clip in relevant session

Result: Camera brand is detected automatically (in this case VIRB). Locates all clips in `INDIR/` (`-i`) for the recording session that contains `VIRB0001-1.MP4` (`-v`) together with the corresponding FIT-file.

inspect

- *Command/alias:* `inspect` / `i`
- *Help:* `geoelan inspect --help`
- *Basic usage:*
  - GoPro: `geoelan inspect --gpmf GH010026.MP4`
  - VIRB: `geoelan inspect --fit 2017-01-28-05-16-40.fit`

`inspect` prints an overview or the detailed contents of a GoPro MP4 or a Garmin FIT-file. If a video clip is passed with no other options, the embedded identifiers will be printed. Options include filtering to print only a sub-set of the data, such as GPS-data only, data corresponding to a specific recording session, or both. `locate` is more of a technical aid to, for example, verify that the GPS really did log coordinates. KML or GeoJSON files can also be generated.

Flags

Short	Long	Description
	`--debug`	Print FIT definitions and data while parsing
	`--kml`	Generate a KML-file
	`--ikml`	Generate an indexed KML-file
	`--json`	Generate a GeoJSON-file.
	`--verbose`	Print raw data
	`--gps`	Print processed GPS data
	`--meta`	Print MP4 custom user data (`udta` atom)
	`--atoms`	Print MP4 atom hierarchy

Short	Long	Description
`-o`	`--offsets`	[GoPro] Print `DEVC` byte offsets for GoPro MP4-file
`-s`	`--session`	[VIRB] Select session from a list. GoPro: Merges session data.

Note that the `--offsets` flag can only be used if the GoPro MP4-file was specified via the `--gpmf` option below. It has no use for other MP4-files. `DEVC` is a container of sorts that holds GPMF data, and is interleaved with the image and audio samples inside the MP4 `mdat` atom.

Options

Short	Long	Description	Required
`-f`	`--fit`	[VIRB]FIT-file	unless `-g`, `-v`
`-g`	`--gpmf`	[GoPro]-file (MP4 or raw GPMF-file)	unless `-f`, `-v`
`-t`	`--type`	Data type to filter on	
`-v`	`--video`	MP4-file	

Note that `--type` takes a string for GoPro and a numerical identifier for VIRB. `--video` accepts any MP4-file. See the sections below.

Inspecting telemetry

`inspect` will mostly print raw values - down to a list of bytes for some kinds of data - that require further processing to be of use. The exact nature of this data differs between GoPro and Garmin. For GPS data, the flag `--gps` can be used for either device to print a processed GPS-log showing coordinates in decimal degrees etc. The other GeoELAN commands, such as `eaf2geo`, always convert data to the relevant forms.

If a GoPro MP4 or a Garmin FIT-file can not be properly parsed, GeoELAN will often return an error message that may hint at the issue. Try `inspect` on files that raise errors with the other commands.

GoPro

GoPro cameras embed all logged telemetry inside the MP4-files. In contrast to Garmin FIT, data types have no numerical identifier (see below) so internal text descriptions are used instead.

To list all data types logged in a GoPro MP4-file, run:

```
geoelan inspect --gpmf GOPROVIDEO.MP4
```

GPS is identified as `GPS (Lat., Long., Alt., 2D speed, 3D speed)`, for example. However as GoPro release new models some descriptions change. For Hero 11 (and later) GPS can also be identified as `GPS (Lat., Long., Alt., 2D, 3D, days, secs, DOP, fix)`, and some data types may be deprecated or eventually removed so list data first to see what's available.

This will list all data streams (every `DEVC` stream holds one logged instance of each listed item):

```
Unique data stream types (1018 DEVC streams in total):
    Accelerometer
    Average luminance
    Exposure time (shutter speed)
    Face Coordinates and details
    GPS (Lat., Long., Alt., 2D speed, 3D speed)
    Gyroscope
    Image uniformity
    Predominant hue[[hue, weight], ...]
    Scene classification[[CLASSIFIER_FOUR_CC,prob], ...]
    Sensor ISO
    Sensor read out time
    White Balance RGB gains
    White Balance temperature (Kelvin)
```

Find the data type you wish to inspect further in the list. To print GPS data in its "raw" form, run:

```
geoelan inspect --gpmf GOPROVIDEO.MP4 --type 'GPS (Lat., Long., Alt., 2D speed, 3D speed)'
```

Note the citation marks (single or double). These are necessary to pass the description as a single string to GeoELAN. Your terminal shell may otherwise try to interpret anything delimited by space as a command or a parameter, which will fail. Unlike FIT, GPMF has no numerical identifier for data types.

