AMI Tracking Evaluator 1.0

Technische Universität München

Developed at BUT FIT,

Department of Computer Graphic and Multimedia

by Vítězslav Beran, 2 January 2006

Content

Introduction

According to published paper proposing the method for multi-object tracking evaluation [Schreiber05], we have implemented the evaluator and tested it on AMI data. First part of this document covers brief overview of proposed methods, defines configuration and identification errors with examples.

[Schreiber05] Schreiber, S., Gatica-Perez, D., Potúček, I., Thean, A., Wrigley, S. N.: *AMI WP4 Tracking: Evaluation scheme (Draft)*, 2005.

Second part describes the application, used data formats and graphical user interface, etc.

Multi-object tracking evaluation

Fundamental concept for evaluation the performance of the different tracking algorithms is introduced. The quality of tracking result for a single object is based on shape shape-independent measures.

In following sections, the labeled tracking targets are denoted as ground truth objects *GT*, tracker outputs are referred to as estimates *E*. The output of a tracking approach is considered to be correct, if and only of one *GT* (resp. *E*) is tracking exactly one *E* (resp. *GT*).

Coverage test

Two measurements, recall represents the ration of the ground truth *GT* area, which is covered by the estimate *E*, and precision represents the ration of the estimate *E* area, which is covered by the ground truth *GT*:

$$\alpha_{i,j} = \frac{\left|E_i \cap GT_j\right|}{\left|GT_j\right|}; \qquad \beta_{i,j} = \frac{\left|E_i \cap GT_j\right|}{\left|E_i\right|} \qquad \textit{Eq. 1}$$

where $\alpha_{i,j}$ is the recall and $\beta_{i,j}$ is precision. Returning high value only when both recall and precision are high, F-measure is used:

$$F_{i,j} = \frac{2\alpha_{i,j}\beta_{i,j}}{\alpha_{i,j} + \beta_{i,j}} = \frac{2\left|E_i \cap GT_j\right|}{\left|E_i\right| + \left|GT_j\right|}. \qquad \textit{Eq. 2}$$

Configuration errors

In this context, configuration means the number, the location and the size of all objects in a frame of the scenario. To identify all types of errors, 5 configuration measures are introduced.

- False positive (*FP*) – there is an *E* indicating object, where no *GT* is.
- False negative (*FN*) – *GT* is not tracked by an *E*.
- Multiple trackers (*MT*) – more than one *E* is associated with only one *GT*.
- Multiple objects (*MO*) – more than one *GT* is associated with only one *E*.
- Configuration distance (*CD*) – normalized difference between amount of *E* and *GT* objects. Relevant only together with other errors.

Occlusion handling

The occlusion flag is defined, enlarging *GT* objects, if the ratio of *GTj* area, which is covered by *GTk* exceeds certain threshold t_O:

$$occ_j = \begin{cases} 1, & \exists GT_k : \dfrac{\left|GT_j \cap GT_k\right|}{\left|GT_j\right| + \left|GT_k\right|} > t_O \\ 0, & \text{otherwise} \end{cases}. \qquad \textit{Eq. 3}$$

In situation when occlusion flag is set, there is no evaluation of any error.

Evaluation procedure

The F-measure is used to evaluate all 4 types of errors and create the configuration map table with respect to occlusion flag of particular *GT* object.

$$FP = \sum_i \mathbf{I}\left[\sum_j \mathbf{I}\left(F_{i,j} > t_I\right) = 0\right] \quad FN = \sum_j \mathbf{I}\left[\sum_i \mathbf{I}\left(F_{i,j} > t_I\right) = 0\right] \qquad \textit{Eq. 4}$$

$$MT = \sum_i \mathbf{I}\left[\sum_j \mathbf{I}\left(F_{i,j} > t_I\right) > 1\right] \quad MO = \sum_j \mathbf{I}\left[\sum_i \mathbf{I}\left(F_{i,j} > t_I\right) > 1\right] \qquad \textit{Eq. 5}$$

$$CD = \frac{N_E - N_{GT}}{\max\left(N_{GT}, 1\right)} \qquad \textit{Eq. 6}$$

where t_I is the threshold of F-measure, when *E* maps *GT*, *GT* maps *E* resp., **I** function returns 1 when the expression is true, 0 otherwise, and i, resp. j is an index of E, resp. GT objects.

