Event Detection subtask 2 (OS): specification

DATA

The test data consists of mono recordings (WAV, 44.1 kHz) of sequences containing artificially concatenating overlapping acoustic events in an office environment. Original recordings of isolated acoustic events were made using a Soundfield microphone system, model SPS422B. The dataset contains various SNRs of events over background noise (+6, 0, and -6 dB) and different levels of "density" of events (low, medium, and high).

The distribution of events in the scene is random, following high-level directives that specify the desired density of events. The average SNR of events over the background noise is also specified upon synthesis and, unlike in the natural scenes, is the same for all event types. The synthesized scenes are mixed down to mono in order to avoid having spatialization inconsistencies between successive occurrences of a same event.

The test dataset contains events from 16 different classes, which are as follows:

alert (short alert (beep) sound)
clearthroat (clearing throat)
cough
doorslam (door slam)
drawer
keyboard (keyboard clicks)
keys (keys put on table)
knock (door knock)
laughter
mouse (mouse click)
pageturn (page turning)
pendrop (pen, pencil, or marker touching table surfaces)
phone
printer
speech
switch

Submitted event detection systems can be tuned and trained using the publicly released training and development datasets.

EVALUATION

Participating algorithms will be evaluated using frame-based, event-based, and class-wise event-based metrics.

The computed metrics will consist of the AEER, precision, recall, and F-measure for the frame-based, event-based, and class-wise event-based evaluations. For the event-based evaluations, both onset-based and onset-offset-based metrics will be computed.

In addition, computation times of each participating algorithm will be measured.

SUBMISSION FORMAT

Command line calling format

Executables must accept command-line parameters which specify:

- A path to an input .wav file.
- A path to an output .txt file.

For example:

```
>./eventdetection /path/to/input.wav /path/to/output.txt
```

If parameters need to be set for the program, this can be done, provided the manner in which the parameters is set are well documented by the submitter. So if, for example, your program needs a specified frame rate, set by a -fr flag, an example calling format could be of the form:

```
>./eventdetection -fr 1024 /path/to/input.wav /path/to/output.txt
```

where the manner by which, as well as the desired parameters to be used are specified upon submission and in the corresponding readme file bundled with the submission of the algorithm.

Programs can use their working directory if they need to keep temporary cache files or internal debugging info.

Output file

The output ASCII file should contain the onset, offset and the event ID separated by a tab, ordered in terms of onset times (onset/offset times in sec):

```
<onset1>\t<offset1>\t<EventID1>
<onset2>\t<offset2>\t<EventID2>
...
```

E.g.

```
1.387392290  3.262403627  pageturn
5.073560090  5.793378684  knock
...
```

There should be no additional tab characters anywhere, and there should be no whitespace added after the label, just the newline.

PACKAGING SUBMISSIONS

For Python/R/C/C++/etc submissions, please ensure that the submission can run on the Linux disk image we provide, WITHOUT any additional configuration. You may have modified the virtual machine after downloading it, but we will not be using your modified disk image - we will be

running your submission on the standard disk image. This means:
- if you have used additional Python/R script libraries, they must be included in your submission bundle, and your script should be able to use them without installing them systemwide.
- if you have used any additional C/C++ libraries, they must be statically-linked to your executable.

For Matlab submissions, ensure that the submission can run with the toolboxes and system that the organisers have specified. If you need any particular toolboxes or configuration please contact the organisers as soon as you can. Please aim to make MATLAB submissions compatible across multiple OS (usual problems exist in the file/path separators). All Matlab submissions should be written in the form of a function, e.g. eventdetection(input,output); which can allow calling the script from the command line very easily.

Please provide some console output, which can provide a sanity check to the challenge team when running the code. This can be of the form of simply writing out a line corresponding to different stages of your algorithm

All submissions should include a README file including the following the information:

- Command line calling format for all executables including examples
- Number of threads/cores used or whether this should be specified on the command line
- Expected memory footprint
- Expected runtime
- Approximately how much scratch disk space will the submission need to store any feature/cache files?
- Any special notice regarding to running your algorithm

Note that the information that you place in the README file is extremely important in ensuring that your submission is evaluated properly.

Time and hardware limits

Due to the potentially high resource requirements across all participants, hard limits on the runtime of submissions will be imposed. A hard limit of 48 hours will be imposed for each submission.