GPS specifically, can be printed in a more conventional form via the flag `--gps`:

```
geoelan inspect --gpmf GOPROVIDEO.MP4 --gps
```

Generate a KML-file with:

```
geoelan inspect --gpmf GOPROVIDEO.MP4 --kml
```

DEVC byte offsets

To list the byte offsets for GPMF DEVC containers (these hold raw GPMF data), run:

```
geoelan inspect --gpmf GOPROVIDEO.MP4 --offsets
```

This returns a table listing byte offset in the MP4-file (`@XXXXX`), container size, and duration covered in milliseconds e.g.:

```
DEVC @332392   size: 4836   duration: 1001ms
DEVC @681939   size: 4840   duration: 1001ms
DEVC @1037851  size: 4904   duration: 1001ms
DEVC @1326489  size: 4924   duration: 1001ms
```

Images

Original GoPro JPEG-images can also be inspected. These will contain much less GPMF data than the MP4-files, and are currently not used elsewhere in GeoELAN's workflow. If no named data shows up in the summary, try `geoelan inspect --gpmf GOPROIMAGE.JPG --verbose` to print the raw data. Early GoPro models do not embed GPMF data in JPEG-images.

Garmin FIT

The FIT-format is quite different to GoPro's GPMF, apart from being a separate file. There is among other things, additional information about VIRB recording sessions. The VIRB starts logging to a FIT-file the moment the camera is turned on, and only stops when it is turned off. This means that a single FIT-file may contain data for multiple recording sessions. Data is logged continuously - even between recordings.

Inside a FIT-file, data is identified by a numerical identifier. For example, GPS data is `160`, also referred

to as `gps_metadata` in the FIT Software Development Kit (FIT SDK). `inspect` lists both identifiers in the summary table, but only the numerical identifier is logged inside the FIT-file.

To list all data types logged in a VIRB FIT-file, run:

```
geoelan inspect --fit FITFILE.FIT
```

This will return a table:

```
 Global ID | Message type                | Count
.............................................
         0 | file_id                     |     1
        18 | session                     |     1
        19 | lap                         |     1
        20 | record                      |  6209
        21 | event                       |     1
        22 | UNKNOWN_TYPE_22             |     2
        23 | device_info                 |     3
        34 | activity                    |     1
        49 | file_creator                |     1
       104 | UNKNOWN_TYPE_104            |   104
       160 | gps_metadata                | 60114
       161 | camera_event                |    24
       162 | timestamp_correlation       |     1
       164 | gyroscope_data              | 20405
       165 | accelerometer_data          | 20405
       167 | three_d_sensor_calibration  |    59
       208 | magnetometer_data           | 20405
       209 | barometer_data              |  6209
       210 | one_d_sensor_calibration    |     1
       219 | UNKNOWN_TYPE_219            |     1
.............................................
                                 Total:  133948
```

Find the data type you wish to inspect further in the list and take note note of the "Global ID". To print GPS data in its "raw" form, run:

```
geoelan inspect --fit FITFILE.FIT --type 160
```

GPS specifically, can be printed in a more conventional form via the flag `--gps`:

```
geoelan inspect --fit FITFILE.FIT --gps
```

Generate a KML-file with:

```
geoelan inspect --fit FITFILE.FIT --kml
```

To print a single type of data belonging to a specific recording session (VIRB only) use `--session`:

```
geoelan inspect --fit FITFILE.FIT --type 160 --session
```