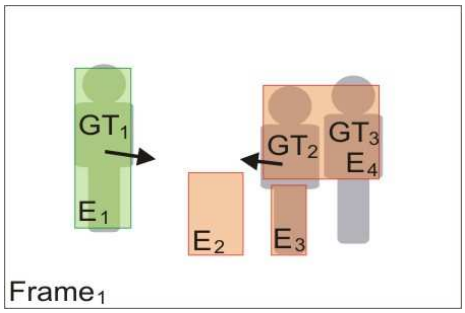


Figure 1. Frame configuration example.

		GT_j			$\sum_j \mathbf{I}(F_{i,j} > t_I)$
		1	2	3	
E_i	1	0,89	0,00	0,00	1
	2	0,00	0,00	0,00	0
	3	0,00	0,51	0,00	1
	4	0,00	0,47	0,47	2
$\sum_i \mathbf{I}(F_{i,j} > t_I)$		1	2	2	

Table 1. F-measure table with error evaluation.

E	*GT*	*occ*
1	1	0
3	2	0
4	2	0
4	3	0

Table 2. Configuration map.

Error type	value
FP	1
FN	0
MT	1
MO	1
CD	0,33

Table 3. Evaluated configuration errors.

For an easy comparison of tracking algorithm errors, are normalized over entire sequence using amount of *GT* objects in each:

$$\overline{X} = \frac{X}{n} \sum_{t=0}^{n} \frac{1}{\max\left(N_{GT}^{t}, 1\right)} \qquad \textit{Eq. 7}$$

where $X \in \{FP, FN, MO, MT\}$.

Identification errors

Identification means that particular *E* tracks exactly one *GT* over its entire lifetime (correctly identifies this *GT* object). 'Majority rule' was used to represents identification association. Two errors are defined and the degree of consistency.

- Falsely identified tracker (*FIT*) – *GT* is mapped by different *E* than frame before.
- Falsely identified object (*FIO*) – *GT* is mapped but frame before was not mapped.
- Object purity (*OP*) – ratio between amounts of frames when GT was correctly identified to the overall amount of frames.
- Track purity (*TP*) – same as *OP* but for *E*. Not interesting for our purpose.

Evaluation procedure

The identification map is constructed during configuration errors are evaluated for each frame in the sequence. Configuration maps of each pair of frames serve to evaluate identification errors.

$$\overline{FIT} = \frac{1}{n}\sum_{t=1}^{n}\frac{\mathbf{I}\left(GT_j^t \rightarrow E_i^t \wedge GT_j^{t-1} \rightarrow E_k^{t-1}\right)}{\max\left(N_{GT}^t,1\right)} \qquad \textit{Eq. 8}$$

$$\overline{FIO} = \frac{1}{n}\sum_{t=1}^{n}\frac{\mathbf{I}\left(GT_j^t \rightarrow E_i^t \wedge !\exists i : GT_j^{t-1} \rightarrow E_i^t\right)}{\max\left(N_{GT}^t,1\right)} \qquad \textit{Eq. 9}$$

$$\overline{OP} = \frac{1}{j}\sum_{j}\frac{\max_i\left(IM_{i,j}\right)}{\sum_i IM_{i,j}} \qquad \textit{Eq. 10}$$

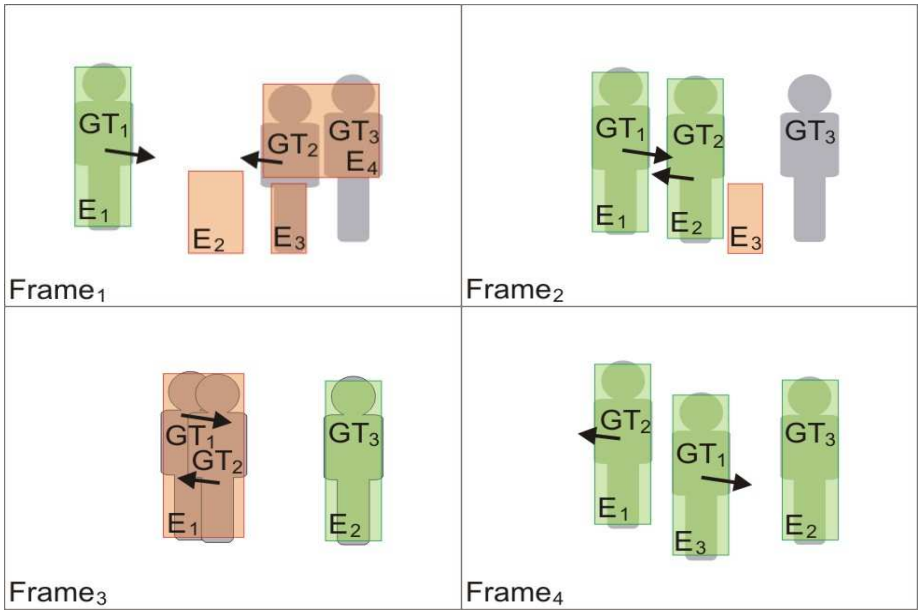


Figure 2. Example of sequence with GT and E objects.

