**Texte explicatif du fonctionnement du code**

Le code est regroupé dans une catégorie nommée « VideoTreatment », qui contient les outils qui vont nous servir à différents endroits, et les codes qui permettent l’analyse vidéo et la production du résultat souhaité.

Voici la théorie lorsqu’on lance l’exécution du code :

1. On lance le main
2. On appelle la classe XML Handler, qui renvoie les noms de fichiers vidéo à analyser (au format .avi) ou rien s’il n’y en a pas
3. Sur chacune de ces vidéo, on exécute VideoProcessor (de VideoConstructor) qui prend en entrée le chemin de la vidéo, et qui retourne les données dans un fichier txt ou csv, nommé par le nom du fichier vidéo.
4. On entre alors dans VideoFrameDisplay, on fait un passage dans VideoDataAnalyzer pour faire l’interpolation sur les données récupérées dans le csv (ou le txt), on ressort de VideoDataAnalyzer avec le numéro de frame qu’on veut.
5. Dans VideoFrameDisplay, on récupère la frame voulue avec son numéro.
6. On passe par VideoDirectoryTool pour créer un dossier qui y met cette frame

* VideoTreatment

**CircleLEDExtractor** : Code that, when executed, returns a frame (chosen number) from the video with the LED, and also an image with a black mask over the entire image except on the analysed area (circular).

**Pointer**: Code that allows you to point a location on a frame from the rear camera video and retrieves its x and y coordinates. This is used to define the areas to analyse to retrieve data (km, hour, date, speed).

**Rectangular\_preview** : Same concept as CircleLEDExtractor, on one side the frame from the “video\_arriere” video, on the other side a black image except in the rectangular analysed area.

**Base** : Code that groups various error/information messages during code execution.

**Modules**: Code that groups the importation of various modules to be used.

**Settings** : Code that groups the different parameters used (the Pytesseract parameters are separate from those used for video analysis choices).

**Main** : Code to be executed to retrieve the desired data (frames of interest, list, or dictionary (txt file)). It links the XML file and video analysis, retrieves the video files of interest (via functions implemented in VideoXMLHandler), and proceeds with the analysis.

**Find\_Peaks\_Interpolation** : Code that interpolates detected peaks on the red LED and plots the regression line closest to each detected peak. It displays on the same graph the LED’s step (each time it lights up, the step goes up) and the regression passing through the end of each step.

**Interpolation** : Contient plusieurs fonctions.

* Convert\_to\_hour\_minute\_second that converts a time in seconds to a time in HH:mm:ss
* Convert\_to\_seconds that performs the inverse operation of Convert\_to\_hour\_minute\_second
* **JSP**

**LED\_on\_multiprocess** : Contains several functions, aiming to minimize computation time by allowing multiple functions to run in parallel. Time markers are used to get the total execution time.

* \_init\_ that initializes the processing (retrieves intrinsic video data, the number of frames, fps, etc.)
* Is\_led\_on that returns a Boolean of the LED state (0 or 1)
* Process\_frame that analyses a frame, and returns its number, its state (always 0 or 1), as well as the average states 0 or 1 of the LED area. **PAS SUR**
* Generate\_mask that creates a round mask for analysing a circular area to detect the LED’s on or off state
* Process\_video that uses Process\_frame on all frames of a video to analyse them, using multiprocessing (via ThreadPoolExecutor). It retrieves the different RGB triplets from the LED area.
* Plots\_results that groups the display of the RGB values’ evolution of the LED

**VideoConstructor** : Contains many functions. Contains the VideoProcessor class, which is a crucial part of the code. In this class, we find the following functions :

* \_init\_ that initializes data, computation times, path name, data (which are thus empty lists)
* Rewrite\_marker\_format that corrects Pytesseract analysis errors. Sometimes a 4 is detected instead of a +, os this is automatically corrected.
* Convert\_ms\_to\_time\_format that converts a time in milliseconds to a time in HH:mm:ss
* Convert\_time\_format\_to\_ms that performs the inverse operation of convert\_ms\_time\_format
* Progress\_bar that provides an overview of the remaining computation time
* Measure\_time that retrieves the time of each part of the analysis (useful for knowing where to save computation time)
* Initialize\_video\_properties that retrieves the main info (dimensions, frames, duration) of the video
* Read\_frame that reads a single frame
* Detect\_change that compares 2 matrices (2 images), and finds the difference between the two to return whether or not the image has changed (useful for saving computation time).
* Extract\_data\_from\_frame that retrieves different data, analysing the requested area only if a change has been detected compared to previous frame (via detected\_change).
* Get\_text that retrieves data in the form of strings
* Save\_data that saves data in various formats as desired by the user (csv, list, dictionary, txt)
* Get\_filename\_without\_extension to rename the desired file with the video name
* Process\_video that depends on user input to return the correct format and thus analyses the entire video and retrieves the data.
* Cleanup that releases all data we were handling after video analysis.
* Display\_changes that shows changes (or not) over time of different data (for example, the date never changes, but the milestone almost changes at every frame).
* Display\_results that shows the breakdown of computation time according to what is being analysed (milestone, date, time)

**VideoDataAnalyzer** : contains several functions aiming to analyze the data retrieved with VideoConstructor

* \_init\_ to initialize variables that will be used later.
* Convert\_time\_to\_seconds that converts time in the “HH:mm:ss” format from the txt file to seconds.
* Convert\_marker\_to\_meters that converts the milestone (KM+meters format) to meters (for instance : 005+768 -> 5768)
* Read\_data\_from\_file that reads data from the text fil created after analysis
* Calculate\_regression that creates a dictionary with frame numbers as keys, and time or kilometers as values.
* Get\_frame\_number to retrieve the frame number
* Plot\_markers, plot\_time\_progression and plot\_speed to display the evolution of these data over time, and detect any outliers if there are any.

**VideoDirectoryTool** : Code that groups several useful features for later use:

* \_init\_ to initialise variables
* Get\_desktop\_path to retrieve the path to a video fil (which will be coordinated with another method to be completed by the desired video file)
* Delete\_dir et get\_or\_create\_dir to delete or create the directory **PAS SUR**
* **JSP**

**VideoFrameDisplay**: function that outputs the result desired by the user (eg, the frame when given time is provided)

* \_init\_ that initialises the variables used afterward.
* Display\_frame\_for\_time which returns the frame corresponding to the time specified by the user.
* Realease\_ressources that releases the data we were mobilizing.

**VideoXMLHandler** : Code that contains various functions to analyse the given XML file. For a date and time provided by the user, the code returns the video files to be analysed.

* Check\_date\_format verifies that the format used by the user is correct according to the XML file.
* Parse\_datetime that checks if the specified time contains milliseconds, in which case it rounds and formats it correctly.
* Get\_seconds\_from\_time that analyses the time and returns the total seconds (eg, if it is 01:04:27, it converts to retrieve 3600+4\*60+27 seconds)
* Time\_to\_video which goes through the entire XML file and retrieves the video files containing the specified time, to then process theses files via other codes.
* Get\_all\_video\_files retrieves all video files to analyse (user option)
* Videos\_in\_tie\_interval retrieves all video files within a specified time interval A to B provided by the user