This will return a table listing all VIRB recording sessions together with an input prompt (UUIDs shortened to fit):

```
 Session | Clips | First UUID in session
.....................................................................................
  1.     |  1    | VIRBactioncameraULTRA30_Tall_2688_2016_29..._1_17_2017-01-28-05-16-40.fit
  2.     |  1    | VIRBactioncameraULTRA30_Tall_2688_2016_29..._1_18_2017-01-28-05-16-40.fit
  3.     |  3    | VIRBactioncameraULTRA30_Tall_2688_2016_29..._1_19_2017-01-28-05-16-40.fit
         |       | VIRBactioncameraULTRA30_Tall_2688_2016_29..._2_19_2017-01-28-05-16-40.fit
         |       | VIRBactioncameraULTRA30_Tall_2688_2016_29..._3_19_2017-01-28-05-16-40.fit
  4.     |  1    | VIRBactioncameraULTRA30_Tall_2688_2016_29..._1_20_2017-01-28-05-16-40.fit
  5.     |  1    | VIRBactioncameraULTRA30_Tall_2688_2016_29..._1_21_2017-01-28-05-16-40.fit
.....................................................................................
Select session:
```

Type the number in the "Session" column for the relevant session and press enter. The output will now be limited to the selected recording session, which also applies when generating KML/GeoJSON-files.

You could also specify an original VIRB video via `--video`, or a UUID directly (any UUID in the session) via `--uuid` to achieve the same result.

To find out the embedded UUID of a VIRB MP4-file, run:

```
geoelan inspect --video VIRBVIDEO.MP4
```

This will return the embedded UUID:

```
UUID: VIRBactioncameraULTRA30_Expansive_1920_1440_29.9700_3937280306_3af2a648_1_299_2021-05-03-14-23-23.fit
```

Most FIT-files, from e.g. watches, bike computers, will work with `inspect`. Custom developer data is also supported (such fields will be prefixed 'DEV' when inspecting). However, some FIT features are exclusive to VIRB, such as UUID and selecting sessions, and other features are not implemented, such as compressed timestamp headers. In such cases, the tool will report the error and exit. Missing features may or may not be implemented in future versions.

> For those who wish to dig deeper, the Garmin FIT Software Development Kit contains a spreadsheet, `Profile.xlsx`, which lists the kinds of data a FIT-file may contain. Not all of those apply to every device however, and undocumented data types exist.

Video/MP4-files

MP4-files have a few options, besides inspecting embedded GoPro GPMF data. Access these by using the `--video` option.

The `--meta` flag will show raw (i.e. bytes) content for the so-called `udta` atom for any MP4-file. GoPro also embeds undocumented GPMF data in the `udta` atom which will also be listed, whereas Garmin logs a unique identifier here. The `--atoms` flag will show the MP4 atom hierarchy. To most users these will not be of much use, but may provide technical context for troubleshooting. If `--atoms` is not enough, try the command line tool AtomicParsley.

For example, for a GoPro video run:

```
geoelan inspect --video GOPROVIDEO.MP4
```

This returns:

```
Identified as Hero11 Black MP4 file
  MUID: [928039425, 4049218210, 3428159554, 991755340, 1293346567, 3926065152, 6753, 0]
  GUMI: [99, 186, 221, 126, 76, 138, 61, 112, 215, 119, 168, 236, 184, 24, 94, 158]
Creation time: 2023-01-25 12:15:42.0
Duration:      896.896s
To inspect GPMF run 'geoelan inspect --gpmf GOPROVIDEO.MP4'
```

To print the internal MP4 hierarchy run (try Atomic Parsley for much better support):

```
geoelan inspect --video GOPROVIDEO.MP4 --atoms
```

The udta atom can contain various kinds of metadata. Its contents heavily depends on what generated the MP4 file. GoPro logs additional camera metadata in GPMF form in the udta atom, including MUID, and GUMI, whereas VIRB logs the clip UUID in here. This will be printed as well if present. To print, run:

```
geoelan inspect --video GOPROVIDEO.MP4 --meta
```

plot

- *Command/alias:* plot / p
- *Help:* geoelan plot --help
- *Basic usage:*
  - GoPro: geoelan plot --gpmf GH010026.MP4 --y-axis accelerometer --x-axis time
  - VIRB: geoelan plot --fit 2017-01-28-05-16-40.fit --y-axis accelerometer --x-axis time

plot can plot some of the telemetry in a semi-interactive web view, such as sensor data (accelerometer, gyroscope over time or sample count), and GPS data (latitude, longitude, altitude over time or distance).

