Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 198 (2022) 237–242

Procedia Computer Science

www.elsevier.com/locate/procedia

International Workshop on Artificial Intelligence Methods for Smart Cities (AISC 2021)
November 1-4, 2021, Leuven, Belgium

ML-based re-orientation of smartphone-collected car motion data

Enrico Bassetti[a], Alessio Luciani[a], Emanuele Panizzi[a,*]

[a] *Sapienza University of Rome*

Abstract

Smartphone sensors can collect data in many different contexts. They make it feasible to obtain large amounts of data at little or no cost because most people own mobile phones. In this work, we focus on the collection of smartphone data in the car. Motion sensors, such as accelerometers and gyroscopes, can help obtain information about the vehicle's dynamics. The possible spatial orientations of the smartphone in the car are infinite, and this can be a problem in an attempt to extract patterns from the data. Thus, we propose an approach to automatically re-orient smartphone data collected in the car to a standardized orientation (i.e. with 0 yaw, roll, and pitch angles with respect to the vehicle). We use a combination of a least-square plane approximation and an ML model to infer the relative orientation angles. Then we populate rotation matrices and perform the data rotation. We trained the model by collecting data both in an actual vehicle and using a vehicle physics simulator.

Keywords: smartphone; parking; sensing; implicit interaction; machine learning; curb; parallel; angle parking; smart city; context aware

1. Introduction

The widespread presence of powerful smartphones in people's pockets incentivized discovering new ways to exploit their sensors. A particular setting in which smartphone sensors such as accelerometers and gyroscopes can be beneficial is in cars. They can replace more sophisticated and expensive approaches for motion data collection. Innovative transportation challenges need vehicle data: using a tool that is so popular among people makes data collection much more scalable.

Unfortunately, obtaining high-quality data from smartphone sensors is far from being straightforward. Users can position their smartphone in the car in many different ways (e.g. the smartphone can be in a pocket with the screen facing the car window or in a cup holder facing the driver), and often they use it and move it while traveling. This fact can impact the interpretation of the sensed data, making the collection useless due to an unknown orientation.

[*] Corresponding author.
E-mail address: panizzi@di.uniroma1.it

Therefore, this work aims at processing data collected from a smartphone in a car to orient it along the car's axes. We accomplish this by performing a series of data point rotations in a 3D space and similar operations, using a Machine Learning model trained with vehicle simulated data. Using a simulator, we could speed up the data collection procedure, although a simulation comes with some drawbacks, as described in this paper.

The resulting processed data looks like if collected by a smartphone placed horizontally in the car, with the screen facing the car roof and the top oriented towards the car's front (Figure 1).

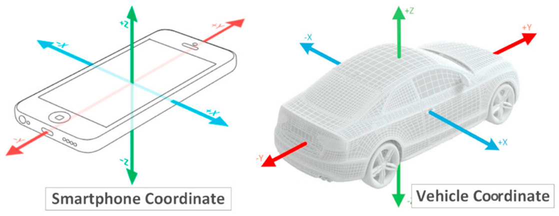


Fig. 1. Smartphone and car axes resulting orientation

1.1. Overview

Initially, we will discuss the first step, namely, the finding of the horizontal least-squares plane and the subsequent data rotation. The least-square plane allows us to understand the orientation Euler angles between the raw data points and the gravity plane. Then, the smartphone's data can be rotated according to those two angles: pitch and roll. The following step is to find the remaining orientation yaw angle. We decided to use an ML technique to find it depending on the car's motion data points. So, the final sections will cover all the processes that took us to develop the model and perform the eventual yaw rotation: data collection in the simulation, model training, data rotation, and performance evaluation.

2. Related Work

Other works about the orientation of a smartphone in a car are present in literature. Some of them are about classifying the type of placement of a smartphone in the vehicle by devising several positions that it could take: [9] describes a procedure that uses a clustering approach to learn different classes of smartphone positions. However, there is no focus on processing the data to standardize it to a fixed position and orientation. Our proposed work, instead, mainly focuses on that part. [4] focuses on detecting front-to-back and side-to-side car moving dynamics to understand whether the driver is using the smartphone. This approach is similar to our ML technique to infer the smartphone's orientation. The approach in [5] is similar to ours. Specifically, ML is used to infer the entire 3D rotation matrix. Instead, we first make a horizontal alignment with a more deterministic least-square approximation and then use ML to infer the remaining 1-axis angle.

