Analysis Module for Surveillance Cameras

User Manual

Klara Hellgren, klahe156@student.liu.se
Helena Kihlström, helki570@student.liu.se
Axel Nyström, axeny846@student.liu.se
Denise Härnström, denha296@student.liu.se
Hanna Hamrell, hanha664@student.liu.se

October 2018

1 Introduction

This is the user manual for the Analysis module for surveillance cameras.

Section 2 gives an overview of the system and its modules. Section 3 describes the system requirements and section 4 gives a step-by-step explanation on how to use the system.

2 System overview

The Analysis module for surveillance cameras can, given a folder with video files, detect and track persons in all videos. It can also segment the person objects into parts such as coat, head, hair, left hand and pants etcetera. The results are saved as JSON objects. There is also an option of re-identifying a specific person seen in a video frame, both using the feature vectors from the tracking algorithm as well as an algorithm called AlignedReID. Finally, it is possible to visualize the detection, tracking, re-identification and segmentation results.

2.1 The Main Module

The main module is from where detection, tracking and segmentation are run. From here, it is also possible to visualize the detection. Flags determine which parts to run, as well as from where to load and save data.

2.2 The Multivideo Module

The multivideo module is used to compare objects tracked in different cameras. It generates bounding box images for each tracked person and these images are then used to compare people. The user specifies an original id and which camera this person was found in, the multivideo module then compares this person to people in other videos.

2.2.1 Multivideo matching using deep-sort feature vectors

This is an alternative to the main multivideo module, that performs the multivideo matching by re-using the feature vectors from the tracking with deep-sort. This module runs much faster than the main one, but might be less accurate.

2.3 The Visualization Module

The visualization module visualize the tracking result. When running the visualization module a video file will be saved in a folder specified by the user. The length of the sequence can be change by adding a minimum and maximum frame number. If specific ID:s are chosen the module will visualize the tracking result only for these ID:s. If no ID:s are chosen the module will show the tracking result for all objects.

3 System Requirements

All necessary files are downloaded through the gitlab repository, except for one folder used for the segmentation, and all packages necessary are found in the tsbb11 Anaconda environment. Everything is explained in more detail in section 4 How to Use It. It is not necessary to have CUDA, but recommended since it will make the system a great deal faster. The code tested on Windows and Linux.

4 How to Use It

Below are step-by-step instructions you need to follow to get your system up and running.

4.1 Setting up the environment

1. In directory of your choice, clone gitlab repository by typing "git clone repository-name".
2. Install the Anaconda prompt.
3. Use the environment.yml file in in the git repository to recreate our tsbb11 environment as described here: https://conda.io/docs/user-guide/tasks/manage-environments.html. This will create an environment with all libraries and packages necessary to run the code.
4. Activate your environment by writing "activate tsbb11" in the anaconda terminal prompt.
5. To enable use of the segmentation, do the following: In the README.md under the headline Pre-trained models, download the pre-trained PGN model from drive here: https://github.com/Engineering-Course/CIHP_PGN. Create a folder called 'checkpoint' in `\external\cihp_pgn` and put the model folder there.
6. Put your video files in a folder of your choice, preferably outside of the repository.
7. Run your system using some of the arguments listed below.

4.2 Running the main module

Below, all possible input arguments for the main module are explained and examples of how to run it are shown. All arguments are optional except for the video_dir argument.

video_dir Path to directory of videos to be processed.

--detection_output_dir Path to directory of detection output, default is '../ detection_output'.

--detection_output_suffix Suffix of detection output, default is 'txt'.

--min_frame Frame number to start at, default is 0.

--max_frame Frame number to end at, default is -1.

-c OR --classes Classes to detect, default is 'person'.

-v Enables visualization of detections, default is False.

--cfg_file Yolov3 config file to use, default is '`external/pytorch_yolo3/cfg/yolov3.cfg`'.

--weight_file Yolov3 network weights to use, default is '`external/pytorch_yolo3/yolov3`'.weights'.

--names_file List of classes that can be detected, default is '`external/pytorch_yolo3/data/coco.names`'

--tracking_output_dir Path to tracking output directory, default is '`../tracking_output`'.

-pgn To use segmentation using PGN, default is False.

--save_pgn_res Save images resulting from PGN segmentation, default is False.

--pgn_output_dir Path to PGN segmentation output directory, default is '`../segmentation_output`'.