> plot uses plotly.js via the plotly Rust crate.

manual

- *Command/alias:* manual / m
- *Help:* geoelan manual --help
- *Basic usage:* geoelan manual --pdf

manual exports or prints the contents of this file to screen, bute the full PDF-manual is also embedded within the compiled executable for convenience. Running geoelan manual with no flag prints the full manual to screen.

Flags

Short	Long	Description
		Print full plain text version to screen
	`--pdf`	Save the full manual as a PDF to current directory
	`--pdf-a4`	Save the A4-guide as a PDF to current directory

Appendix

A few notes and help texts on ELAN, telemetry formats, video processing etc.

References

Larsson, Jens, Niclas Burenhult, Nicole Kruspe, Ross. S Purves, Mikael Rothstein and Peter Sercombe. 2020. Integrating behavioral and geospatial data on the timeline: towards new dimensions of analysis. *International Journal of Social Research Methodology*. doi: 10.1080/13645579.2020.1763705

ELAN (Version 6.4) [Computer software]. 2022. Nijmegen: Max Planck Institute for Psycholinguistics. Retrieved from https://archive.mpi.nl/tla/elan

GoPro and Garmin telemetry formats

As of GeoELAN v2.0 GoPro Hero 5 Black and newer is supported. There are significant differences between Garmin's FIT-format and GoPro's GPMF-format. Here are a few:

	Garmin FIT	GoPro GPMF
Storage form	Separate file (binary)	Embedded in MP4 (binary)
Time stamps	Explicit, absolute time stamps for each data point	Absolute time stamps for GPS log only, otherwise have to be derived from MP4 timing
GPS	10Hz, individual points time stamped	10 or 18Hz, logged in one-second clusters. Only the cluster is time stamped.

For performance reasons both GoPro GPMF and Garmin FIT are binary formats, and thus can't be viewed in a text editor.

Documentation and development

Support for GPMF (GoPro) and FIT (VIRB) formats were written from scratch for GeoELAN with the help of the official documentation for both formats.

- Garmin FIT development kit and documentation: https://developer.garmin.com/fit/
- GoPro GPMF documentation and example code: https://github.com/gopro/gpmf-parser

GoPro

File structure

GoPro recording sessions are split over multiple clips depending on recording time, quality settings, and SD card size. GoPro provide an estimate here: https://community.gopro.com/s/article/GoPro-Camera-File-Chaptering-Information?language=en_US

SD Cards

GoPro cameras use micro SD Cards. GoPro provide recommendations here: https://community.gopro.com/s/article/microSD-Card-Considerations?language=en_US.

Internal file layout

Since Hero 5 Black all GoPro cameras use a telemetry format called GPMF developed by GoPro. Currently, the best overview can be found in their Github repository: https://github.com/gopro/gpmf-parser. (Note that GeoELAN does not make use of GoPro's parser, but a GPMF-parser developed from scratch tailored for our needs).

Garmin VIRB

Note that the Garmin VIRB Ultra 30 is no longer available for purchase. Garmin currently has no replacement product.

File structure

Example VIRB SDCard file structure:

```
├── DCIM
│   └── 100_VIRB
│       ├── VIRB0001-1.GLV          Recording session split up
│       ├── VIRB0001-1.MP4          as 10 minutes clips. Low (GLV)
│       ├── VIRB0001-2.GLV          and high (MP4) resolution clips.
│       └── VIRB0001-2.MP4
└── GMetrix
    ├── 2017-01-01-12-00-00.fit     Telemetry files, a.k.a "FIT-files".
    └── 2017-01-02-12-00-00.fit     May contain data, such as GPS-logs,
                                    for multiple recording sessions.
```

The VIRB and the FIT-format

To pair and match VIRB video clips belonging to the same recording sessions with a FIT-file unique identifiers (UUID) are embedded both within the original video clips and the FIT-files. Preserving these are key to synchronise and extract relevant GPS-data.

