Alarming system for the detection of abandoned luggage

Authors: F. Giusti, M. Mannino, S. Palmucci

INSTALL

The following libraries are needed in order to execute the python scripts for abandoned luggage detection:

- OpenCV [v. 4.5.1]
- PyTorch [v. 1.8.1] (optional: Cuda [v. 11.1])
- Detectron2

  Installation command example:

  ```
  python -m pip install detectron2 -f https://dl.fbaipublicfiles.com/detectron2/wheels/cu111/torch1.8/index.html
  ```

- sklearn [0.24.1]
- argparse [1.1]

All the items in the previous list are links to the documentation of the related library.

PROJECT ORGANIZATION

The project directory contains several python scripts that can be conceptually separated in the following way:

BASE SCRIPTS	MAIN SCRIPTS	RUN SCRIPTS
accumulation_mask.py	alert_boxByDetectron.py	run_alert_boxByDetectron.py
sliding_window.py	alert_BG_BoxByDetectron.py	run_alert_BG_BoxByDetectron.py
background_creator.py	alert_BoxByShape.py	run_alert_BoxByShape.py
stationary_silhouette.py		
stationary_object_detector.py		

In the base scripts group there are all the scripts that contain the base classes used by the scripts in the main scripts group. The run scripts group contains a script for each script in the main scripts group, and through this group it is possible to launch the effective alarming system.

INTRO

It is possible to run the program through the python scripts that start with “run”.

- run_alert_BoxByDetectron.py [“Version 1” in the report]
- run_alert_BG_BoxByDetectron.py [“Version 2” in the report]
- run_alert_BoxByShape.py [“Version 3” in the report]

SYSTEM SETUP

All the “run scripts” use the script alert_configuration.py to set up the internal parameters. To change these parameters open alert_configuration.py and change the parameters you want. Below the list of all the parameters that can be changed, the meaning of each parameter is explained in the report.

```
# file: alert_configuration.py
...
        self.STATIONARY_SECONDS = 30
        self.PEOPLE_STATIONARY_SECONDS = 10
        self.SLIDING_WINDOW_SECONDS = 10
        self.BACK_STEP = 1
        self.SKIPPED_FRAMES = 2
        self.THRESHOLD_ACCUMULATION_MASK = 0.5
        self.PEOPLE_ID = 0
        self.CATEGORIES_ID = [24, 26, 28]   # backpack, handbag, suitcase
        self.COLORS = {24: (65, 14, 240), 26: (33, 196, 65), 28: (236, 67, 239)}
        self.NOISE_SCALE_FACTOR_BAGGAGE_SILHOUETTE = 5e-4
        self.NOISE_SCALE_FACTOR_PEOPLE_SILHOUETTE = 1e-3
        self.NOISE_SCALE_FACTOR_PEOPLE_SILHOUETTE_REDUCED = 7e-4

        self.BACKGROUND_METHOD = 'MOG2'
        self.BACKGROUND_LEARNING_RATE = 0.0007
```

HOW TO RUN

In order to better manipulate input and output video is possible to pass some arguments to the “run scripts”. To show all the possible argument type:

```
$ python3 run_run_alert_BoxByDetectron.py --help
usage: run_alert_BoxByDetectron.py [-h] [--input_file INPUT_FILE]
                                   [--output_dir OUTPUT_DIR]

optional arguments:
  -h, --help            show this help message and exit
  --input_file INPUT_FILE, -i INPUT_FILE
                        Path to the input video file, (default:
                        ./data/video1.mp4)
  --output_dir OUTPUT_DIR, -o OUTPUT_DIR
                        Path to the output folder (default: ./output)
```

It is possible to specify the input file and the output directory.

Below are written the commands to type in order to run all the three “run scripts” where the input file is *./data/video1.mp4* and output directory is *./output.*

```
$ python3 run_alert_BoxByDetectron.py -i ./data/video1.mp4 -o ./output/
$ python3 run_alert_BG_BoxByDetectron.py -i ./data/video1.mp4 -o ./output/
$ python3 run_alert_BoxByShape.py -i ./data/video1.mp4 -o ./output/
```