Identification map for example sequence shows amounts of *GT* and *E* mapping and evaluation purity of *GT*, resp. *E*.

$IM_{i,j} = \sum_{t}^{n} \mathbf{I}(E_i \rightarrow GT_j)$		GT_j		
		1	2	3
E_i	1	2	1	1
	2	1	1	0
	3	0	1	1
	4	0	1	2

Table 4. Identification map.

Final error values for example sequence.

Configuration errors		Identification errors	
$\overline{FP}$	0,17	$\overline{FIT}$	0,17
$\overline{FN}$	0,08	$\overline{FIO}$	0,08
$\overline{MO}$	0,08	$\overline{OP}$	0,53
$\overline{MT}$	0,08	$\overline{TP}$	0,58
$\overline{CD}$	0,17	$\overline{ME}$	x

Table 5. Errors of example sequence.

AMI Tracking Evaluator

The Evaluator is a Windows application for evaluation of tracked video data. Two data sets are necessary at least. One, the annotated data referred to as the Ground Truth (GT) objects, and second, the output of an image-based tracking system referred to as the Estimations (E). An output of the evaluation is error values (defined and described in previous document part). The configuration errors can be evaluated for particular frame and the identification errors for entire sequence. Error values are displayed on interactive graph, which allows finding trouble frames effectively.

For scripting purposes, there exists also console application described in the final part of the document.

User manual

The most important and controlling data set is the Ground Truth one. According to this set, the numbers and amount of frames in the sequence are initialized and also consequent browsing is allowed only through frames occurring in GT set.

Figure 3 shows the application GUI. Data in GT or E sets are managed using load, reset or save functions provided by menu or buttons.

