Crowdsourcing Data for Autonomous Driving and Applying It

By Yaoyan Xi

Advisor: Prof. Chris Pollett
Abstract

A key to the success of autonomous driving hinges on the availability and the quality of the dataset upon which a steering wheel movement recommendation is based. This report summarizes the initial research efforts that have been made to crowdsource collecting the data for the subsequent application to train the model. Toward this end, three Android Apps have been developed and tested to capture a user’s GPS location, a vehicle’s movement, and road scenery recording. Apps have functioned well and fulfilled their purposes.

Keywords: autonomous driving, Android App, GPS location, accelerometer, video recording, data persistence

Introduction

Autonomous driving is an artificial intelligence guided transportation mode that is capable of sensing the environment and moving with little or no human input [1]. It is a fledgling realm that will significantly transform the way people interact with the vehicles, vehicles interact with each other, and how the roads are built 'smarter'. It will enable more productivities from human drivers, decrease the traffic accidents, and potentially redistribute traffic volume to tackle the traffic congestion.

The concept of autonomous driving has been around for a long time and can track back to 1920s [2]. In 1950s, some promising trials took place, and steady progression has been made since then. The first self-reliant and truly autonomous vehicles debuted in the 1980s thanks to Carnegie Mellon University’s Navlab [3] and ALV [4] projects and Mercedes-Benz and Bundeswehr University Munich’s Eureka Prometheus Project [5]. Since then, major players in the auto industry jumped into the wagon to conduct their own research and advance the technology.

With the rapid development of chip technology and significant improvement in the computing powers in the past few years, high-tech companies like Google, Uber, and Lyft have joined the game and taken the competition up a notch by racing to bring the deployment ready software to the market [6]. Google’s autonomous driving unit, Waymo, has 8 million miles of safe driving under the belt last July [7]. Uber has piloted the program that could transport the passengers with autonomous driving vehicles in Pittsburg [8].

This project strives to build a mobile phone application that collects real-time road scenery and steering wheel movement data, both direction and steering wheel arc degrees, via sensors. It is a fact that every
road has already been driven on by someone at some point and will likely be driven on by someone again in the future. It is also a fact that mobile phones are ubiquitous. So it seems promising to collect car driving data using a mobile application and then use that data to train autonomous vehicle AI systems. We will in this project develop applications to actually do this.

Specifically, our application will collect and integrate data such as lane, travel time or traffic volume, season or weather, speed limit, and GPS coordinates into the machine learning model to decide how to guide the steering wheel of autonomous vehicle. Road sceneries will be captured and saved with cell phone back camera, steering wheel movement recorded with motion sensor such accelerometer with the GPS coordinates. We will use the Tensorflow framework to build and train the model. More details will be discussed in the following sections.

**Deliverable 1: Getting familiar with Android Studio by building real-time GPS App**

Since this writing project collects and uses the raw data with users' cellphones, it calls for an App that is able to record the actual or analogous information on steering wheel movements and real time traffic condition among other data. The first step to take toward this goal is to code and test some Apps to realize some required functions separately before integrating them into a finalized App.

The App platform chosen here is Android Studio due to the fact that Android OS has been the most common cellphone operating system worldwide [9]. A good test App to get familiar with Android Studio is GPS App.

The source code was written with Android API 28, the latest version as of 2019. App uses LocationManager class and LocationListener interface to obtain GNSS coordinates. LocationListener requires the implementation of onLocationChanged(), onStatusChanged(), onProviderDisabled(), and onProviderEnabled() methods. onLocationChanged() is where the latitudes and longitude are obtained. Finally, permission to access the fine and course locations is provided in the manifest.

The source code is posted at the following URL:


Figure 1 shows the screenshot of this App as follows:
Android Studio provides an emulator to virtually test an App’s code and due to the lack of a working android cellphone at the time of coding this App, I decided to take advantage of this function to test my code as shown in Figure 1. Later, my tests were done on both an android cellphone and this emulator. The tests on the emulator turned out to be much slower than those on the physical cellphone.

Figure 1 shows the real time GPS coordinates at the time of the testing. With these coordinates, many further manipulations are possible with the Google Map API if wanted.

**Deliverable 2: Securing image dataset**

Sequential images are ideal for this project because the images and videos raptured on the road are temporally sequential, but static sequential images are hard to come by. Therefore, I turned my attention to the static discrete images instead. It turns out that collection of static discrete images are readily available. I downloaded the free images from Next7 project, which has 17,000 images under different lighting conditions, road conditions, and weather.

Here is one sample image from the dataset:
For this project, recognition of the objects such as the traffic lanes, other vehicles, pedestrians, traffic lights and weather features such as snow and rain is important. Our model will analyze these features and combine with the steering wheel movement dataset during the model training process. Figures 2 shows a typical static image that captures a traffic condition in San Francisco during the day time.

**Deliverable 3: Building movement detection and video capturing Apps**

Two Apps were coded with Android Studio to capture the accelerometer data and to record the real time traffic data with the camera. The accelerometer can capture the spatial accelerations in three dimensions and can capture the angular movement of a vehicle when it takes turns. Eventually, I intend to convert the accelerometer readings into the degrees of angular movements of steering wheel later on. Sensor.TYPE_ACCELEROMETER is used and SensorEventListener is implemented. onSensorChanged() method gives the accelerations in three dimensions.

The video recording App, on the other hand, captures the traffic conditions and stores the videos to an external SD disc. Internal storage will be investigated and tested later on. Intent class is used to delegate the recording via Android existing ACTION_VIDEO_CAPTURE action. Videos are saved as URI objects for subsequent retrieval and play. StartActivityForResults() and onActivityResult() are the centerpiece of the code. onClickListener class is invoked for RECORD button on the main view.
Figures 3 and 4 show the accelerometer application’s user interface.

