Sign Finder – High Level Description of Detection API

YAML Configuration File

The detector parameters are set up using the provided YAML files (one for each sign at this stage), which currently include `exit_sign_config.yaml` and `restroom_sign_config.yaml` (currently for both men and women bathroom signs, but in the future we will be able to distinguish between the men's and women's).

At the moment there shouldn't be any need for you to change these files; in the future when the detection algorithms have been finalized, we will document the parameters in these files and the likely ways you may want to change them.

Classifier Model Files

The detection algorithm uses two stages of classification. The first stage uses an AdaBoost cascade classifier trained on LBP features. The output of the first stage (i.e. candidate image patches) is fed to the second stage, which is an SVM classifier that assigns a confidence to each candidate patch from 0 to 1 (0 means very unlikely to be the target sign, 1 means very likely to be the target sign).

A model file is associated with each classifier and sign. For instance, in the case of the exit signs the AdaBoost classifier is defined by the file `exit_sign_cascade.xml` while the SVM is defined by the file `exit_sign_model.svm`. The XML files that describe the AdaBoost cascade are in a standard format used by OpenCV. The SVM model files are plain text files generated by libSVM during the training.

These files are generated during the training phase and should not be edited. In the future, once the algorithms have been finalized we will document the training process so that you can use your own training data and generate your own classifier model files or improve the current ones.

Invoking the Sign Detection Code

The detection algorithm is written in C++ 11 and it relies on OpenCV 2.4.11 for image data structures and image processing and on libSVM 3.20 for the SVM classifier.

The file `SignFinder/src/main.cpp` shows how to use the detection algorithm.

The detector is initialized passing the string containing the full path to the configuration file to the constructor. The constructor will take care of initializing the classifiers and the detector itself, in preparation for processing video frames or images in the OpenCV format (cv::Mat). The detection method is invoked on line 227,

```
auto result = detector.detect(frame, fps);
```

where `result` is a vector of detections in the frame. Each detection is represented as a bounding box (`cv::Rect`) specified by the top left corner and its height and width, as well as a confidence value in our detection.

Running the SignFinder application

SignFinder is a console application and it takes a minimum of two parameters in input: the full path to a configuration file and the full path to a video. To launch it, open a shell interface in the folder where the binary is and use the following syntax to run the detector on a video file:

SignFinder -c *path_to_config_file video_file*

To launch the detector using a webcam, use the following:

SignFinder -c *path_to_config_file* 0|1 (0 to use the integrated webcam, 1 to use a camera connected via USB if there is one).

A complete list of input arguments for the detector can be found by

SignFinder –h
and are also listed in the README.md file that is in the main project folder.

List of parameters, default parameters are in braces:

-c, --configFile	location of config file
-d, --debug=[false]	whether to show intermediate detection stage results.
-f, --flip=[false]	whether to flip the input image
-m, --maxdim=[640]	maximum dimension of the image to use while processing
-n, --notrack=[false]	whether to turn off tracking
-o, --output	if a name is specified, output is saved to this video file
-p, --patchPrefix	prefix for saving detected image patches to disk if one is not provided, patches are not saved
-s, --saveFrames=[false]	whether to save frames where detections have occurred
-t, --transpose=[false]	whether to transpose the input image

The –t and –f flags should be used to correct the orientation of the video if the video is displayed in the wrong format (portrait vs landscape).

While the software is running, as the video plays, a window will show the detection results over each single frame in the video sequence in the form of one or more rectangles that highlight patches of the image in which the detection of an exit sign has occurred. The patches that pass the first stage but fail the second are shown in red, confirmed detections instead are highlighted in green. When tracking is on (default) yellow signs show initial detections that need to be confirmed before being declared to the user (i.e., detections that have successfully passed both stages of the detector, but have not yet been confirmed by subsequent detections in following frames).

Press Esc to halt the application.