flight-analyzer

Python 3.9,3.10,3.11,3.12 License INDIVIDUAL Testing formatting Black linting Pylint

Overview

The `flight-analyzer` program is part of the scientific paper "Fliegen am Limit - Aktive Sicherheit im Gleitschirmsport", that was first published on 10/24/2022 and is being further developped by 03/31/2024 as of the "Schweizer Jugend forscht 2024" initiative. The application automates the analysis of paragliding flights and contains a selection of algorithms to process tracklogs. As part of the paper, this tool is designed to deliver a clean dataset that can be used to conduct and optimize advanced analyses and simulate the stationary glide of a paraglider. Please find a detailed description of the algorithms as well as concepts and findings in the sections below, in the paper itself or in the code.

The original paper (10/24/2022) can be downloaded here: nicolas-huber.ch/docs.

As soon as the refined version of the 2022 version has been published, the link will be listed here. Until then, please refer to the original paper.

The documentation is available in `.pdf` format and can be downloaded here.

Contents

Technical documentation

Introduction

The goal of the `flight-analyzer` application is to automatically manipulate large datasets, which contain the track logs of paragliding flights. The program evaluates for each point of the flight if it is on a straight line or not, which is relevant for modelling the stationary glide of a paraglider. This allows further analyses and the filtering by position of the point. After the input dataset has been filtered the tool can apply a selection of algorithms, which have been developped as of the "Fliegen am Limit" paper, to simulate the stationary flight of a paraglider. This serves as a foundation for the development of new algorithms, more advanced models and to conduct further analyses. In addition, the tool offers some helpers such as tools to visualize the manipulated data or preprocess raw input files.

Detailed descriptions can be found in docstrings and comments within the source code of this project.

Getting started

Make sure to install `Python 3.9` or higher on your machine. The code has only been tested for the Python versions 3.9, 3.10, 3.11 and 3.12-dev and should properly work on MacOS, Ubuntu Linux and Windows. It's recommended to use pyenv to manage local python environments as well as dependencies. To run this project make sure to activate an environment that supports Python 3.9 or higher and then run `pip install -r requirements.txt`. The application should work fine with the dependencies (indicated version) listed in `requirements.txt`.

Architecture

The application is structured as follows:

```
┌ flight-analyzer
│ ⟶ .github/
│ ⟶ assets/
│ ⟶ docs/
│   ⟶ datasets/
│   ⟶ reports/
│ ⟶ src/
│   ⟶ algorithms/
│   ⟶ executor/
│   ⟶ helpers/
│   ⟶ constants.py
│ ⟶ tests/
│ ⟶ main.py
│ ⟶ main.ipynb
│ ⟶ LICENSE.md
│ ⟶ README.md
│ ⟶ requirements.txt
└
```

Basic Usage

The main entry points of this application are the `main.py` and `main.ipynb` files. Both the Python file and the Jupyter Notebook output the same result - you can chose the script that fits you best. If you'd like to visualize data or run a specific analysis check out the `src/executors/` directory, which contains a collection of Jupyter Notebooks. Please find listed below some instructions for the Notebooks in `src/executors/`, the `main` scripts and some guidenines for preprocessing.

Preprocessing flight data

The flight data consumed by this program is read from `.igc files`, which are written by flight computers such as variometers or similar. Reading these files is currently not supported in this program, which is why external applications are used. It's recommended to use IGC2KML for `.igc` to `.kml` conversion and KML2CSV for `.kml` to `.csv` conversion as well as Microsoft Excel to clean up the raw table data.

After processing the tracklog using the linked tools you're dealing with `.csv` data of the following format:

▶ Show Data

```
name,description,altitudeMode,visibility,tessellate,WKT
<<< SOME RANDOM HTML>>>
"12:25:30 0m 5kmh 0m/s 0km",,"clampToGround",,"true","LINESTRING Z
(7.530683 46.213083 2612, 7.5307 46.213083 2612)"
"12:25:31 1m 0kmh +1m/s 0km",,"clampToGround",,"true","LINESTRING Z
(7.5307 46.213083 2612, 7.5307 46.213083 2612)"
...
```

This data is to be processed using a tool like Microsoft Excel to remove the HTML content that's marked as `<<< SOME RANDOM HTML>>>` as well as clean up the data. After your final manual preprocessing steps the `.csv` data is supposed to look like this:

▶ Show Data

```
name,description,altitudeMode,visibility,tessellate,WKT
12:25:30 0m 5kmh 0m/s 0km,,clampToGround,,TRUE,"LINESTRING Z (7.530683
46.213083 2612, 7.5307 46.213083 2612)"
12:25:31 1m 0kmh +1m/s 0km,,clampToGround,,TRUE,"LINESTRING Z (7.5307
46.213083 2612, 7.5307 46.213083 2612)"
```

Please check your input data before running any of the algorithms in this application to prevent unexpected errors. The preprocessed input is consumed by the `main` scripts in `.csv` or `.xlsx` format. It can also be manually fed into the `FileConvertor` helper class, on which further records can be found here.