Others exclude the orientation calculation by simply fixing the device in a given position [3].

3. Least squares

The acceleration samples are studied to figure out which is the most probable car horizontal plane (Figure 2). During a car trip, accelerations[1] on the vertical axis (i.e. Z-axis) are close to 0 since they are due mainly to road disturbances and beginnings of ascents or descents. So, in most cases, the accelerations are concentrated on the axes X and Y. This fact lets us compute the least-squares plane [7] for the acceleration points in time.

However, before deriving least-squares plane candidates, we need to remove outliers. To do that, we use the Z-Score algorithm [8]. First, we compute the standard deviation for every axis. Then, we empirically set the max z value

[1] The acceleration points are represented by three coordinates: X, Y, Z

that we accept to 5 (value chosen by optimizing FAR and FRR when looking for points of the plane in our dataset), and calculate the Z-Score as $z_{p_i} = (x_{p_i} - \mu)\sigma$ for each point, for each axis. Finally, we discard all points exceeding the max z value. This removal is only temporary to compute the right least-squares plane. Once the plane coefficients are defined, we use them to rotate all points in the space towards the horizontal plane (i.e. the plane $Z = 0$).

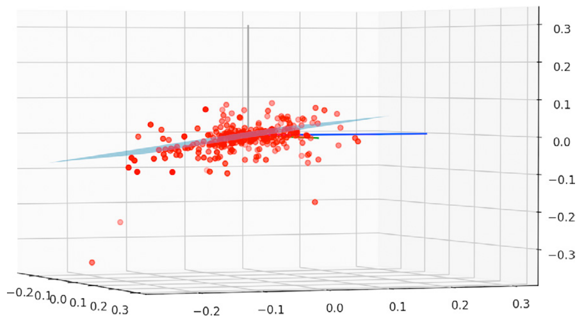


Fig. 2. 3D distribution of acceleration points. The least squares plane is drawn in blue (side view) - values are in m/s^2

The idea behind the rotation is to rotate each acceleration data point (i.e. a 3D vector composed of the acceleration components on the three axes: X, Y and Z) around the axis common to the two planes.

Let N be the vector normal to the plane $Z = 0$, M be the vector normal to the newly found car horizontal plane, $dot()$ the dot product calculation, $norm()$ the normalization and $cross()$ the cross product calculation. First, we calculate the required angle Θ as $cos\Theta = dot(M, N)/(norm(M) * norm(N))$. Then, the axis around which the rotation will be made is calculated as $cross(N, M)/norm(cross(N, M))$. Finally the rotation matrix is calculated as:

$$c = cos\Theta \qquad s = sin\Theta \qquad C = 1 - c$$

$$Q(\Theta) = \begin{bmatrix} xxC + c & xyC - zs & xzC + ys \\ yxC + zs & yyC + c & yzC - xs \\ zxC - ys & zyC + xs & zzC + c \end{bmatrix}$$

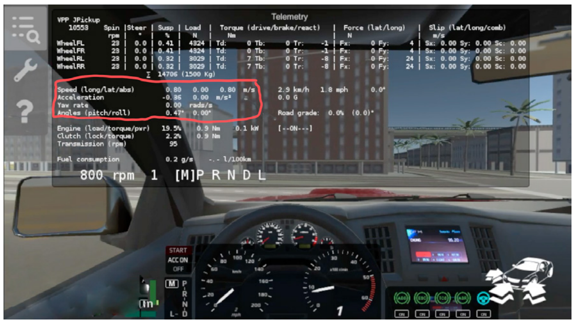
The rotation matrix calculation is further described in [1].

Then, the Q matrix can be used to rotate every single point p_i as $p'_i = dot(Q, p_i)$. If the two planes are similarly oriented, the rotation does not happen since it would not improve the precision. At this point, the processed data has a similar shape to the one that a smartphone would collect in a parallel position relative to the car's horizontal plane.

