A categorisation of performance errors in continuous context recognition

Jamie A. Ward[1], Paul Lukowicz[2], Gerhard Tröster[3]

Abstract. This paper addresses the shortcomings of existing methods for continuous context and activity recognition systems. It presents an evaluation methodology that (1) provides an objective, non ambiguous way to score event recognition, (2) includes event merges and fragmentations in the error summary, and (3) accounts for timing error as a separate category.

1 Motivation

As more and more groups look at the problem of continuous context recognition and the field as a whole becomes more mature, the question of systematic, reliable comparison of the performance of the different approaches comes into play.

While working on different recognition problems, e.g. [2, 1] and looking at related publications, we have frequently noticed that existing methods such as frame by frame confusion matrices, accuracy, or standard event based insertion/deletion/substitution metrics fail to capture important performance aspects. To illustrate the problem consider the plots shown in Figure 1. The first is is an example of the output from an experiment into recognition of continuous activities, [2]. The second is a synthesized variant of the same sequence. Each plot shows hand-labelled ground truth for five activities which we attempted to recognize in this experiment: use of a grinder, file, screwdriver, vice and drawer; it also indicates the time where no relevant activity was performed as *NULL*. Plotted above the ground truth are the recognition system's predictions, output on a timewise frame by frame basis.In example 1, most of the non-*NULL* activities visually correlate well with ground truth. Example 2 however seems much poorer, with several insertions, and with one event (filing) fragmented by short error segments.

Frame based performance A frame by frame comparison of ground and predictions leads to the confusion matrices shown next to the graphs. The matrices have been simplified to the summation of positive classes vs. *NULL*. Disappointingly, these are very similar suggesting more or less the same recognition quality. This is confirmed by the standard accuracy measure $acc_s = \frac{TP+TN-Sub.}{Total}$ which is identical for the two (where $TP = True\ Positive$,$TN = True\ Null$, and $Sub = Substituted$).

Event based performance When counts of insertion, deletion and substitution event errors are made for each of the examples (shown alongside the confusion matrices in Figure 1) the differences between the two examples become much clearer. However there remain two problems with this analysis. The first is with the definition of a correctly recognized event. As an example consider the events marked A, B and C in Figure 1. All three are counted as correct in both examples. However there are big differences in the way in which these are recognized. In example 2 events A and B are correctly recognized as two separate events, but in example 1 they are merged into a single event. Depending on the application one could argue that we have one correct and one deletion rather then two correct classifications. Event C on the other hand is correctly identified as a single event in example 1 while being split into three events in example 2. Thus one could argue for one correct and two insertions of the filing event.

The second problem is related to timing. While in example 2 the timing of event D exactly correlates with the ground truth, in example 1 its length is grossly overestimated (by a factor of 5).

2 Full categorization of errors

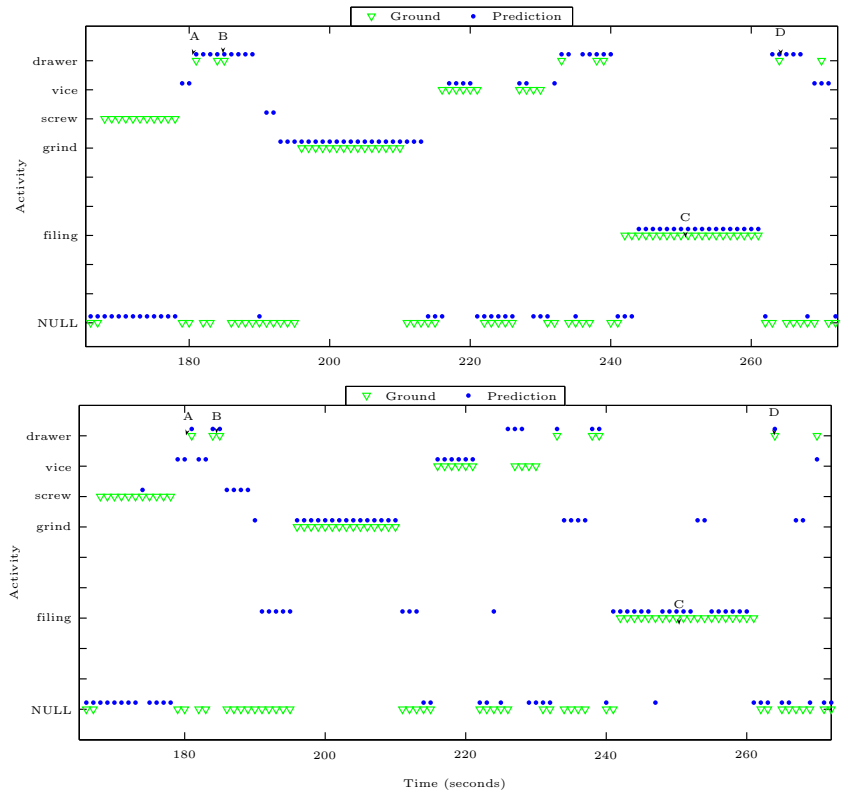
Based on the above observations we propose an evaluation methodology that

1. provides an objective, non ambiguous way to score event recognition,
2. includes event merges and fragmentations in the error summary, and
3. accounts for timing error as a separate category.