Running main.py / main.ipynb

In order to run the main script of the `flight-analyzer` application make sure to prepare all necessary files and properly preprocess them. Please find instructions on preprocessing here.

Further documentation will follow as soon as the `main` script have been developped and properly tested.

Run individual analysis

To run a particular algorithm of the `flight-analyzer` application, e.g. to visualize data, please refer to the executor scripts that can be found in `src/executors/`. The algorithms and some examples can be found in the Algorithms and helpers section.

This application provides the following executor Notebooks:

- `execute_file_convertor.ipynb`: This Notebook allows you to normalize the manually preprocessed data.
- `execute_angle_analyzer.ipynb`: This Notebook allows you to process a normalized dataset. The AngleAnalyzer algorithm is applied.
- `execute_data_analyzer.ipynb`: This Notebook allows you to extract points on a straight line from a processed dataset based on the AngleAnalyzer algorithm.
- `execute_optimize_thresholds.ipynb`: This Notebook allows you to automatically determine the best thresholds and constants to process your dataset (run before data analysis).

Further documentation on the inputs and outputs of these executors can be found in the Notebooks.

Algorithms and helpers

FileConvertor

Description of FileConverter helper class.

AngleAnalyzer

Description of AngleAnalyzer algorithm class.

input

▶ Conditions

```
INDEX: int = 1400 # point to be analyzed

ANGLE_PAST_THRESHOLD: int = (
    80  # number of points in the past that are considered for the angle
evaluation
)
ANGLE_FUTURE_THRESHOLD: int = (
    35  # number of points in the future that are considered for the angle
evaluation
)
ANGLE_THRESHOLD: int = 20  # angle < 20° is considered as straight line
LINEAR_REGRESSION_THRESHOLD: float = 0.9  # r-value > 0.9 is considered as
straight line
```

Punktvariation mit relativer Höhe *Fig. 1: Point variation with relative height*	Winkelvariation *Fig. 2: Angle variation*
Lineare Regression (Vergangenheit) *Fig. 3: Linear regression (past)*	Lineare Regression (Zukunft) *Fig. 4: Linear regression (future)*

▶ Output

```
Angle Analysis
--> Past: True
--> Future: True

Past Linear Regression
--> Status: True
--> Slope: 0.24496415928278734
--> Intercept: 44.41611343989336
--> R-Value: 0.9970350725539303
--> P-Value: 1.2004630598704919e-88
--> Standard Error: 0.0021406452009797685

Future Linear Regression
--> Status: True
--> Slope: 0.11306255384150318
--> Intercept: 45.41304614404877
--> R-Value: 0.9736381906284354
--> P-Value: 9.2965712930775e-23
--> Standard Error: 0.004610898882940768

Data Analysis
--> Status: (True, 'Straight Line', 0)
```

DataAnalyzer

Description of DataAnalyzer helper class.

input

▶ Conditions

```
ANGLE_PAST_THRESHOLD: int = (
    80  # number of points in the past that are considered for the angle evaluation
)
ANGLE_FUTURE_THRESHOLD: int = (
    35  # number of points in the future that are considered for the angle evaluation
)
ANGLE_THRESHOLD: int = 20  # angle < 20° is considered as straight line
```

```
LINEAR_REGRESSION_THRESHOLD: float = 0.9  # r-value > 0.9 is considered as
straight line
```

