A Semi-Automatic Data Annotation Tool for Driving Simulator Data Reduction

Chris Schreiner, Harry Zhang, Claudia Guerrero, Kari Torkkola, Keshu Zhang

Motorola Labs, MD DW286
2900 S Diablo Way
Tempe, AZ 85282 USA

Abstract

Manually annotating large video and digital databases of driving behavior is costly and time-consuming. In this paper we discuss a data annotation tool that automates the process and reduces the number of man-hours required to annotate data. Our laboratory has utilized this tool on a large database of simulated driving data to develop context aware driving systems. Our semi-automatic data annotation tool supports our research efforts for driving database creation to enable data-driven approaches in the driving domain such as driving state and maneuver classification. The annotation tool employs Random Forests as bootstrapped classifiers which are then used to predict annotations for new data files. We describe an experiment which generated a large database of driving data with our DriveSafety simulator, the process by which annotations are automatically generated, and the results of how using the data annotation tool markedly reduced the amount of time required to annotate the data among three users with varying levels of annotation experience. Our major contribution in developing this tool is making parts of the annotation process automatic enabling the user to verify automatically generated annotations, rather than annotating from scratch. This tool has the potential to become a standard data reduction technique.
**Introduction**

In many cases, analysis of data from driving simulator research requires annotation of video and raw data variables to identify situations of interest that are not directly captured by any one particular variable. However, manual annotation can be a very time- and resource-consuming process. Our lab has developed a data annotation tool which can reduce the number of man-hours required to annotate data, by making parts of the annotation process automatic enabling the user to verify automatically generated annotations, rather than annotating from scratch. We have utilized this tool on a large database of simulated driving data to develop context aware driving assistance systems.

Our context aware driver assistance system tracks the state of the current traffic, the driving situation, and the state of the driver. Our approach to constructing this system is to learn directly from collected sensor data using statistical machine learning techniques. For example, our system can learn to classify different driving situations by collecting sensor data of those situations, training or learning models of the situations from the data, and then classifying incoming new sensor data stream using the models.

This paper will describe the annotation tool, the database of driving data we created on which we used this tool, and report on the effectiveness of using this tool in terms of the time required to annotate data.

**Creating the Database**

**Apparatus**

We collect data utilizing a DriveSafety simulator, which consists of a fixed based car surrounded by five front and three rear screens. We augment the simulator with our own custom software called Driver Advocate (DA) [1]. DA is a software state machine and a set of virtual sensors that monitor the virtual world of the driving simulator. The DA constantly monitors the environment, car position, traffic situation and gives ‘advice’ to the driver in real-time about ‘problems’ in that environment. In addition, our simulator setup also has several video cameras, microphones, and a SeeingMachines infrared eye/head tracker to record all driver actions during the drive that is synchronized with all the sensor output and simulator tracking variables. Altogether there are 425 separate variables describing an extensive scope of driving data - information about the car, the driver, the environment, traffic, and associated conditions. An additional screen of video is digitally captured in MPEG2 format, consisting of a quad combiner providing four different views of the driver and environment. Combined, these produce around 400Mb of data for each 10 minutes of drive time.
Experimental Methodology and Data Collection

Thirty-six participants took part in a study, the purpose of which was to develop an algorithm to directly identify driving maneuvers, which are not directly captured in the data. Each participant was scheduled to complete ten one-hour driving sessions. Thirty participants completed all ten sessions. In each session, after receiving practice drives to become accustomed to the simulated driving environment, participants were given a task for which they have to drive to a specific location. So that the drives were as natural and familiar for the participants in the simulator, the simulated world replicated the local metropolitan area as much as possible, so that participants did not need navigation aids to drive to their destinations. Signage corresponded to local street and Interstate names and numbers, and the topography corresponded as closely as possible to local landmarks. Participants were only instructed to drive as they normally would. Each drive varied in length from 10 to 25 minutes. As time allowed, participants did multiple drives per session.

The experiment produced a total of 132 hours of driving time with approximately 265 GB of collected data. Data from the simulator, DA, and eyetracker were synchronized and aligned using custom software [2].

Table 1 shows the 29 different driving states in which we were interested.

Table 1: Driving maneuver classes. “Cruising” captures anything not included in the other 28 classes.