Event Scoring As a starting point we divide the pair of event sequences - ground truth and prediction - into a third sequence made up of segments. Each segment s_n has its boundaries marked by a change in either the ground or prediction as shown in Figure 2. Unlike whole events the segments can be objectively and without ambiguity scored. Also, on segment level, there can be no timing errors. Thus either the ground truth and the prediction are 100% identical in a segment, in which case it is scored as OK, or they are 100% different which means that the segment is scored as wrong.

Our method uses segment scores as a basis for event scoring in three stages.

[1] Swiss Federal Institute of Technology (ETH), Wearable Computing Lab, Zurich
[2] UMIT (University of Health Sciences, Medical Informatics and Technology, Hall i. Tirol, Austria
[3] ETH, Wearable Computing Lab, Zurich



frame by frame

traditional

ground

predict.	P	N
P	46(1)	27
N	17	16

SET based

#segments (#frames)

	D	U	F	N
I	1(1)			5(7)
O				7(20)
M				1(2)
N	1(11)	4(6)		

event

traditional

I	3
D	1
S	1

SET based

I'	4
D'	2
M	1
F	0

traditional

ground

predict.	P	N
P	45(5)	26
N	14	17

SET based

#segments (#frames)

	D	U	F	N
I	2(3)		1(2)	9(25)
O				1(1)
M				
N	1(2)	3(11)	1(1)	

event

traditional

I	10
D	0
S	2

SET based

I'	12
D'	2
M	0
F	1

Figure 1. Example from multi-class continuous activity problem (top); and (bottom) synthesized with identical sample accuracy. Tables on the right show errors, both frame by frame and event error count, for each example comparing traditional methods with proposed CET based methods

Errors First we look at insertions, deletions, merges and fragmentations

Insertion - Every *prediction* event that contains no segment which has been scored OK is a insertion.

Deletion - Every *ground truth* event that contains no segment which has been scored OK is a deletion.

Merge - Every *prediction* event that contains more than one segment which has been scored OK is a merge. Each merge event causes one or more deletion events of another class.

Fragmentation - Every *ground truth* event that contains more than one segment which has been scored OK is a fragmentation.

2. Correct Next we determine which events have been correctly recognized. To this end we search for pairs of *ground truth* and *prediction* events that (1) share a common OK segment and (2) have not been assigned to any of the above error categories. Such pairs are correctly recognized events.

3. Timing Timing considerations only make sense for events that are either correct, fragmented or merged. In fact timing scores are orthogonal to the other error scores, which means that each event is assigned both a 'true' error score and one or more timing scores.

Underfill - Any *ground truth* event which has been scored as either correct or as fragmented, and begins or ends with an non-OK segment, can be scored as an Underfill.

Overfill - Any *prediction* event which has been scored as either correct or as merged, and begins or ends with a non-OK segment, can be scored as an Overfill.

These scores provide information on the length of events; further information on more specific timing errors can be provided by dividing overfill and underfill into four additional categories. An underfill at the start of an event can be called *Delay*; one at the end can be called a *Shortening*. Equally, an overfill at the start of an event can be called a *Preemption*; and at the end a *Prolongation*.

Segment Error Table (SET) While the above error scoring is more informative than mere insertion/deletion based schemes it still fails to provide some useful information. The obvious missing part is absolute time duration (in terms of frames or seconds) for each type of error. In addition subtler information such as the relationship between error pairs is not captured. For example, it can be shown that the following parings are possible:

1. Where an event of one class is deleted, an event of another class is inserted, merged, or overfilled.
2. An underfill of one class means that there is an overfill or an insertion of another.
3. A fragmentation of one class means that events of another class are inserted at the fragmentation points.