When synchronising and locating data, GeoELAN will sometimes list all sessions present in the FIT-file. As a help, the number of video clips and the *UUID for the first clip* in each session is listed.

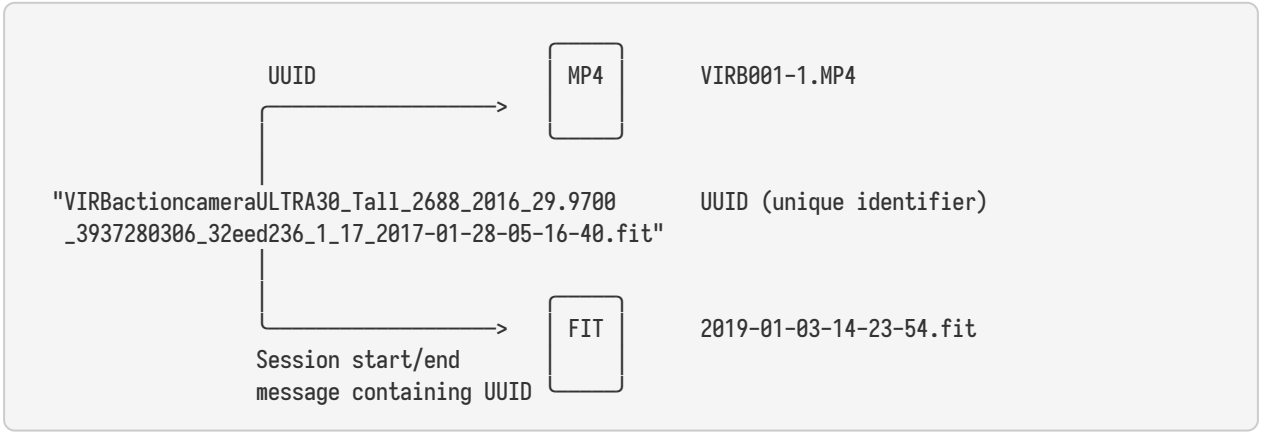
A single FIT-file may contain telemetry for multiple recording sessions. When the camera is turned on, it immediately starts logging data into a new FIT-file, regardless of a video being recorded or not. The camera will keep logging to this file until completely turned off. If turned on again, a new FIT-file will be created. All data points in a FIT-file are explicitly timestamped, which technically allows synchronisation against any data type in the file. Further, with the help of the built-in GPS, absolute timestamps can be derived for all data types. These can be used for documentation purposes or to synchronise against external data sources.

For geo-referenced annotations, GeoELAN always embeds absolute timestamps in the resulting KML-file.

