Short description

Has two functionalities:

- Sample images from given video streams, put the streams together and marks black regions. This part can be used as a (more convenient but not stand-alone) replacement of create_annotation_images.py). Creates a folder containing subfolders for each enclosure.
- Run yolov4 object detection on those images creating .xml files containing bounding box information (readable by the evaluation script during OHEM). Will use the object detection network provided for a given enclosure code by global_configuration.py.

Requirements

- Packages: opencv, numpy, xml, tensorflow
- Additional resources of BOVIDS: global_configuration (BOX_PLACEMENT, VIDEO_BLACK_REGIONS), yolo-library
- For step 1 (sampling images), a comma-separated training-csv file is required. It contains the necessary information which nights (thus, enclosures with given video streams and individuals on a specific date) should be used for sampling. An example is stored in the example folder.

Date	**Species**	**Zoo**	**Enclosure**	**Video**	**Individuals**
18.02.2020	Wildebeest	FancyZoo	1	1	1
31.12.2018	Eland	NiceZoo	2	3;4	2
10.01.2020	Kudu	GreatZoo	3	4	2;6

Step 1 - open spyder:

- Terminal:
  - conda activate bovids
  - spyder

Step 2 – adjust parameters:

- GLOBAL_CONFIGURATION_PATH: folder in which global_configuration.py is stored. [string]
  - E.g.: …/bovids/global/

- YOLO_LIBRARY: folder of yolo library [string]
    - E.g.: …/bovids/global/yolo-v4-tf.keras-master/

Task 1: Create images

- CREATE_IMAGES: (True / False). If True, the task „create images“ is carried out. [boolean]
- CSV_OVERVIEW_FILE = Path to the training csv file (see above) [string]
- BASE_PATH_TO_DATA = Path (folder) which is the starting point of data navigation as explained in readme.md. (“video storage”) I.e., contains a subfolder for each species containing subfolders of the zoos etc. [string]
- IMAGES_OUTPUT_FOLDER = Output folder for task 1 – is **also input folder for task 2**. The script creates subfolders such that the structure becomes (/species/zoo/enclosurenumber/imagename.jpg) [string]
- VIDEO_LEN_SPECIAL = Dictionary with zoonames as keys. Can be used if the video length varies between different zoos. [dictionary]
    - E.g.: {'FancyZoo': 12}
- IMAGES_PER_INDIVIDUAL = Number of images that will be sampled per enclosure [integer]

Task 2: Generate labels automatically

- Input folder is IMAGES_OUTPUT_FOLDER of task 1. The content of the training csv file is ignored, the labels will be created for all images in this folder. I.e., the script decides which object detection network is used by the information from global_configuration.py based on the enclosure code.
- CREATE_YOLO_ANNOTATION: If True, task 2 is conducted. [boolean]
- YOLO_OUTPUT_FOLDER = Outputfolder for the annotation files. Will create a structure as in the input folder (species/zoo/enclosure/) [string]
- YOLO_NETPATH = Folder (path) in which the object detection networks can be found as the global_configuration does only contain the filename. [string]
- MAX_DETECTIONS = Number of (different) individuals that might be detected. [integer]

- MIN_CONFIDENCY = Number between 0 and 1. Bounding boxes with smaller certainty will be dismissed. [float]
- VIDEO_HOURS = 14 Standard length of the video files [integer]
- VIDEO_LEN = VIDEO_HOURS * 3600 (seconds) [integer]

Step 3 – run the script

- Processing will start immediately.