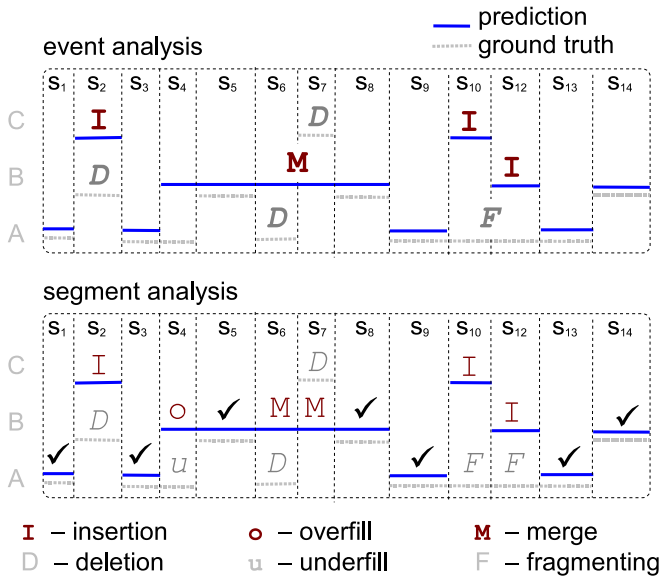


Figure 2. Some possible error combinations for a three class (A, B, C) example; the dotted vertical lines show how the sequence is broken up into segments s_n; upper diagram shows event errors, lower shows segment (frame by frame) error pairs

Our approach to deriving and presenting the above pairing information is based on three observations

1. The error pairing information can only be consistently provided on segment level. This is due to the fact, that a pairing implies that within the ground truth and the prediction remain constant. This is (per definition) the case for segments, but not for events.
2. When considering error pairing we need to treat 'true errors' and timing errors as part of a single error metric rather than two orthogonal evaluation measures. This is obvious from the listing of possible pairings above.
3. The error pairs always consist of a 'negative' and a 'positive' error. The former includes deletion, fragmentation and underfill (delay, shortening). In all cases part of the ground truth is 'overlooked' by the recognition system. As a consequence the negative error is derived by looking at the ground truth and checking whether the segment contributes to 'taking something away' from the ground truth. The positive errors are insertion, merge and overfill (preemption, prolongation). Here the recognition system adds something not present in the ground truth. We derive positive error by looking at the predictions and checking if a segment has contributed to such a false addition.

Based on the above considerations, assigning segments to pairs is done as follows:

Negative Errors derived from the ground truth events

1. A non OK segment that is part of a deleted *ground truth* event is assigned to the **deleted** error category
2. A non OK segment located within a correct *ground truth* event is assigned to the **underfill** error category
3. A non OK segment within a fragmented *ground truth* event is assigned to the **fragmentation** error category if it *is located somewhere between two OK segments of that event*
4. A non OK segment within a fragmented *ground truth* event is assigned to the **underfill** error category if it *is located within the event but outside of the first or last OK segments*

Positive Errors derived from the prediction events

1. A non OK segment that is part of a inserted *prediction* event is assigned to the **insertion** error category
2. A non OK segment located within a correct *prediction* event is assigned to the **overfill** error category
3. A non OK segment within a merge *prediction* event is assigned to the **merge** error category if it *is located somewhere between two OK segments of that event*
4. A non OK segment within a merge *prediction* event is assigned to the **overfill** error category if it *is located within the event but outside of the first or last OK segments*

To present the above information we propose what we call segment error tables (SET). As shown in Table 1 the rows of the SET correspond to the positive and the columns to the negative errors. Thus each field in the SET corresponds to a particular error pair and contains the number of segments that belong to this pair. Note that only 6 fields of the SET are occupied as there are only 6 possible error pairs.

	Deletion	Underfill	Fragmenting
Insertion	x	x	x
Overfill	x	x	
Merge	x		

Table 1. Continuous Error Table: rows denote Insertion, Overfill or Merging by predictions; columns denote Deletion, Underfill or Fragmenting of corresponding ground truth

Frame by Frame SET As with the standard confusion matrix, the SET can be used to present frame by frame information. Instead of counting segments belonging to each error pair the duration of the segments is summed up. Figure 1 shows SETs for the two examples with both segment information and the corresponding frame by frame counts (in brackets). Note that in these examples we have expanded the table to treat *NULL(N)* as a special case - we assume that we do not care to know whether *NULL* is being deleted, underfilled or fragmented, but we do want to know if it affects one of the positive classes. The top left section of these SET tables effectively represents all the positive class substitution errors.

Conclusion Whereas frame by frame confusion matrices were nearly identical, the SETs are very different reflecting the varied performance of the two examples. In particular the SETs show how much of the frame by frame error really contributes to event errors.

References

[1] Holger Junker, Paul Lukowicz, and Gerhard Tröster. Continuous recognition of arm activities with body-worn inertial sensors. In *ISWC*, pages 188–189, 2004.

[2] Paul Lukowicz, Jamie A Ward, Holger Junker, Gerhard Tröster, Amin Atrash, and Thad Starner. Recognizing workshop activity using body worn microphones and accelerometers. In *Pervasive, LNCS 3001*, 2004.