<table>
<thead>
<tr>
<th>ChangingLaneLeft</th>
<th>PanicStop</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChangingLaneRight</td>
<td>Parking</td>
</tr>
<tr>
<td>ComingToLeftTurnStop</td>
<td>PassingLeft</td>
</tr>
<tr>
<td>ComingToRightTurnStop</td>
<td>PassingRight</td>
</tr>
<tr>
<td>Crash</td>
<td>ReversingFromPark</td>
</tr>
<tr>
<td>CurvingLeft</td>
<td>RoadDeparture</td>
</tr>
<tr>
<td>CurvingRight</td>
<td>SlowMoving</td>
</tr>
<tr>
<td>EnterFreeway</td>
<td>Starting</td>
</tr>
<tr>
<td>ExitFreeway</td>
<td>StopAndGo</td>
</tr>
<tr>
<td>LaneChangePassLeft</td>
<td>Stopping</td>
</tr>
<tr>
<td>LaneChangePassRight</td>
<td>TurningLeft</td>
</tr>
<tr>
<td>LaneDepartureLeft</td>
<td>TurningRight</td>
</tr>
<tr>
<td>LaneDepartureRight</td>
<td>WaitingForGapInTurn</td>
</tr>
<tr>
<td>Merge</td>
<td>Other (Cruising)</td>
</tr>
<tr>
<td>PanicSwerve</td>
<td></td>
</tr>
</tbody>
</table>

The Annotator

Our annotation tool provides an easy to use interface for annotating the data. The major properties of the tool are

- Ability to navigate through any portion of the driving sequence.
- Ability to label any portion of the driving sequence with proper time alignment
- Synchronization between video and other sensor data
- Ability to playback the video corresponding to the selected segment
- Ability to visualize any number of sensor variables.
- Providing persistent storage of the annotations
- Ability to modify existing annotations

The annotator uses Matlab for the entire user interface except for the video interface. Matlab is also used for graphing, clustering, and classification. Quicktime for Java, called through Matlab, is used for the video interface.

The user has three windows at their disposal during annotation (see Figure 1). The first is a main window (bottom of Figure 1), which shows a customizable view of the data collected. The main window is customizable in that the user can display as many or as few of the individual variables that were collected, can add multiple variables to each chart, and can add and remove charts from the window.

Figure 1. The video window, annotation window, and data stream window which makes up the user interface of the annotator.
The second window is the video window, which in our case shows the quad splitter, with the time synchronized and aligned to the data files. The third is the annotation editor where the user chooses the annotation label and marks the beginning and end of each.

**Manual Annotation**

Even though we automate parts of the annotation process, each new database from a particular study has to begin with manual annotation, the results of which will be used to train the annotation classifier. For this part of the process, the user would manually select the beginning and end points of the maneuver in the main window (lower part of Figure 1) and choose the correct annotation from a list of predefined annotations in the annotation editor window (upper right of Figure 1).

**Semi-Automatic Classification**

Here we will provide an overview of the machine learning techniques used for classification. A more detailed discussion can be found elsewhere [3]. We automated parts of the process by taking advantage of classifiers trained for the driving maneuvers [4]. Annotation becomes then an instance of active learning [5]. The human annotator is presented with automatically generated annotations derived from the classifier, along with each annotation’s associated confidence level. The human annotator can review all the generated annotations and focus on the annotations with low confidence levels. After the data has been verified, the data is added to the database and the classifier is retrained.

As the classifier improves due to increased size of training data, the confidence levels for the generated annotations improve, too, and the verification process takes less time.

As a classifier we used Random Forests [6]. Random Forests appear to be a very well suited tool for massive heterogeneous data sets, such as driving data. A driver activity classifier based on Random Forests is also fast enough to run in real time in a vehicle.

A characteristic of the driving domain and the chosen 29 driving maneuver classes is that the classes are not mutually exclusive. For example, an instance in time could be classified simultaneously as “SlowMoving” and “TurningRight”. The problem cannot thus be solved by a typical multi-class classifier that assigns a single class label to a given sensor reading vector and excludes the rest. This dictates that the problem should be treated rather as a detection problem than a classification problem.

Furthermore, each maneuver is inherently a sequential operation. For example, “ComingToLeftTurnStop” consists of possibly using the turn signal, changing the lane, slowing down, braking, and coming to a full stop. Ideally, a model of a maneuver would thus describe this sequence of operations with variations that naturally occur in the data (as evidenced by collected naturalistic data). Earlier, we have experimented with Hidden Markov Models (HMM) for maneuver classification [7]. A HMM is able to construct a model of a sequence as a chain of hidden states, each of which has a probabilistic distribution (typically Gaussian) to match that particular portion of the sequence [8].
sequence of sensor vectors corresponding to a maneuver would thus be detected as a whole.

Because the maneuver labels may be overlapping, we trained a separate Random Forest for each maneuver treating it as a binary classification problem – the data of a particular class against all the other data. This results in 29 trained “detection” forests. New sensor data is then fed to all 29 forests for classification. Each forest produces something of a “probability” of the class it was trained for. An example plot of those probability “signals” is depicted in Figure 2. The horizontal axis represents the time in tenths of a second. About 45 seconds of driving is shown. None of the actual sensor signals are depicted, instead, the “detector” signals from each of the forests are graphed. These show a sequence of driving maneuvers from “Cruising” through “LaneDepartureLeft”, “CurvingRight”, “TurningRight”, and “SlowMoving” to “Parking”.