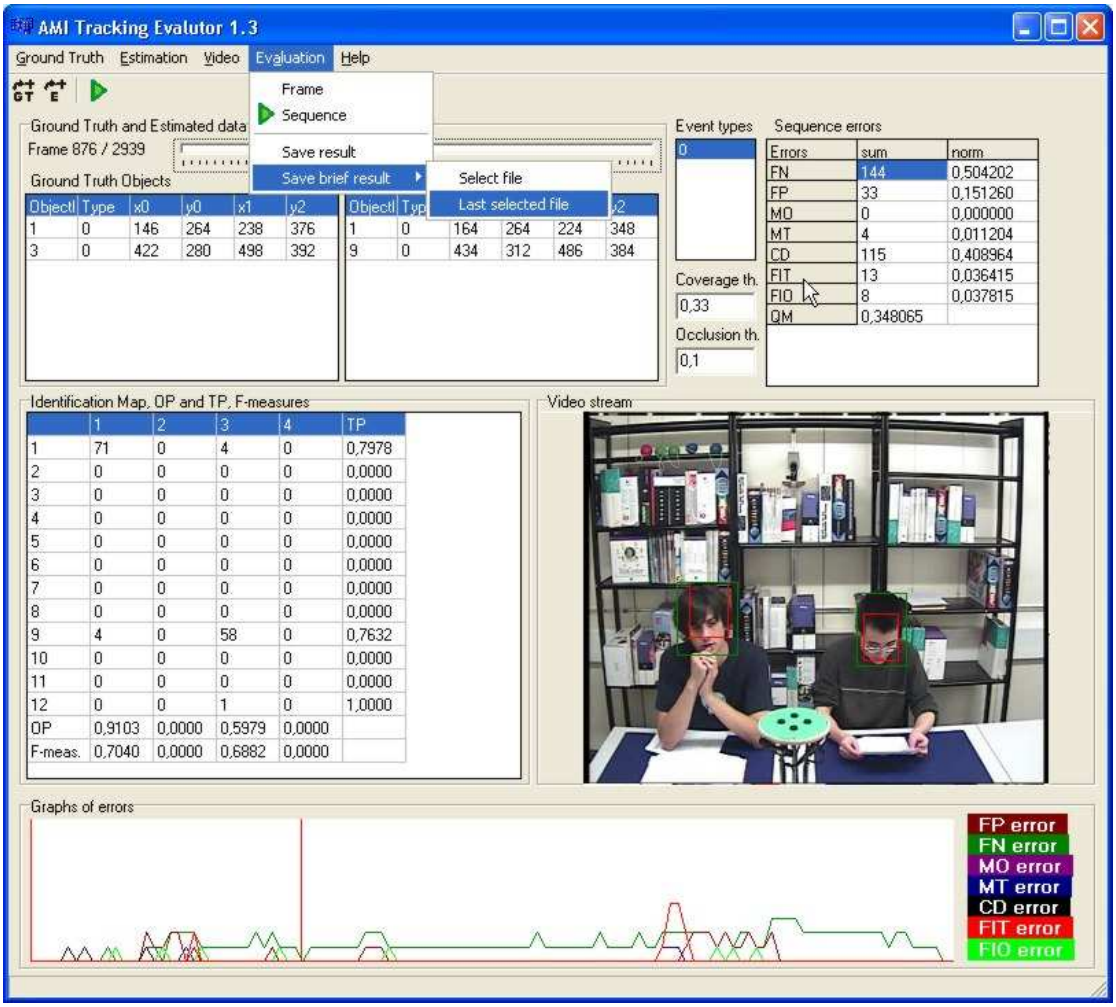


Figure 3 AMI Tracking Evaluator GUI.

- *Load to set* - shows ‘Open file’ dialog and **add** new events from opened file into existed set.
- *Reset* - **clear** the set.
- *Save as* - **store** data set into file selected in ‘Save as’ dialog and in XML format (not supported yet) or TXT format 1 (described later)

The XML event source contains information about event type (if the tracked object is the hand, head, face, etc.). Such information is not provided by TXT source format, so it must be set manually (see Figure 4).

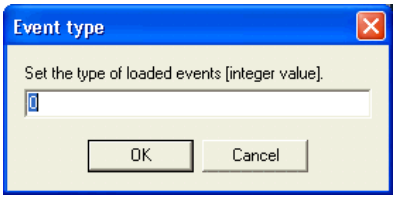
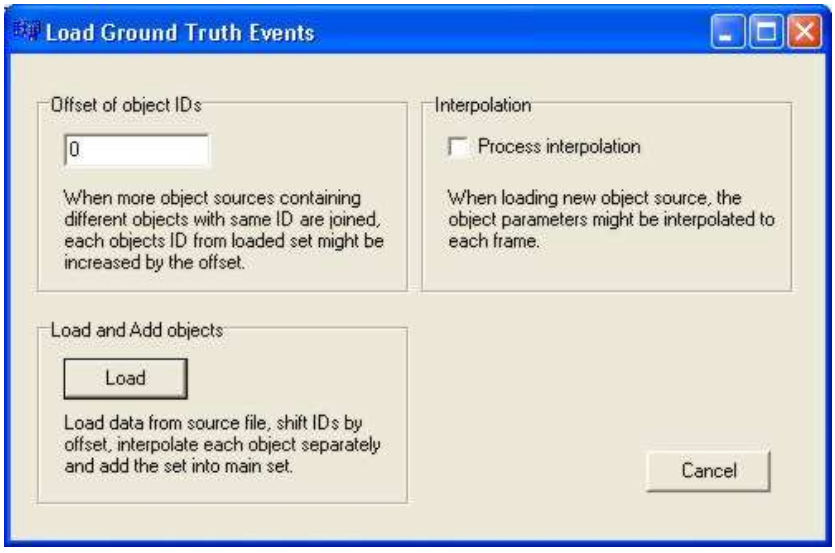


Figure 4 Dialog for default event type setting.

When loading events from the source file, it is possible to re-index event's IDs and also process the interpolation of event object parameters. For particular sequence, such functionality provides possibility to compose several annotations of different people which are annotated in different frames.



Browsing through entire sequence is provided by tracking bar with **Frame** numbers. The frame amount and frame numbers is taken from GT set, not from E set! All events of particular frame are displayed in two tables (**GT Objects** and **E Objects**) where *Type* is an event type and *x0*, *y0* and *x1*, *y1* are the object area corners. The frame evaluation is processed automatically when browsing over sequence and results of **F-measurement** and **Configuration errors** are also displayed in particular tables.

Due to several types of possible events in some data sets, the event types that are processed can be selected in ‘**Event types**’ table. **Configuration Map** shows mapping of GT and E objects with flag of occlusion.