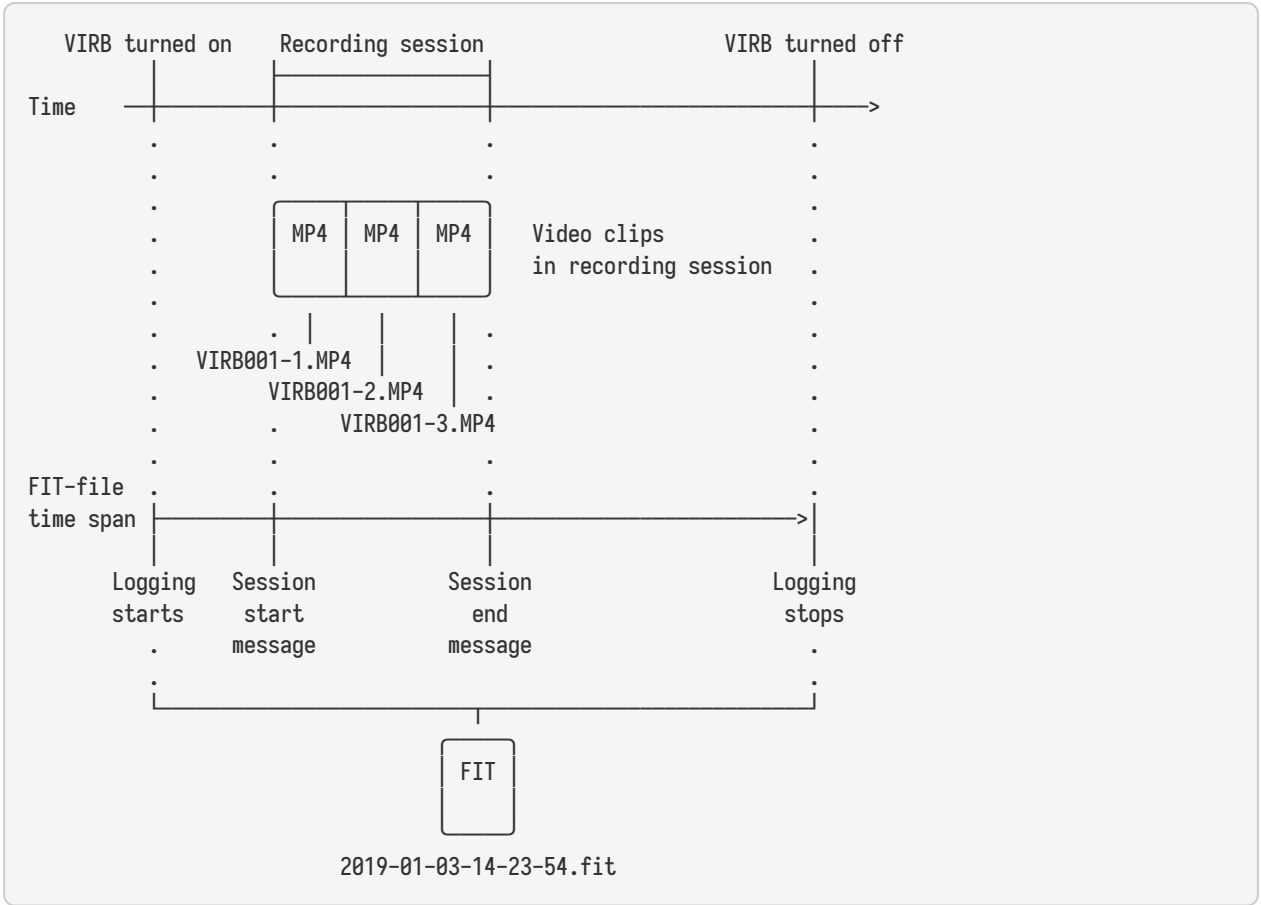
The VIRB cameras split up recording sessions into video clips, each approximately 10 minutes in length, with no option to turn this off. To link VIRB video to its corresponding telemetry (e.g. coordinates logged by the GPS during the recording session), both the clips and the FIT-file contain UUIDs. When the user starts recording, a "video recording session start" message is logged to the

current FIT-file together with the UUID embedded in the first clip, denoting the start of a recording session. Similarly, when recording ends, a "video recording session end" message is logged together with the UUID embedded in the last clip in the session. Since all logged FIT-data is timestamped, this creates a timeline for the session that can be related to any logged data in the FIT-file.

Matching MP4 and FIT-files via embedded UUIDs



Logging telemetry and boundaries for a recording session in a FIT-file



The VIRB logs location, barometric pressure, and rotation among many other data types. Since the FIT-format is not a text based data format, and thus cannot be inspected using a text editor, the `inspect` command allows for some exploration of a FIT-file (see command *inspect*). GeoELAN will also help out with matching recording sessions to the corresponding FIT-files (see commands *virb2eaf*, and *locate*).

Preserving UUIDs

Concatenating or converting the video clips will usually discard the UUIDs, so the user is advised to save the original video clips. The `inspect` command can be used to display the UUID for a specific VIRB MP4-file, just run `geoelan inspect --video VIRBVIDEO.MP4` with no other options.

Most of the commands allow for selecting UUID from those present in the relevant FIT-file when matching files or geo-referencing annotations. The `locate` command can also be used to locate all files for a specific session.