The final task is to convert the detector signals into discrete and possibly overlapping labels, and to assign a confidence value to each label. In order to do this, we apply both median filtering and low-pass filtering to the signals. The signal at each time instant is replaced by the maximum of the two filtered signals. This has the effect of patching small discontinuities and smoothing the signal while still retaining fast transitions. Any signal exceeding a global threshold value for a minimum duration is then taken as a segment. Confidence of the segment is determined as the average of the detection signal (the probability) over the segment duration.

![Figure 2. A segment of driving with corresponding driving maneuver probabilities. Note that the classes are not exclusive. For example, a driving situation can be classified simultaneously as “TurningRight” and “SlowMoving”.](image-url)
An example can be seen at the bottom window depicted in Figure 1. The top panel displays some of the original sensor signals, the bottom panel graphs the raw maneuver detection signals, and the middle panel shows the resulting labels.

**Evaluation of the Classifier**

There were several questions we considered in evaluating the effectiveness of the annotator. First, how much faster the semi-automatic data annotation tool was compared with manual annotation? Second, was the benefit similar across different annotators of the same level of expertise? Third, was the benefit similar for novice annotators as for experts?

**Methods**

Three human factors professionals annotated the same data files. Two of the annotators had several years of experience in annotating data, while the third had no experience. The annotator was first familiarized with the tool to prevent any learning effects. Then a set of drives were manually annotated to create a baseline measure. The classifier was trained on these drives, and then the annotation expert was presented with 10 files to verify. After the ten files were verified, the classifier was re-trained with the additional files. Then ten more files were presented for verification. The files were the same for each annotator so that there were no differences in the classifier based on individual differences in the drives. The only differences in the classifier would then be the result of how each human annotator classified the data. This process continued until there was no significant improvement in the annotation metrics across two iterations.

Verification required correcting errors in the annotations from the semi-automatic annotation process. There were four types of errors that the classifier could make. They are: 1) boundary errors, where an event was correctly identified, but its start and/or end points are incorrect; 2) incorrect classifications, in which one state was misrecognized as another; 3) insertion errors, where the classifier inserts an event where none occurred; and 4) omission errors, in which an event occurred but was not classified. However, defining these errors is somewhat arbitrary. For example, the classifier may insert an event A that starts at time 1 and ends at time 4. The annotator may deem that this is incorrect and should be classified as event B beginning at time 2 and ending at time 3. These errors could be marked as a boundary error and an incorrect classification, or else as an insertion error and an omission error, and neither is more valid than the other. Therefore, for this paper we will report only the total number of errors.

**Results**

Results from one of the expert annotators were previously reported [9], and are presented again here to compare those results with the other expert annotator and the novice annotator.
Figure 3 shows the results for the two expert annotators and the one novice annotator in terms of the time spent annotating each file. An Analysis of Variance for the common iterations (manual and iterations 1-4) shows that both experts improved their annotation time across iterations ($F(4,91)=72.47$, $p<.001$). There was no interaction between expert and iteration ($F(4,91)=0.98$, $p=0.42$), and no significant differences in annotation times between the two experts ($F(1,91)=0.23$, $p=0.63$). Both expert annotators quickly reached a stabilized performance with the semi-automated annotation process, requiring less than one minute to annotate one minute of video (0.69 minutes for Expert #1 and 0.88 minutes for Expert #2). Expert #1 reduced annotation time compared to manual annotation by 80%, and Expert #2 reduced annotation time by 75%.

The results from both experts were combined and compared to the novice annotator. There was a significant interaction between expert level and iteration ($F(4, 137)=2.55$, $p<.05$). The novice user took longer to annotate data overall, but showed more dramatic improvement than the experts, reducing annotation time from 6.51 minutes per minute of video to 1.21 minutes per minute of video, an 81% reduction compared to manual annotation. So while the novice annotator’s performance did not reach less than one minute per minute of video as the experts did, there was still significant improvement in terms of the time required to annotate one minute of video.

Conclusions

We have presented an annotation tool that uses machine learning techniques to reduce the amount of time required to annotate video captured from a driving simulator. The tool enables the user to verify automatically generated annotations, rather than annotating
from scratch. Results show that using this tool to annotate a sample of driving data significantly reduces total annotation time for both expert and novice annotators. How effective the tool is on other driving databases will be heavily influenced by the underlying statistical differences between the annotations of interest. The annotator would not be as effective in automatically annotating events in which there are extremely small or no statistical differences in the variables or combinations of variables captured in the database. Our data annotation focused on classifying maneuvers, for which some maneuvers had clear statistical differences (e.g., the direction of steering angle and lateral acceleration were directly opposite for “TurningLeft” and “TurningRight” maneuvers) while others were much more subtle (e.g., identifying “WaitingForGapInTurn” or differentiating between “EnterFreeway” and “ChangingLaneLeft”). However, if we were instead interested in annotating driver behavior, such as driving with one hand or performing a secondary task, in which all the statistical differences among the collected variables were less perceptible, then the annotation tool would either take longer to effectively train, resulting in many more iterations before a significant improvement is made, or the tool would be unable to successfully generate accurate annotations.

References