An accelerometer’s readings change with a cellphone’s orientation. Figure 3 shows the three dimensional readings when the cellphone is placed with the head up where the y-axis reading is relevant. Since no calibration was done, the reading was different from the gravity acceleration, i.e. 9.8 m/sec², but this reading was consistent as a different axis’s reading which corresponded to a different orientation as shown in Figure 4, where the cellphone’s head was oriented to the right. Some easy tests like dropping the cellphone free fall showed the reasonable readings.
Other orientations (left and head down) were also tested, and the screenshots will not be attached here due to the space restraints, but they are available upon request.

The second half of this section is devoted to the discussion of the video recording App, and Figures 5 - 9 show the screenshots at the different stages of recording.
This is the initial interface a user will see upon activating the App. When the RECORD button is pressed, the App interface will switch to the recording interface as shown in Figure 6.
When the interface is changed to the recording interface, inherent Android video recording suite will be deployed here. When the recorder icon is pressed, the recording starts.

Figure 7. Emulator is recording a video.

When the recording is active, a timer will appear at the upper left corner and the cellphone’s camera will capture the video.

Figure 8. Emulator replays the video.
Depending on your preference, once the recording is terminated, a video replay will start immediately if you choose so. Repeatability can also be specified.

After the video recording is completed, the video will be saved to the storage of your choice. For this App, I chose to save to the extra storage in the cellphone. One can also code such that the storage goes to the internal storage and set up the communications with the database if present. I will work on this extension during the next phase of the project.

**Deliverable 4: Surveying image processing frameworks**

During the past five years, a new breed of artificial intelligent frameworks has sprouted thanks to the advancement in the open source software arena and their adoption by the heavyweight players in the IT world. I conducted a brief survey on these different platforms so that one can be selected for the next phase of this project.

A total five frameworks have been studied: Tensorflow, Keras, PyTorch, Caffe, and Theano per their popularities among the developers. Tensorflow is an open source framework backed by Google and based on Theano, and it was published on Nov. 9, 2015 on Apache 2.0. It is by far the most popular framework
with the developers. It is searched for on Google the most and has more code on GitHub than any other frameworks or libraries. Part of its appeals is that Tensorflow is production facing with good scalability in addition to research purposes, which means it can be deployed on a gamut of hardware machines, starting from cellular devices to computers with complex setups. It allows customization and tracking and controlling on the node level and provides high level operations such as multithreading and queues. Tensorflow is good for pattern recognition for images, videos, sound, voice, text, and time sequential data. One distinct characteristic of Tensorflow is that it embraces static computational graph philosophy, which requires definition of entire graph before using it, and that partially explains why Tensorflow has a steeper learning curve to the beginners. Despite of these drawbacks, we will select Tensorflow to build and train the machine learning model to process the sequential images.

The second candidate was Keras, which has a lot in common with Tensorflow. Why? Because Keras is a high level API based on Tensorflow, therefore Keras is easier to pick up and learn and only takes simple elegant codes to build some generic neuron network models, hence it is very user friendly. However, this benefit is based on the sacrifice of functionalities, that is, Keras can not do high level operations like multithreading or queues. Moreover, Keras has no control over nodes or weights associated, hence less flexibilities vs TensorFlow.

The third candidate, PyTorch, was another very popular framework with the developers. It is an open source framework based on Torch and backed by Facebook, and it was made public in Oct. 2016. Contrary to Tensorflow, PyTorch practices dynamic graph philosophy where computation graph can be defined and updated on the go, hence more flexibilities to handle variable length inputs. Modularity is another strength of PyTorch thanks to python. PyTorch is good for NLP applications.

Facebook simultaneously developed Caffe in addition to PyTorch. It was made public in April, 2017 and relatively lightweight and easy to use. However, it will need to define each layer in the neuron network. It is very good for image but bad at sequence applications, which is a mixed bag for our project. It is popular with mobile phone and other computationally constrained platforms, and it is not as good as TensorFlow in terms of speed, memory use, portability, and scalability.

The last candidate was Theano. It was published in 2007 as open source project by University of Montreal, and it pioneered in the machine learning and used to be popular in academic community, but its current popularity is in decline.
Next steps

So far, I have developed and tested three separate Android Apps to detect a user’s real time GPS location, to capture a vehicle’s movement to correlate with steering wheel’s movement direction and degrees, and to record the traffic scenery and save to an external storage. Moving forward, I will integrate these three Apps into one to be downloaded and tested by the end users.

Also, the recording function needs to incorporate the internal storage option in addition to the current external storage option. This will introduce the backend aspects into the equation: database communication with the App, database management and concurrency, and cloud operation and so forth.

With all the data collected, I will feed them to the TensorFlow framework to initially train the models to process both static images and continuous videos and check how the model prediction and recommendation fare in the simulated environment. TensorFlow boasts a list of Off The Shelf (OTS) models in its library to the developers but requires some time understanding all the nuts and bolts under the hood. For the image and video inputs, feature selection is very important to recognize the key elements in the image and video. I will anticipate some efforts and resources spent on this subject.

TensorFlow also allows the control on the node level so more insights can be had in terms of how hidden layers work and whether manual adjustment of weights is necessary. TensorFlow is an excellent and powerful framework to work with to gain a better understanding of convolutional neuron network (CNN) and its inner work.

If the lab simulation goes well, the next step is to test the App on the road with the actual streaming video as model input, and let model recommendation guide the driving manually because the automatic hand-free operation of steering wheel based on the model recommendation is beyond my current research arena due to the time and cost restraints.
Reference