Video file management and options

On the VIRB MicroSD card, the low-resolution clips have a `.GLV` extension. These are generated by the VIRB for quick viewing on the internal camera display. If available, GeoELAN will prefer to link these in the ELAN-file over the high-resolution video due to their smaller size (both resolutions will still be concatenated by default). GeoELAN will not be able to identify the low-resolution `.GLV` as such if renamed to `.MP4` and they may even be mistaken for the high-resolution versions. If you only require the low-resolution videos to be concatenated, use the `--low-res-only` flag when running `virb2eaf`. This will ignore the high-resolution `.MP4`-files as a concatenation target, with an option to copy these as-is (`--copy`) to the output directory (see the *virb2eaf* section for further information).

FFmpeg

The `gopro2eaf` and `virb2eaf` sub-commands require FFmpeg for concatenating MP4-clips and to extract the audio track as a WAV-file (required to display a wave form in ELAN while annotating).

The video and audio streams are by default only concatenated, not converted, to avoid data loss and to save time, but note that **VIRB UUID and GoPro telemetry will still be discarded - save the original files**. See the *virb2eaf* section for more information.

There are two main options for installing FFmpeg:

1. Download the *static build* of FFmpeg, and specify its path using the `--ffmpeg` option
2. Install via a *package manager*. FFmpeg will be automatically available to `cam2eaf` in this case.

> **Static build** The *static build* option means that the relevant media codecs are included in a single, executable file that can be used as is. The FFmpeg download page provides links to static builds for macOS, Windows and Linux. Put the downloaded `ffmpeg`-file in a convenient location and use the `--ffmpeg` option when running `gopro2eaf` or `virb2eaf`. Optionally moving or symlinking this file to a directory in PATH will yield the same result as using a package manager below.
>
> **Package manager** Installing via a *package manager* means the `ffmpeg` command can be executed from anywhere in a terminal. Linux distributions usually come with one pre-installed. For macOS Homebrew is a popular choice, whereas Windows has Chocolatey (or WSL). This option means you do not have to specify the location of `ffmpeg` each time `gopro2eaf` or `virb2eaf` is run. If a package manager is not for you, go with the *static build* for your platform.

ELAN

ELAN is a completely free, advanced tool for time-aligned annotations of audiovisual media developed by the The Language Archive, Max Planck Institute for Psycholinguistics in Nijmegen. While it is well-known in academia, and particularly in the humanities for transcribing recordings, its use goes well beyond this, since anything observed can be annotated, and thus time-aligned (GeoELAN is but one example). Since annotations are kept aligned in parallel on separate tiers - as many as required, similar to multi-track audio editors - the possibilites are almost endless. The ELAN Annotation Format is XML-based which makes it both human-readable and fairly straightforward to parse.

- Download (Windows, macOS, Linux): https://archive.mpi.nl/tla/elan/download
- Documentation: https://archive.mpi.nl/tla/elan/documentation

GeoELAN Rust crates

GeoELAN is written in Rust and uses four custom libraries (aka crates) that were developed from scratch in parallel with the tool itself. This was partly due to the need to get to know the formats better, partly because the only existing option was to create bindings to existing libraries in several other programming languages.

Since these Rust crates are still in development they are not yet available on crates.io, but can be specified as a git resource in `Cargo.toml` (see the respective repository URLs).

Three + one crates were developed for GeoELAN:

- `eaf-rs`
  - Read, write, and process EAF-files. Uses quick-xml and its serialization support via serde.
  - Repository: https://github.com/jenslar/eaf-rs
- `gpmf-rs`:
  - Read GoPro GPMF-files. Either directly from GoPro MP4-files, or "raw" GPMF-files (e.g. a track extracted to file with FFmpeg).
  - Repository: https://github.com/jenslar/fit-rs
- `fit-rs`:
  - Read Garmin FIT-files. Supports custom developer messages. Some added bloat due to VIRB-specific functionality that seemed better to include in `fit-rs` directly.
  - Repository: https://github.com/jenslar/fit-rs
- `mp4iter`:
  - A simple crate to move around, search atoms on FourCC, and read values in an MP4 container. Includes finding `mdat` byte offsets for tracks via atom `hdlr` component name, and deriving duration for the longest track without the need for FFmpeg or mediainfo. This crate does not (and will not) support any kind of media en/decoding. See https://developer.apple.com/library/archive/documentation/QuickTime/QTFF for more information on the QuickTime/MP4 container. It may well be that there are much better options out there.
  - Repository: https://github.com/jenslar/mp4iter