4. Data collection in the simulation

Collecting a sufficient amount of data in actual vehicles is a non-trivial task for different reasons. Since the considered system deals with supervised learning and requires labeled data, the collection process has to be conducted exclusively by operators that are well aware of the system requirements and pay close attention to the collection details. Wrongly collected or incorrectly classified data are deleterious for the quality of the final model. Also, operators should annotate details about the smartphone's orientation in the car during the collection and input them into the dataset. Measuring orientation angles in a natural environment can be challenging for human operators. Finally, to have a consistent amount of samples, the operators would have to make many authentic vehicle trips, which is time- and fuel-consuming.

According to fundamental physics laws, the simulation provided by vehiclephysics.com (Figure 3) allows us to emulate the behavior of a car that moves almost precisely (at least for our purposes) like a real one in the real world. The simulation generates all the movements and the data that we need out of the car. We can extrapolate this data and prepare them in the same way we do in the actual samplings.



```
                                         Telemetry
VPP JPickup
   10553    Spin |Steer | Susp | Load  |  Torque (drive/brake/react)    | Force (lat/long)     | Slip (lat/long/comb)
             rpm |   °  |   %  |   N   |    Nm                          |   N                  |   m/s
WheelFL       23 |  0.0 | 0.41 |  4324 | Td:     0 Tb:     0 Tr:    -1 | Fx:     0 Fy:      4 | Sx: 0.00 Sy: 0.00 Sc: 0.00
WheelFR       23 |  0.0 | 0.41 |  4324 | Td:     0 Tb:     0 Tr:    -1 | Fx:     0 Fy:      4 | Sx: 0.00 Sy: 0.00 Sc: 0.00
WheelRL       23 |  0.0 | 0.32 |  3029 | Td:     7 Tb:     0 Tr:    -8 | Fx:     0 Fy:     24 | Sx: 0.00 Sy: 0.00 Sc: 0.00
WheelRR       23 |  0.0 | 0.32 |  3029 | Td:     7 Tb:     0 Tr:    -8 | Fx:     0 Fy:     24 | Sx: 0.00 Sy: 0.00 Sc: 0.00
                    Σ 14706 (1500 Kg)

Speed (long/lat/abs)        0.80    0.00    0.80  m/s     2.9 km/h   1.8 mph     0.0°
Acceleration               -0.36    0.00  m/s²            0.0 G
Yaw rate                    0.00  rads/s
Angles (pitch/roll)         0.47°   0.00°                 Road grade:  0.0%  (0.0)°

Engine (load/torque/pwr)   19.5%     0.9 Nm    0.1 kW     [--ON---]
Clutch (lock/torque)        2.2%     0.9 Nm
Transmission (rpm)          95

Fuel consumption            0.2 g/s     -.- l/100km
     800 rpm   1   [M]P R N D L
```

Fig. 3. Vehicle Physics simulator

4.1. Structure of the simulation

The simulation comes in the form of a customizable Unity 3D project, adaptable to specific needs. As a foundation, there are some basic vehicles and maps provided. One of the default maps includes on-street parking with lines drawn on the floor that specify the exact positions of the parking lots. This map also has city streets and highways to emulate real driving situations and different parking motions.

The two default car models provided are a sports car and a pick-up truck. We decided to use the latter because its acceleration and brakes are more balanced and similar to most cars. Furthermore, we decided to limit the vehicle's maximum acceleration and braking power to avoid unnatural levels. In fact, by default, the vehicle would accelerate significantly before automatically shifting the gear. In that case, the engine revolutions indicator would touch 4000-5000 rpm, and the acceleration would be too disruptive. Those levels are far from the usual city driving.

The vehicle can be either controlled with a keyboard or with a game controller. The main difference between the two options is that the car control is discrete in the first case, while in the second one, it is continuous thanks to moving cursors. Thus we can apply a variable acceleration to the vehicle that better resembles a real interaction with the car.

Since this is a Unity 3D project, we also have access to external frameworks. We experimented with connecting the simulation to an Oculus VR. The Unity project sets a camera in the car by default. In order to connect the Oculus visor, we had to replace the default camera with the Oculus one and attach it to the vehicle. After doing that, the connection was seamless, and the driving experience was even more realistic and straightforward to simulate.

4.2. Obtaining the required simulation data

The Vehicle Physics Pro simulation offers a comprehensive API and access to vehicle components. In particular, there is a wide variety of metrics related to physics values in a whole section of the API dedicated to telemetry data. The telemetry section is responsible for collecting the physics data generated by the core simulator and making them programmatically readable.

