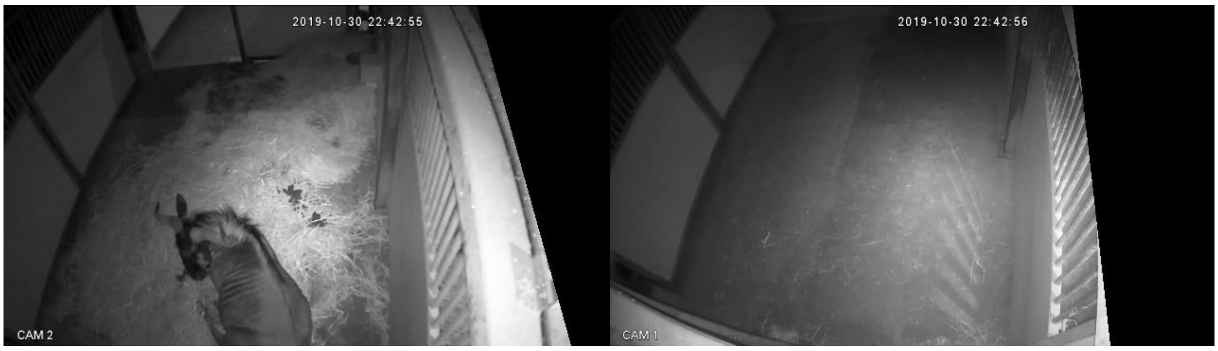
Short description

Samples a given number of images out of video files. Has the feature to "black out" some regions of the video streams that are misleading (as during prediction) and to merge various streams (see merge video files).



Requirements

- packages: opencv, numpy
- data structure described globally

Step 1 – open spyder

- teminal / shell:
  - conda activate bovids
  - spyder

Step 2 – adjust parameters

- BASE_PATH_VIDEO = Path to the folders containing the video streams of a specific species in a specific zoo.
  - E.g.: '.../Wildebeest/FancyZoo/Videos/'
- OUTPUT_FOLDER_BASE = Outout destination (folder) for the single images, inside this folder, the script will create subfolders species/zoo/enclosure_number/
  - E.g.: '.../Desktop/Training-Images/'
- ZOO: zooname [string],
  - e.g. ZOO = „FancyZoo"
- ENCLOSURE_NUMBER: enclosure number as a string.
  - E.g. ENCLOSURE_NUMBER = ‚4'
- SPECIES: species as a string.
  - E.g.: SPECIES
- VIDEO_NUMBERS: List of integers which video streams should be set side by side (see merge video files). The script will use the ordering of the list if no entry is in BOX_PLACEMENT (see below).
- VID_LEN_HOURS: Length of the video files in hours [integer]
- LIST_OF_DATES: List of strings containing all dates out of which images of the given streams should be sampled.
  - E.g: LIST_OF_DATES = ['2020-03-07', '2020-03-16', '2020-03-22', '2020-03-03']
- NUMBER_IMAGES_CREATE = Number of images that are sampled in total. Will be splitted equally across all dates.

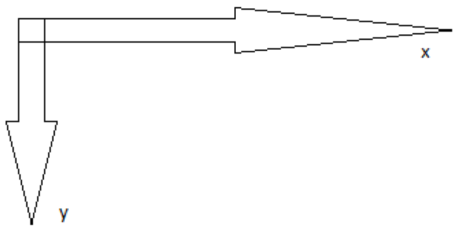
- BOX_PLACEMENT : Dictionary such that the keys are enclosure_codes. Corresponding values are lists of integers. If the enclosure_code of the current operation in contained in BOX_PLACEMENT, the entries of VIDEO_NUMBERS will be permuted accordingly to the relative ordering of BOX_PLACEMENT[enclosure_code].
    - E.g: BOX_PLACEMENT = { ‘Wildebeest_FancyZoo_4’: [6,4] }
    - POLYGON_ENDPOINTS should contain the same information as VIDEO_BLACK_REGIONS out of the global_configuration.py – it is redundant to have this script as a stand-alone version which was helpful during some studies. Detailed instructions and an example can be found below.