The error measurement is influenced by two thresholds – **Coverage** (how much must be GT and E objects overlapped to be mapped together) and **Occlusion** (how much must be two GT objects overlapped to be flagged as occluded).

When entire sequence evaluation is processed, the **Identification Map** is displayed instead of F-measurement table and shows, how many times where GT and E objects mapped together over entire sequence.

Graphs of errors show the error values courses over entire sequence. Each graph can be turned off, so the observation of particular error graph is transparent.

Data formats

The Evaluator is able to import events from text and XML file and also to export data.

The XML files must follow format defined for Event Editor application:

```
<!ELEMENT AVEvents (EventGroups?, EventTypes?, File?, Editor?)>

<!ELEMENT EventGroups (Group+)>
<!ELEMENT EventTypes (Type+)>

<!ELEMENT Group (ID, Name, Meaning?, Enabled?)>
<!ELEMENT ID (#PCDATA)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Meaning (#PCDATA)>
<!ELEMENT Enabled (#PCDATA)>

<!ELEMENT Type (ID, Name, Key?, Group?, GroupIndex?, Offset?,
Parameters?, Secondary*)>
<!ELEMENT Key (#PCDATA)>
<!ELEMENT Group (#PCDATA)>
<!ELEMENT GroupIndex (#PCDATA)>
<!ELEMENT Offset (#PCDATA)>
<!ELEMENT Parameters EMPTY>

<!ELEMENT Secondary (Key, Offset?, Parameters?)>

<!ELEMENT File (Source*, TimeFormat?, Event*, Title*)>
<!ELEMENT Source (#PCDATA)>
<!ELEMENT TimeFormat (#PCDATA)>

<!ELEMENT Event (ID, Time, Text?, Parameters?)>
<!ELEMENT Time (#PCDATA)>
<!ELEMENT Text (#PCDATA)>

<!ELEMENT Title (Time, Text)>

<!ELEMENT Editor ANY>
```

The TXT files can be one of followed formats: **format 1**

```
frame    frameID
object   objectID   BoxCenterX   BoxCenterY   Width/2   Height/2
```

where all spaces are tabs, or **format 2**

```
image[frameID].* objectID minXPos minYPos maxXPos maxYPos
```

or **format 3**

```
frameID objectID visibility minXPos minYPos maxXPos maxYPos
```

The results of evaluation might be reported in two ways - full and brief. The full version reports all types of errors, tables with F-measure values for each E and GT event combination, errors for each frame, etc. The brief report print out only main errors in one line using CSV format:

```
Sequence; F-Measure; FN; FP; MT; MO; CD;
  FNbar; FPbar; MTbar; MObar; CDbar;
  FIT; FIO; FITbar; FIObar; TPbar; OPbar
```

The brief output is appended to file so the final overview of evaluation in table is easy to make.

The Evaluator provides also help containing this document.

Implementation

The evaluation structures and functions were implemented using C language (see Figure 5).

In the schema, there are not displayed the functions for allocation and de-allocation of the structures. Each function has also one parameter for more, the pointer to particular structure, which is not stated in the UML diagram to make the diagram small.