The Vehicle Physics Pro simulator is written in C#, and its code is accessible from third-party scripts, like ours. We used a "vehicle" object that contains most of the information related to the car's motion. This way, we extracted all the features that we needed: *heading* (represented as the Euler angle on a specific axis), *acceleration* on X, Y and Z axes (represented as the local acceleration), *rotation rate* on X, Y and Z (represented as the angular velocity), and *speed*.

The script launches when the simulation starts, and it contains a thread that can be in two states: "not recording" and "recording". As soon as the script starts its execution, it spawns the new thread in the "not recording" state. When the user presses a specific key on the keyboard, the thread goes into the "recording" state and starts sampling the data at a fixed rate (10 times per second). After the pressure of another button, the thread goes back to the "not recording" state and saves the samples in a JSON file. By the way, it does not store the entire sampling, but only the last minutes: there is a fixed buffer that, once filled up, starts dropping the oldest samples in favour of the new ones. As we do not need very long sampling periods, we can save memory and maintain the size of the output files beneath a certain threshold. So, we followed this procedure: i) starting the simulation, ii) pressing the record button, iii) driving the car around the map and iv) stopping the recording. We did this many times, trying to cover as much terrain as possible.

We tried to make each drive slightly different from the previous ones to populate the dataset with diverse information. To do this, we chose different streets, turns, acceleration and braking patterns. We also simulated parking motions in different styles and shapes, like parallel parking along the street and angle parking in the dedicated parking location.

4.3. Simulated data processing

Unlike the actual data situation, simulated data does not have human interference, so the gyroscope fluctuations problem does not appear. The ”virtual smartphone” is constantly locked in the same position and orientation in the car, parallel to its horizontal plane and heading towards its front.

4.4. Features extraction for orientation model

The required angle in the simulated data is of 0°. So, to create some valid labels, the data can be rotated on the horizontal plane of a random angle. This kind of 3D rotation is the one based on Euler angles [6]. We labeled the sample with the random angle value. We repeat this process many times for every sample, picking random angles, eventually obtaining a larger dataset. In our case, we repeated it 100 times. Starting from a 150-sample dataset, we rapidly ramped up to a more significant 15000-sample one.

We rotate the data points around the Z-axis (i.e. the axis orthogonal to the car’s horizontal plane) using a simple rotation matrix (Equation 1, where θ is the randomly picked angle) since it only interests one axis.

$$R_z(\Theta) = \begin{bmatrix} cos(\Theta) & -sin(\Theta) & 0 \\ sin(\Theta) & cos(\Theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

At this point, we can rotate every single point using the R matrix, using the procedure described in [2]: $p'_i = dot(R, p_i)$, where $dot()$ is the dot product calculation.

At this point, we must compose the feature vector. We used the following features for each sample: acceleration mean for X and Y axes, acceleration standard deviation for X and Y axes, acceleration mean error for X and Y axes, rotation rate mean for X and Y axes, rotation rate standard deviation for X and Y axes, rotation rate mean error for X and Y axes. The Z-axis is not significant in this case.

A tabular dataset is generated, composed of rows of features and a final target value. The file format is CSV (comma separated values), so the first row indicates the names of the columns, separated by commas. All the other rows contain data, namely the relative value for every column, including the target value in the target column. Also, the data values are separated by commas, according to the column names.

5. ML orientation model

We created the model using Apple’s Create ML tree ensemble regressor. Create ML offers a wide range of tools to create, train and test new ML models for different purposes, either via a GUI or programmatically in Swift. Create ML provides both classifiers and regressors. For our purposes, we need a regressor that outputs a floating-point number based on the input sample. In our case, we need to obtain an angle value as the output, expressed in radians, from 0 up to $2 * PI$.

The regressor used is a tree ensemble, a predictive model composed of a weighted combination of multiple regression trees. In general, combining multiple regression trees increases the predictive performance.

This kind of model receives a vector of features as input 4. So, the set of features used as input are previously vectorized and made compatible with the model.

The training procedure does not imply long waiting times since this type of model does not have the recursive complexity and layers depth of a more sophisticated neural network.