Step 3 – run the script

- Processing will start immediately.

Further information: „POLYGON_ENDPOINTS“

- Coordinates of an image will be given as (x, y) coordinates.
- A polygon can consist of arbitrarily many nodes (at least 3) in general position.
- The ordering of coordinates is counterclockwise as usual in math.

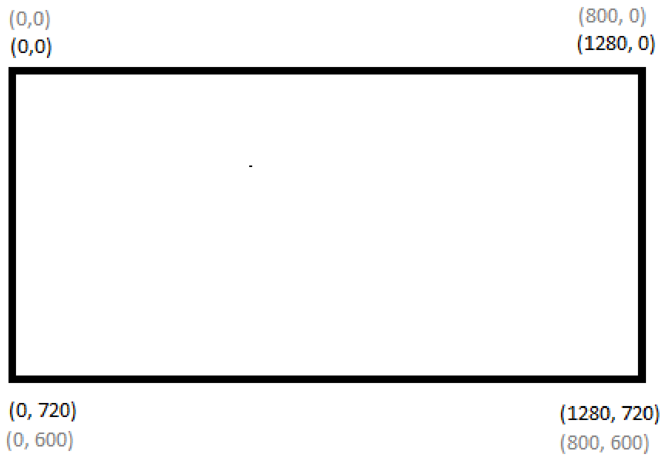


In the following examples we suppose that:

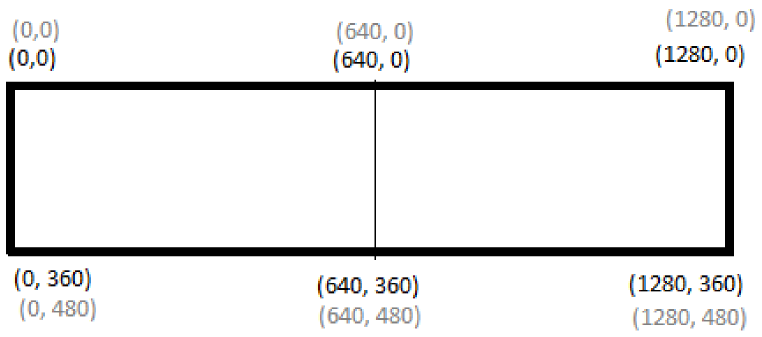
Image material has format 16:9.

Image material has format 4:3.