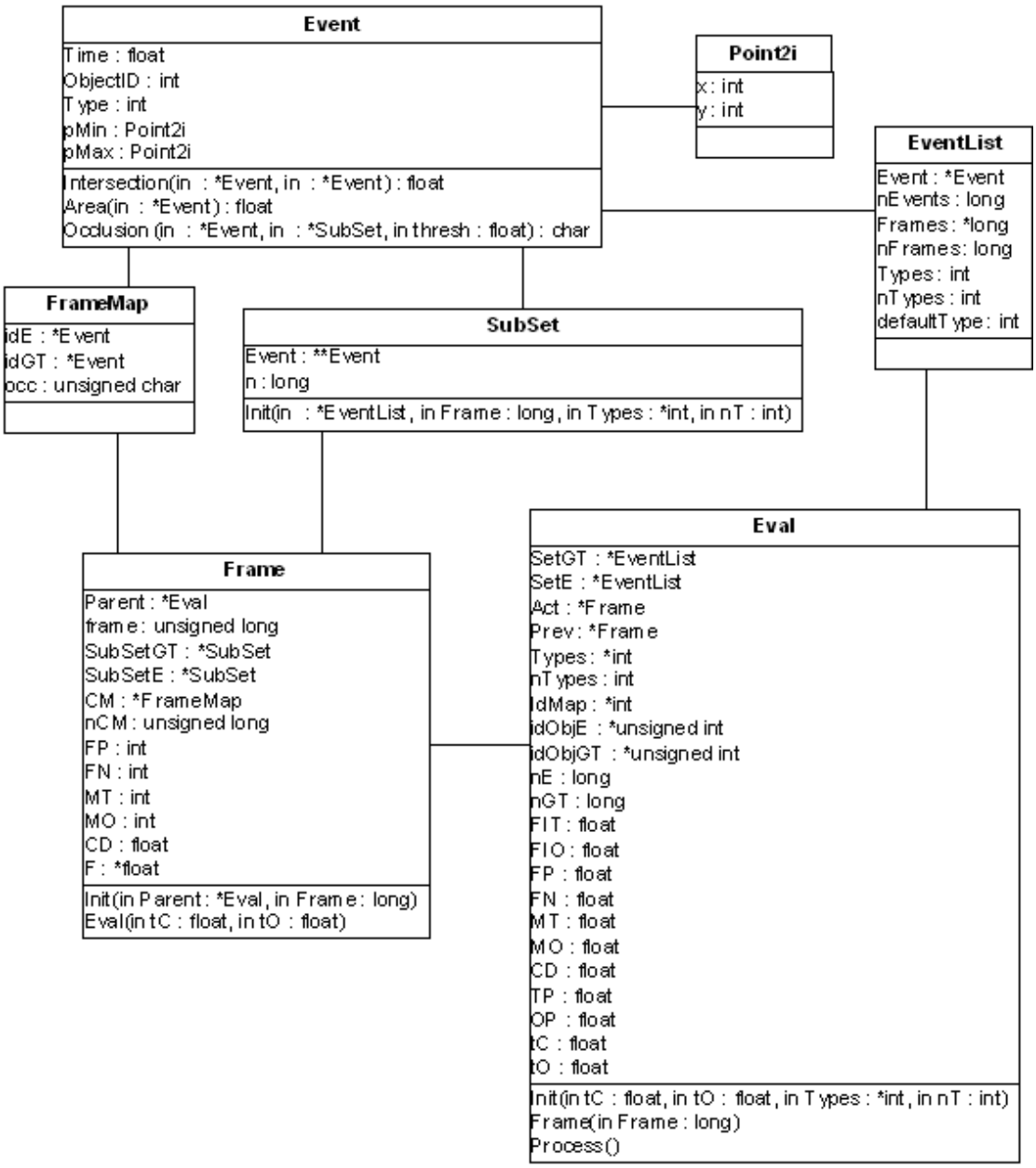


Figure 5 UML diagram of structures and functions for tracking evaluation.

The GUI was designed and generated using Borland Builder C++.

Console Evaluator

The functions for tracking evaluation were used also for console version of the evaluator. The output of the Console Evaluator is stored to text file described later.

The Console Evaluator needs several parameters and all are necessary for proper algorithm's run. The program usage is as follows:

```
Ami_Evaluator_Console.exe [params]
   -gtf [filename]  -  name of the file with Ground Truth objects,
   -ef  [filename]  -  name of the file with Estimate objects,
   -out [filename]  -  name of the output file,
   -gtt [integer]   -  default Event Type for Ground Truth objects
                       when loaded from TXT source,
```

```
-et  [integer]   -  default Event Type for Estimate objects when
                    loaded from TXT source,
-tc  [float]     -  coverage threshold (0.33),
-to  [float]     -  occlusion threshold (0.8),
-b               -  brief output; will be appended at the end of
                    output file,
-nt  [integer]   -  amount of inserted Types,
     [integer]   -  allowed Types, amount of inserted integers
                    must be same as set 'nt' params.
```

Example (data are in the same directory as the evaluator):

```
Ami_Evaluator_Console_10 -gtf GT.txt -ef E.txt -out
evaluation.txt -gtt 1 -et 1 -tc 0.4 -to 0.8 -nt 2 1 2
```

The format of store results contains more information than the application. Besides final Identification errors, there are also stored the Configuration errors for each evaluated frame in CSV format. The evaluation parameters are also stored in the output file.

Tool comparison

There exist two evaluation tools, one designed in IDIAP by Mr Smith, and the other developed by us. We tried to compare the results of these two tools for multi-object tracking evaluation. Unfortunately, we met problems with data format. The data format is very strict for IDIAP evaluator and even if we follow the definitions, we were not successful. The tool seems to have troubles if the GT and E source data contains different amount of frames. The IDIAP annotations contain each 25^{th} frame. Our tool is able to load, convert and evaluate different source formats.