Data extracted with both `gpmf-rs` and `fit-rs` will mostly require further processing. Support for this is built-in for some data types (e.g. GPS data, since this is fundamental for GeoELAN, some processing of

sensor data as well), but for others you will have to develop and expand on this yourself. A first pass, extracting and parsing data, should always work for both crates. GeoELAN's `inspect` command with the `--verbose` flag or `--type` option prints data in this "raw" form.

GeoELAN A4 Guide, v2.7, 2024-02-29

Jens Larsson jens.larsson@humlab.lu.se

`geoelan --help`	**General help**
`geoelan COMMAND --help`	Help for specific COMMAND, e.g. `geoelan eaf2geo --help`

	Workflow	**Example**
1.	Concatenate media files, generate ELAN-file	`geoelan virb2eaf --video VIRB0001-1.MP4 --indir DIR_TO_SEARCH/ --outdir OUTDIR/`
2.	Annotate the generated ELAN-file. One tier for each event type.	
3.	Geo-reference annotations on selected tier. Outputs KML/GeoJSON.	`geoelan eaf2geo --eaf VIRB0001-1.eaf --fit FITFILE.fit --geoshape point-single`

Command	**Description**	**Example**
`cam2eaf`	[GoPro] Generate ELAN-file with geo-tier	`geoelan cam2eaf -v VIDEO.MP4 -i INDIR/ -o OUTDIR/ --geo-tier`
`cam2eaf`	[VIRB] Generate ELAN-file with geo-tier	`geoelan cam2eaf -v VIDEO.MP4 -i INDIR/ -o OUTDIR/ --geo-tier`

Command	Description	Example
`eaf2geo`	Geo-reference annotations and generate KML-file	`geoelan eaf2geo -e ELANFILE.eaf -f FITFILE.fit --geoshape point-single`
`locate`	Locate and match MP4 and/or FIT-files	`geoelan locate -i INDIR/ --csv`
`inspect`	Print the contents of a FIT/GPMF-file	`geoelan inspect -f FITFILE.fit --verbose` `geoelan inspect -g GOPRO_VIDEO.mp4 --verbose`
`manual`	View or save manual as PDF	`geoelan manual --pdf`

Argument	Description	Applicable to	Possible values
`--downsample`	Point output divisor (e.g. `10`: 7200 points → 720 points)	`gopro2eaf`, `virb2eaf`, `eaf2geo`	1 (default) to max number of logged points
`--geoshape`	Point/s or line/s in output KML	`eaf2geo`	See table below
`--cdata`	Extended information bubble in Google Earth	`eaf2geo`	

`geoshape` value	Description	Shape	Note
`point-all`	Points intersecting with an annotation will gain a description	Points	
`point-multi`	Points intersecting with an annotation will be exported	Points	
`point-single`	Each annotation will be averaged to a *single point*	Points	Ignores `--downsample`

geoshape value	Description	Shape	Note
line-all	Points intersecting with an annotation will gain a description	Line, continuous	
line-multi	Each annotation will be exported as a line	Line, broken-up	
circle-2d	Each annotation will generate a circle (c.f. point-single)	Circle	Ignores --downsample
circle-3d	Each annotation will generate a cylinder (c.f. point-single)	Circle	Ignores --downsample

References

Larsson, Jens, Niclas Burenhult, Nicole Kruspe, Ross. S Purves, Mikael Rothstein and Peter Sercombe. 2020. Integrating behavioral and geospatial data on the timeline: towards new dimensions of analysis. *International Journal of Social Research Methodology*. doi: 10.1080/13645579.2020.1763705

ELAN (Version 6.4) [Computer software]. 2022. Nijmegen: Max Planck Institute for Psycholinguistics. Retrieved from https://archive.mpi.nl/tla/elan