Video processing pipeline to concat and accelerate flood videos from Google Cloud Storage bucket

Choose base directory

In [1]:

```
cd ../
```

```
C:\Users\luisr\Desktop\Repositories\Data Science Projects\Hackaton COR IV - Centro de Operações do
RJ\INCUBAÇÃO\Cameras
```

Define utility functions

In [2]:

```
from time import time

# Simples class to report execution time

class Timer:
    def __init__(self):
        self.start = time()
    def end(self, decimals=4):
        end = time() - self.start
        print('\n* TIME TO EXECUTE:', round(end, decimals), 's')

# Get blob count, bytes and names from Google Cloud Storage bucket

def gcs_list_folder(folder, ext, bucket_name, print_each=10):
    prefix = folder
    delimiter = None
    names = []
    timer = Timer()
    blobs = gcs.list_blobs(prefix, delimiter, bucket_name)
    for i, blob in enumerate(blobs):
        if blob.name.endswith(ext):
            names.append([blob.name, blob.size])
        if print_each is not None and (i + 1) % print_each == 0:
            print(f'\n- Blobs Searched: {i + 1}'); co(True)
    names = pd.DataFrame(names, columns=['blob_name', 'bytes']) # build blobs dataframe
    if print_each is not None:
        print(f'\n- Blobs Searched: {i + 1}')
        print(f'\n  · Blobs (Matched): {len(names)}')
        print(f'\n  · Giga Bytes (Matched): {round(names["bytes"].sum() / 1e9, 3)} GB')
        timer.end() # prints time to execute
    return names
```

Pipeline methods set up

Import Google Cloud Storage wrapper module and define storage instance

In [3]:

```
from modules.googlecloudstorage import GCS

sa_json = '../../../../Apps/APIs/octa-api/credentials/octacity-iduff.json'
user_project = None
default_bucket_name = 'flood-video-collection'

gcs = GCS(sa_json, user_project, default_bucket_name)
```

Write videos with opencv set up

In [4]:

```
from modules.video import VideoWriter

# Video writer class instance

writer = VideoWriter(fps=3, shape=(854, 480), codec='mp4v')

# Accelerated video writer class instance

writer_speed = VideoWriter(fps=24, shape=(854, 480), codec='mp4v')
```

```
c:\Users\luisr\anaconda3\lib\site-packages\pandas\core\computation\expressions.py:20: UserWarning:
Pandas requires version '2.7.3' or newer of 'numexpr' (version '2.7.1' currently installed).
  from pandas.core.computation.check import NUMEXPR_INSTALLED
```

Video writer class funcitonality

1. Add running clock to video files
2. Concatenate video files from nested folders
3. Accelerate video files from nested folders

Count blobs with .mp4 extension and total file bytes of download

Import python modules

In [5]:

```
import pandas as pd
from IPython.display import clear_output as co
```

Pipeline Execution

Step 0 · Pipeline parameters set up

Download from Google Cloud Storage bucket

In [6]:

```
"""
 - prefix: Folder in any level of the bucket containing sub-folders with videos to feed the pipeline
 * Note: Forward trailing slashses, i.e. `/`, at the end of `prefix` limits results to folders
matching exactly to `prefix`. Otherwise, matches any folder or blob that contains `prefix`.
"""

prefix = 'comando/'
delimiter = None

bucket_name = 'flood-video-collection' # collection bucket name
folder = 'Dados/flood-video-collection' # bucket collection destination folder

overwrite_download = False
```

Annotate videos timestamps

In [7]:

```
folder = 'Dados/flood-video-collection'  # local collection source folder
to_folder = 'Dados/flood-video-collection-stamped' # `time-stamped` local collection source folder
ext = '.mp4' # video file format to search for in nested folders
overwrite_annot = False
```

Concatenate and accelerate videos from folders

In [8]:

```
base_folder = 'Dados/flood-video-collection-stamped' # `time-stamped` local collection source folder
to_base_folder = 'Dados/flood-video-collection-date' # concatenated local collection destination folder
overwrite_concat = False
```

General purpose parameters

In [9]:

```
ext = '.mp4'
report_freq = 5
```

Step 0.1 · Count blobs and download bytes · *Preparation Step*

In [10]:

```
blobs = gcs_list_folder(prefix, ext, bucket_name, print_each=1000)
```

```
- Blobs Searched: 3250

  · Blobs (Matched): 3250

  · Giga Bytes (Matched): 3.264 GB

* TIME TO EXECUTE: 3.069 s
```

Step 1 · Download blobs in Cloud Storage bucket to folder

Download blobs in `bucket_name` matching `prefix` to local `folder`

In [11]:

```
timer = Timer()

gcs.download_to_folder(
    folder, prefix, delimiter, bucket_name,
    overwrite_download, report_freq
)

timer.end()
```

```
PREFIX: comando/ · RUNNING: 36.0 min · RATE: 0.6649 s / file · FINISH-ESTIMATE: 0.0 min · PROGRESS:
3250/3250 · DOWNLOADS: 3246/3250

* TIME TO EXECUTE: 2163.6748 s
```

Step 2 · Annotate videos with dinamic timestamps

Add clock timestamp to nested video files in `folder`

In [12]:

```
timer = Timer()

writer.annot_folder_nested(folder, to_folder, ext, overwrite_annot, report_freq)

timer.end()
```

```
VIDEO TIMESTAMP ANNOTATION · DONE: 13405/13407 · SUCCESS: 3246/13407
ANNOTATE VIDEO TIMESTAMP FAILED. FILE ALREADY EXISTS · FILE: Dados/flood-video-collection-stamped/p
olygons/manual/8/267/CODE267 2023-02-08 15-55-00.mp4
ANNOTATE VIDEO TIMESTAMP FAILED. FILE ALREADY EXISTS · FILE: Dados/flood-video-collection-stamped/p
olygons/manual/8/267/CODE267 2023-02-08 16-20-00.mp4

* TIME TO EXECUTE: 1638.281 s
```

Step 3 · Concatenate and accelerate videos from folders

Concatenate videos by date from nested folders in `base_folder`

In [13]:

```
timer = Timer()

writer_speed.concatenate_videos_from_nested_folders_by_date(
    base_folder, to_base_folder, ext, overwrite_concat, report_freq
)

timer.end()
```

```
CONCAT VIDEOS BY DATE FROM NESTED FOLDERS · DONE: 350/352 · FOLDER: polygons/manual/75/2140
CONCAT VIDEOS BY DATE FROM FOLDER (FAILED) · FILE ALREADY EXISTS · FILE: CODE120 2023-02-08.mp4
CONCAT VIDEOS BY DATE FROM FOLDER (FAILED) · FILE ALREADY EXISTS · FILE: CODE267 2023-02-08.mp4

* TIME TO EXECUTE: 1452.5147 s
```