Tracking and Identifying People in Real-Time: Classifying team membership based on team uniform's color

Documentation

8th January 2016

By:

Justinien Bouron

Bachelor, Year 3, Computer Science,

École Polytechnique Fédérale de Lausanne, Switzerland

and

Nicolas Roussel

Bachelor, Year 3, Communication Systems,

École Polytechnique Fédérale de Lausanne, Switzerland

Supervised By:

Dr. Horesh Ben Shitrit

Prof. Pascal Fua

Computer Vision Laboratory,

École Polytechnique Fédérale de Lausanne, Switzerland

Table of Contents

1. Basics

1.1 System requirements

To be able to run this software you will need **Ubuntu 15.04**. Because of some packages missing, Ubuntu 15.10 cannot run this software.

All the packages dependencies are installed using the setup bash script. This script installs all the libs that cannot be statically linked to the executable. They are dependencies coming from the openCV 2.4.12 library (which is statically linked, so you won't need to build it on your machine). We recommend that you check the bash script before running it for precaution measures.

1.2 Files and folders

This piece of software is composed of one executable, a configuration file, the res/ folder containing everything needed to run it, a test/ folder, a code/ folder containing the source code, a samples/ folder, a frames/ folder and finally a setup bash script.

This program has a command line interface which allows the user to specify the inputs to the algorithm such as the source video, the starting frame, the number of threads to use etc, ...

For any change on the program internal parameters (threshold values, …) changing the internal behavior of it, you must use the configuration file. See the section 4 for more informations.

The res/ folder contains all the files needed by the program to be able to run, except the videos.

This folder contains four subfolders :

- bgs_mask : The static masks used for the background subtraction.
- cluster : The cluster centers used to decide the team membership of the players.
- xmls : DPM model file used by the program to detect players and extract their body parts.
- room_background : Images of the empty room, per camera.

The test/ folder contains a bash script used to test the very basic behavior of the software on your machine. If the script returns “Test succeeded”, the setup on your machine is correct. See section 2.1 for the testing part.

Additionally the samples/ folder contains some sample videos along with their results. For every sample, default configuration file, and arguments were used.

The last folder is named frames/ and contains every frames computer during one run of the software, if the save_all_frames field is set in the configuration file (see section 4).

As said earlier, the setup bash script makes sure that all the dependencies are installed on your machine.

Finally, a little script example.sh allows you to see an example of the software just by executing the bash script.

1.2 Notes to run the software

Some constraints need to be respected to be able to run our software :

- The video should be named “ace_i.mp4” where i ∈ {0, 1, 2, 3, 4, 5, 6, 7}.

- It is recommended that the very first frame of the video is empty (ie no players nor object on the field) as the BGS will take the first frame of the video as a background model. If this constraint is not satisfiable, you can assert the use_empty_room_images_as_background field in the configuration file. This field will tell the BGS to get its background model from the /res/room_background/ folder. Beware ! If the conditions (lightning, open/closed window, objects, …) of the room are not similar between the video and the empty room image, false positives can appear, and computation time can increase. You are free to change these images in order to meet the conditions seen in the video.

- This software is supposed to work with the CSS jerseys (red and green). But it can be changed if more teams are needed. However you will need to change

parameters related to the Features Comparator and add team possibilities in the source code.

- In order to run, the executable file must be in the same folder as the res/ folder and the configuration file.

- The path to the video **should not** contain spaces, otherwise our software might not be able to open it.

2. Install the software on your machine and test it

2.1 Installation of dependencies

To install all the dependencies on your machine and thus be able to run our software, you can follow these steps :

1. Open a terminal and cd to the directory containing the executable file (root of the project directory)

2. Execute the setup.sh script :

```
$ sudo bash setup.sh
```

Note : The setup bash script can take some time on the “Processing triggers for libc-bin” step. This is normal just be patient.

If the setup fails, please contact us by email :

nicolas.roussel@epfl.ch
justinien.bouron@epfl.ch

2.2 Test the software

A test script in the test folder makes sure that the software runs correctly on your machine.

This test can take some time depending on your machine, so do not worry, you can expect one minute at least (unless you have a powerful machine).

Make sure you are in the root folder of the project (containing the setup.sh) and follow these steps :

1. To run this test you need the execution right on the executable, type :

   ```
   $ chmod +x ./Bachelor_Project
   ```

2. Then run the test script :

   ```
   $ bash test/test.sh
   ```

If the test fails, please contact us by email :

nicolas.roussel@epfl.ch
justinien.bouron@epfl.ch

2.3 Run the software by yourself

You can now run the software on your own. The example.sh bash script is a very simple example that show you the result on a short video. Execute this bash script to the the software run and its results.