Kategorisierung der Punkte

Fig. 5: Point position and categorization

▶ Output

```
You lost 115 rows of data due to processing. The data loss is supposed to
be 115 rows, which can be calculated by adding the ANGLE_FUTURE_THRESHOLD
and the ANGLE_PAST_TRESHOLD.

The average accuracy of the AngleAnalyzer algorithm and the past / future
tresholds of 80 / 35 for points on a straight line can be defined as
follows:
--> average r_value: 0.87
--> average p_value: 0.0
--> average std_err: 0.01

A linear regression can be considered as a good fit if the r_value is
close to 1, the p_value is close to 0 and the std_err is close to 0.

The system found 2487 points on straight lines, whereas the amount of
points on a curve is 6114. The expected amount of points on a curve is
6114, which can be calculated by subtracting the count of points on a
straight line from the total point count.

In total, you lost 71.08% of the data after applying the AngleAnalyzer
algorithm as you can only use the points on a straight line for further
processing.
```

ThresholdOptimizer

Description of ThresholdOptimizer helper class.

input

▶ Conditions

```
R_VALUE_WEIGHT: float = 0.6  # weight of the r-value in the optimization
P_VALUE_WEIGHT: float = 0.3  # weight of the p-value in the optimization
STD_ERROR_WEIGHT: float = 0.1  # weight of the standard error in the
optimization

OPTIMIZATION_LIMIT: int = 30  # upper limit of optimization loops
OPTIMIZATION_STEPS: int = 5  # step size per optimization loop
OPTIMIZATION_RUNTIME_ESTIMATION: int = 120  # estimated runtime per loop
in seconds
```

Score und Datenverlust
Fig. 6: Optimization score and data loss

Optimierung der Thresholds
Fig. 7: Threshold optimization

▶ Output

```
Individual thresholds with the best score:
--> past_threshold_optimized: 25
--> future_threshold_optimized: 25

Below is a tabular overview of the 5 best scores and their thresholds.
This information is more meaningful here, as in the analysis later for the
evaluation of a point, both the future and the past are taken into
account, and thus the score considers the interaction of the two
thresholds.

<<< HEADER >>>
0   25  25  0.772521    9.392513e-12    0.012124    0.462300    62.820217
1   25  20  0.753950    2.555986e-10    0.015246    0.450846    61.180948
2   25  15  0.745859    1.490066e-08    0.018904    0.445625    59.843246
3   20  25  0.740292    3.472833e-10    0.014465    0.442729    60.581248
4   25  10  0.724441    2.517912e-06    0.021627    0.432501    58.299735

The best performing thresholds are 25 (angle_past_threshold) and 25
(angle_future_threshold) with a score of 0.4623003930101672.

Another good performing set of thresholds can be found by comparing the
data loss relative to the scores, which are directly related to the
thresholds. In this case, the best performing thresholds are 25
(angle_past_threshold) and 15 (angle_future_threshold) with a score of
0.44562524837811207 and a data loss of 59.84324573536192. The bigger the
difference between the score and the data loss, the better the thresholds
are. This is the case because the precison of the thresholds is overall
better if less data is lost, even if there is a small decrease in the
score.
```

Development

Conventions

Please find naming conventions for this project linked here: click. In addition, static type annotations are used in this project. The codebase has been tested using the `pytest` module. The recent CI/CD status can be found at the top of this page. Click here for a detailed overview and unit testing logs. The code is formatted and linted in VS Code using the Black Formatter Extension and Pylint.

Contributing

At this time, the `flight-analyzer` project is not open for community contributions. The development is currently handled exclusively by Nicolas Huber. Your interest is appreciated and this section will be updated if the policy changes in the future.

Changelog

- **[1.0.0]** - Not released yet.

License & Intellectual Property

The source code of this application is licensed under the license linked here.

If not stated differently, the source code of this project is Nicolas Huber's intellectual property. External sources can be found in the code and are marked as such. Additionally, to improve code quality and speed up workflows, tools like GitHub Copilot and ChatGPT were used. AI generated content is flagged with the following notes:

- For documentation files: *This document { TITLE } has been written by { SOURCE } and verified by Nicolas Huber on { DATE }.*
- For code snippets: *# AI content ({ SOURCE }, { DATE }), verified and adapted by Nicolas Huber.*

AI tools are a powerful and valuable addition to improve the development workflow, as long as sources and contents are scientifically listed. Thus, it's valued a lot to provide proper listings. The following utilities have been used: GitHub Copilot, ChatGPT.

In consideration of the `LICENSE.md`, the licensee, who is considered as such at the point of downloading this application, agrees to respect the terms and conditions. The licensee undertakes to show respect for Nicolas Huber's intellectual property and to use it only in accordance with his instructions.

Thanks for noticing!

Disclaimer

The author is not responsible for any damage caused by the use of the software.