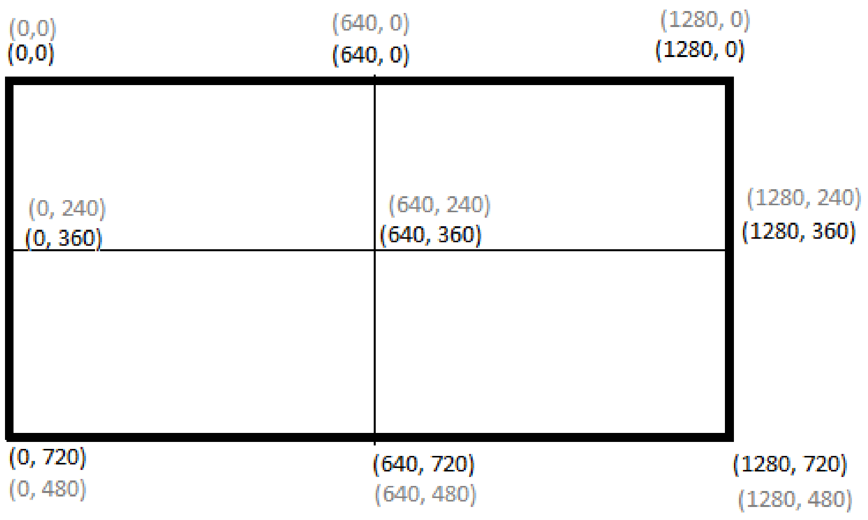
Single stream



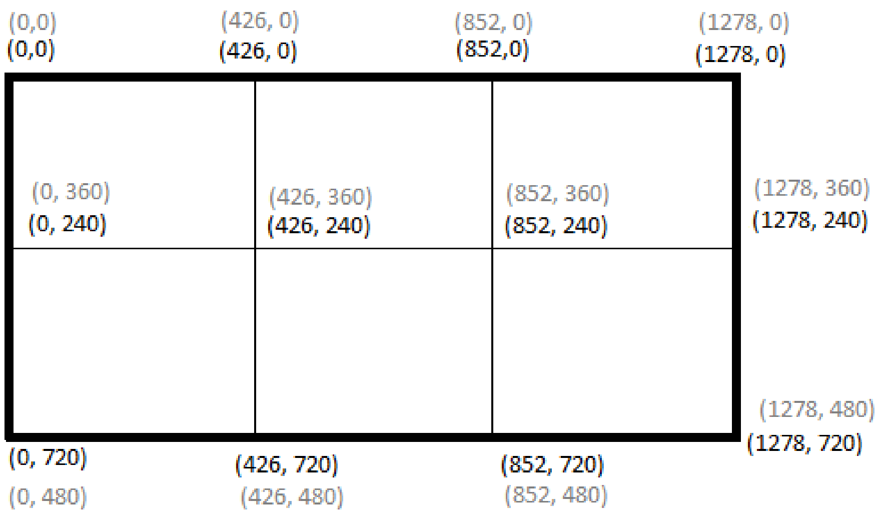
Two streams



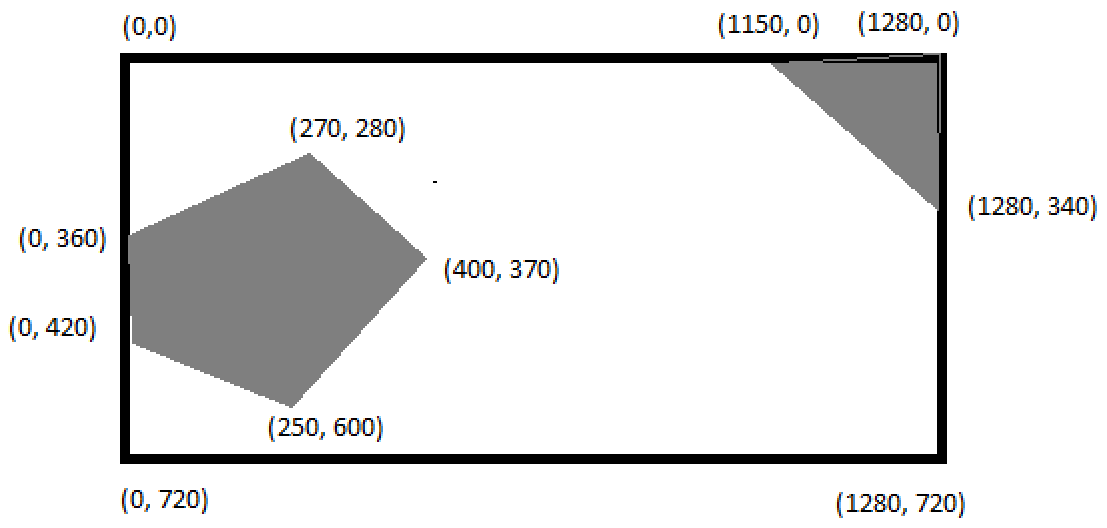
Three or four streams



Five or six streams



Example



```
"Wildebeest_FancyZoo_3":  [
                np.array( [ [0, 360], [0, 420], [250,600], [400,370], [270,280] ]),
                np.array( [ [1150,0], [1280,340], [1280,0] ] )
                ]
```