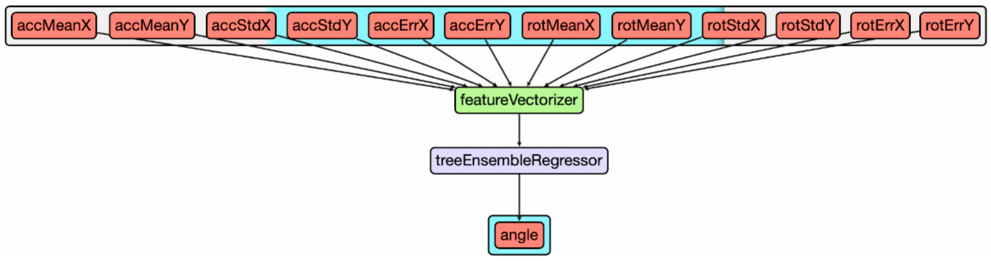


Fig. 4. Orientation ML model structure

5.1. *Orientation model testing*

By definition, we collect correctly oriented data in the simulation with our virtual smartphone in the car. So, we can take advantage of this aspect and generate samples by taking simulation recordings and rotating the motion data of a chosen yaw angle. Since we choose this angle, we can then use it as our ground truth label during training. In other words, our training set is composed of pairs of purposely rotated car motion recordings as input and their respective angle of rotation as output. Moreover, since we can choose random angles for this preliminary rotation, every recording can be used more than once by rotating it using different angles.

We trained the model on 150 recordings, each rotated in 100 random angles. The feature set that reached the best results is composed of the following features for the X and Y axes: acceleration mean, acceleration standard deviation, acceleration error, rotation mean, rotation standard deviation, rotation error. After training the model, we tested it on unseen data, predicting angles. We achieved a root mean square error of 0.34 radians. This means that the model is, on average, only wrong by about 19 degrees.

6. Limitations, conclusions and future work

In this publication we presented a way to identify the car orientation using a combination of least square plane and a tree ensemble ML model using data from a smartphone sensor on-board.

Limitations of this paper include the use of simulated data for training and evaluation. We are currently working on extending the data collection to a real world environment, with a smartphone in a vehicle. This poses additional challenges: e.g. smartphone orientation must be correctly labeled for each recording.

References

[1] , a. Rotation matrix axis and angle. https://en.wikipedia.org/wiki/Rotation_matrix#Axis_and_angle. [Online; accessed 04-May-2021].

[2] , b. Rotation matrix three dimensions. https://en.wikipedia.org/wiki/Rotation_matrix#In_three_dimensions. [Online; accessed 04-May-2021].

[3] Johnson, D.A., Trivedi, M.M., 2011. Driving style recognition using a smartphone as a sensor platform, in: 2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC), pp. 1609–1615. doi:10.1109/ITSC.2011.6083078.

[4] Johnson, G., Rajamani, R., 2020. Smartphone localization inside a moving car for prevention of distracted driving. Vehicle System Dynamics 58, 290–306. URL: https://doi.org/10.1080/00423114.2019.1578889, doi:10.1080/00423114.2019.1578889, arXiv:https://doi.org/10.1080/00423114.2019.1578889.

[5] Kang, L., Banerjee, S., 2017. Practical driving analytics with smartphone sensors, in: 2017 IEEE Vehicular Networking Conference (VNC), pp. 303–310. doi:10.1109/VNC.2017.8275595.

[6] Milligan, T., 1999. More applications of euler rotation angles. IEEE Antennas and Propagation Magazine 41, 78–83.

[7] Romero, L., Garcia, M., Suárez, C., 2014. A tutorial on the total least squares method for fitting a straight line and a plane. REVISTA DE CIENCIA E INGENIERÍA DEL INSTITUTO TECNOLÓGICO SUPERIOR DE COATZACOALCOS 1, 167–173.

[8] Shiffler, R.E., 1988. Maximum z scores and outliers. The American Statistician 42, 79–80. URL: https://amstat.tandfonline.com/doi/abs/10.1080/00031305.1988.10475530, doi:10.1080/00031305.1988.10475530, arXiv:https://amstat.tandfonline.com/doi/pdf/10.1080/00031305.1988.10475530.

[9] Wahlström, J., Skog, I., Händel, P., Bradley, B., Madden, S., Balakrishnan, H., 2019. Smartphone placement within vehicles. IEEE Transactions on Intelligent Transportation Systems PP, 1–11. doi:10.1109/TITS.2019.2896708.