Example of how to run detection, tracking and segmentation using a minimal number of input arguments:

```
python main.py "C:\Users\name\path\to\folder\video_file_folder" -pgn
```

Example of how to run detection, tracking and segmentation in all videos from frame 0 to 42, visualizing detection and saving the tracking results in `../tracking_output` and the segmentation results in `../segmentation_output`. It will work on Windows.

```
python main.py "C:\Users\name\project\video_file_folder" --min_frame=0
max_frame=42 -v --tracking_output_dir="..\tracking_output" -pgn --save_pgn_res
--pgn_output_dir="..\segmentation_output"
```

Please note that all video file names must be unique!

4.3 Running the multivideo module

The multivideo module needs a pre-trained model for AlignedReID. This model can be downloaded from here. Descriptions of the parameters and an example on how to run the multivideo module can be seen below.

```
python -m multivideo -result_dir='C:\Users\name\project\tracking\_output\
data' --origin_dir=HAB-P101 --id=166 --video_dir='C:\Users\name\project\
videos' --model_dir='C:\Users\name\project\models\checkpoint_ep300.pth.
tar' --rm_dir=False
```

This example call will work when Windows is used.

--result_dir Directory where the results from the tracking is saved. This directory is called data and is created when the main tracking is used.

--origin_dir Name of the directory in which the original person you want to find in other videos can be found. This directory should exist inside the directory called 'data'.

--id Id of the person in origin_dir which you want to find in other videos.

--video_dir Directory where all the videos are stored.

--model_dir Directory where the downloaded model is stored.

--rm_dir Removes the directories which are created by the multivideo matching if it is set as True.

4.3.1 Multivideo matching using deep-sort feature vectors

Below, all input arguments are listed. Only the first three arguments are required. Note that the default values of the three last optional arguments are fitted to the output of the main module. If any changes have been made regarding the paths of these files, the new paths will be required as input.

--id ID of desired person to search for in other sequences

--sequence Sequence where desired person appears

--num_candidates Number of candidates to search for in other sequences

--display Flag for whether or not to visualize the result

--output Path to desired output directory (where to save visualization video)

--parameter_file Path to JSON-file with parameters used in tracking

--result_file Path to JSON-file with paths to results from tracking

--tracking_data_dir Path to directory where tracking data is stored

An example of how to run the submodule on Windows using a minimal number of arguments:

```
python multivideo\multivideo_deep_sort.py -id=70 -sequence=HAB-P202
```

An example of how to run the submodule on Windows using a maximal number of arguments:

```
python multivideo\multivideo_deep_sort.py --id=70 -sequence=HAB-P202
--output='.\multivideo_deep_sort_videos' --num_candidates=5 --display=True
--parameter_file='parameters.json' --result_file='results.json'
--tracking_data_dir='..\tracking_output\data'
```

4.4 Running the visualization module

An description of the parameters used and an example of how to run the visualization model is given below. In this example the tracks of objects 1,4 and 7 are shown in the output video.

```
python -m visualization --video='C:\Users\name\project\tracking_output\
data\video_name\video_name_cut.avi' --file_path='C:\Users\name\project\
tracking_output\data\video_name\video_name_track.txt' --output_dir='C:
\Users\name\project\tracking_output\data\video_name\' --output_name=
'my_video' --min_frames=10 - -max_frames=42 - -cut_video='1'
--id='1,4,7' --show_track='1'
```

--video Path to video.

--file_path Path tracking text file.

--output_dir Direction where the output video will be saved.

--output_name Output video name.

--min_frames Frame number to start at.

--max_frames Frame number to end at.

--cut_video If visualizing the result using a cut video, default=False. If a cut video is used the same min_frames and max_frames used for tracking must be added.

--id ID:s of desired objects.

--show_track Visualizing the object track through the sequence.

When running the tracking module all tracking parameters will be saved in a JSON file. This file can be used as an input to the visualization module. When using the parameters file the user do not need to specify video, file_path, min_frames, max_frames and cut_video parameters.

```
python -m visualization --parameters_file='C:\Users\name\project\tracking_
output\data\video_name\video_parameters.json' --output_dir='C:\Users\
name\project\tracking_output\data\video_name\' --output_name='my_video'
--id='1,4,7' --show_track='1'
```

--parameters_file Path to parameters file.