You will find the resulting video named **result.avi** in the root folder.

You can also run the software with the parameters you want, using the command line interface and the configuration file.

3. Command line interface

The command line interface of this program has the following form :

./Bachelor_Project VIDEO_DIRECTORY CAMERA_INDEX [OPTIONS]...

3.1 Required parameters

The first mandatory parameter (VIDEO_DIRECTORY) is the path to the folder containing the video. In this folder, every video must have the name ace_i.mp4 where i ∈{0, 1, 2, 3, 4, 5, 6, 7}.

The second mandatory parameter (CAMERA_INDEX) is the index of the camera/video to work on.

3.2 Optional parameters

A set of optional parameters is defined, you can use them in any order.

-s start	Set the starting frame index (default 0).
-e end	Set the ending frame index (default last frame).
-j size	Set the step size between two consecutive frames (default 1).
-t count	Set the number of threads to use during execution (default 1).
-b size	Set the step size between two box refreshes (default 1).
--show	Show the results in real time as they are computed.
--train	Train the software to create new templates for each team.

Remarks :

When -b is greater than 1, the -j parameter is ignored and the step size between the frames is taken to be 1, however the players won't be computed for each frames but rather every x frames (with x the value assigned to -b). It allows real time computing when parameters are correctly set given the speed of the hardware.

The --train option will create a new "cluster.kfc" file by using every frame on which players are extracted (depends on the -s, -e, -j, -b parameters). This should be used if the jersey colors are changed from red and green or when experimenting with different amounts of training.

4. Configuration file

4.1 Basics

This program comes with a configuration file, containing all the parameters taken by the algorithm and affecting its internal behavior.

As the command line interface allows you to change the basics features of the program (inputs, results, …), the configuration file allows you to change the internal behavior of the program.

4.2 Parameters by type

4.2.1 Miscellaneous

draw_static_boxes : Boolean, draw static boxes (all of the same size).
static_boxes_width : Int, width of the static boxes.
static_boxes_height : Int, height of the static boxes.
static_boxes_thickness : Int, thickness of the static boxes.

4.2.2 Results

save_results : Boolean, save the resulting video in result.avi.
show_torsos : Boolean, show the torsos on the frames.
show_body_parts : Boolean, show the body on the frames.
show_players : Boolean, show the players DPM boxes on the frames.
show_blobs : Boolean, show the blobs on the frames.
show_player_team : Boolean, show the colored box for each players on the frame.
save_all_frames : Boolean, save all the computed frames into the frames/ folder.

4.2.3 BGS, Background Subtraction

bgs_detect_shadows : Boolean, specify if the BGS should detect the shadows or not.
bgs_threshold : Float, the threshold used by the BGS.
bgs_history : Int, history of the BGS.
bgs_learning_rate : Float, learning rate of the BGS.
bgs_blob_buffer_size : Int, size of the window when performing blob detection.
bgs_blob_threshold_count : Int, number of white neighbors to make a pixel white.
use_empty_room_images_as_background : Boolean, use empty room images as a background model.
use_bgs : Boolean, perform background extraction or not.

4.2.4 DPM

dpm_detector_numthread : Int, number of threads to use inside DPM.
dpm_extractor_score_threshold : Float, Score threshold of DPM.
dpm_extractor_overlapping_threshold : Float, Overlapping threshold of DPM.
use_dpm_player_extractor : Boolean, use DPM on the whole frame, not on the blobs.
use_colored_mask_in_dpm : Boolean, use the colored mask when performing DPM.

4.2.5 Features Comparator

features_comparator_correlation_threshold : Float, lower threshold for the correlation between a player and a center, if the correlation is lower than the threshold then the player is put in the Unknown team.

4.2.6 Blob player extractor

blob_player_extractor_buffer_size : Int, buffer size when extracting the blobs from the frame.
blob_player_extractor_min_blob_size : Int, minimum size of a blob, if the size is lower the blob is discarded.

5. Libraries used

For this piece of software the following libraries were used :

- OpenCV 2.4.12 : The most used one, allow us to perform BGS, DPM, ...

- SDL 2.0 : Allows the user to show the results as they are computed. We couldn't display images with openCV because of a bug : If you try to perform DPM after displaying an image, you will end up having a SEGV in the middle of the DPM code. As no fix was found, we decided to use SDL to display images.

- libconfig 1.5 : Very simple and lightweight library to handle configuration files.

From all these libraries only SDL was not linked statically and thus added to the dependencies.