Extraction of vehicle behaviors at intersections

Nelson de Moura, Fawzi Nashashibi

Extraction of vehicle behaviors at intersections

Nelson de Moura and Fawzi Nashashibi

***Abstract*— Simulation of realistic urban environments is essential for the validation of decision-making algorithms in automated driving. To populate such spaces with independent agents that execute trajectories and that correctly interact with each other it is necessary to identify all possible behaviors that may arise from a set of real users trajectories. Using a two-step clustering process on trajectories with different duration, first on the position vector and then on the longitudinal velocity and acceleration, a number of behavior profiles can be identified per trajectory in intersections. To speed up the maneuver identification, the position clustering is executed on the dissimilarity matrix created using the dynamic time warp (DTW) measure between trajectories. As for the second clustering, the use of both longitudinal velocity and acceleration allows the representation of specific behaviors for each possible path. In both procedures the k-means clustering with DTW dissimilarity measure was used, combined with the Davies-Boudin score to define the optimal number of clusters. The final result provides the identification of behaviors in different geometric dispositions independently from map information and keeping the explainability property concerning each cluster.**

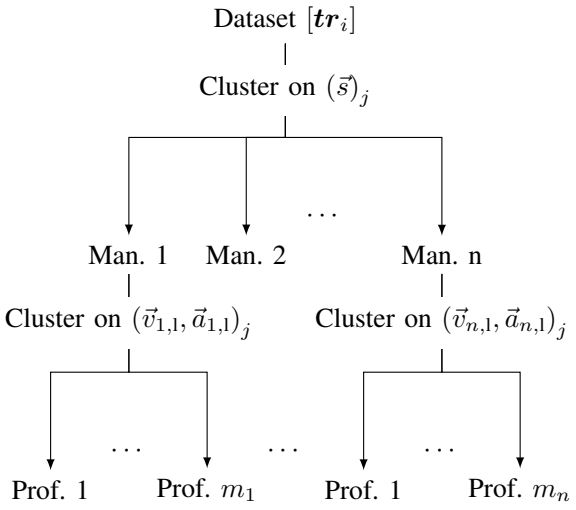


Fig. 1: Representation of the clustering process

I. Introduction

Despite all the hype in the late 2010s, automated vehicles (AV) are still in the research and development phase, with a few level 3 systems [1] starting to be deployed. The burning question on the minds of common folk concerns the behavior of AVs in dangerous/dilemma situations, which will determine its the social acceptability. Simulation is an important and valuable tool not only to develop and validate such systems but also to inform and demonstrate to the public the functioning of a hypothetical AV.

The majority of the research done on decision-making for AVs uses real data to represent real road users, which is ideal, but exactly as it was captured. It is impossible to be more realistic than that, but such rigid use does not expose the tested algorithms to the entire range of behaviors from other road users that the AV might face during its deployment, given that some behaviors might be rare but altogether essential to be considered. The idea presented in this article is then to extract the latent behavior of different road users in some scenarios, allowing in a latter time to be generalize over and to generate synthetic simulation actors that develop trajectories inspired by these latent behaviors and interact with each other in a manner that is compatible with reality.

This research has been funded by the plan "France relance", grant agreement number ANR-21-PRRD-0005-01; Nelson de Moura and F. Nashashibi are with INRIA, 75012 Paris, France. E-mail: {nelson.demoura; fawzi.nashashibi}@inria.fr.

This "extraction" from real data involves the clustering of different trajectories that should be captured from real situations. How to cluster time series of different sizes have been studied for many years, notably starting in the speech recognition domain [2], [3]. For example, some of the seminal works in the subject are [4]–[6], that laid the foundations of the dynamic time warp (DTW) distance measure and its simplification constraints, which are today one of the most efficient method to cluster time series of different sizes. One of the possible alternatives, to fit the available trajectories and then extract an equal size time series has been proven to not produce a difference in accuracy [7], while two other possible approaches are to classify based on extracted features from the raw data (application specific) or to build some model from it [8] (and risk losing the explainability property).

An example of a similar approach proposed here to treat trajectories raw data is [9], where two different clustering methods, the k-means algorithm and the fuzzy c-means (FCM) are used with the longest common sub-sequence (LCSS) measure combined with a Gaussian kernel to define the similarity matrix between trajectories. Using the raw data made available by [10] both clustering methods produced similar results, but no method to determine the correct number of clusters were proposed since the rand index (RI) is used to determine the best result (which entails knowledge of some ground truth). Another example of the same type of approach is [11], which uses the DTW as distance measure with a c-medoids algorithm to cluster time series. Since it uses the gap statistic to obtain the optimal number of clusters,

it unfolds each time series into the frequency domain so as to correctly calculate the null distribution for the gap calculation (which compares a candidate distribution to one that does not have a separable structure). However, the approach was tested only with synthetic data.

A generative model can also be built based on the available raw data, as [12] did; it solves the problem of scarcity directly with a generative model trained with raw data. All the considered trajectories are fed into a partially observable Markov decision process that models an imitation learning task, which is trained using a generative adversarial framework. The entire framework is tested using two datasets: one synthetic, based on a microscopic road simulation and another created by the tachograph data from an entire fleet of taxis inside a district in Seoul. It can be clearly seen that the lack of some types of trajectories affects the accuracy of the generator under real condition, while if known *a priori* which are these rare trajectories they could be duplicated or have their importance increased.

The contribution of this paper can be surmised in two points: fast clustering for maneuver detection and behavior determination from maneuvers in an explainable way and independently from map information.

II. Clustering the trajectories

Each trajectory can have different sizes, since it consists of a time series for the position, velocity and acceleration in a scenario like figure 2, for an agent i that has a trajectory of size j. The first piece of information necessary to be extracted from data is the expanded maneuver[1] being executed to then compare the long behaviors accordingly.

$$\boldsymbol{tr}_i = \left[(\vec{s}, \vec{v}, \vec{a})_j \right], j \in \mathbb{N} \tag{1}$$

Let's assume that the working dataset contains a certain number of trajectories of vehicles in an intersection. To obtain the labeled data so as to interpret the behavior of each vehicle in the intersection, two different clustering operations are proposed: the first one to separate the different maneuvers that might be executed in an intersection, and the other one, to be executed in each defined class from the first one, to finally classify the range of behaviors displayed during an specific maneuver.

The first separation can be done using exclusively the position information in each trajectory. Given that each detected cluster now has the same lateral evolution, the second clustering operation can be done using the longitudinal velocity and acceleration information. This sequence of operation is illustrated by figure 1, where 'Man.' refers to maneuver, 'Prof.' to behavior profile and $\vec{v}_{\text{-,l}}, \vec{a}_{\text{-,l}}$ refers to the longitudinal velocity and acceleration.

The dataset chosen to test the extraction of different driving profiles from real data was the InD dataset [13], which is a bird-view urban dataset of intersection trajectories. All the trajectories are obtained from a video recording of the intersection (details about the trajectory calculation are in [13]). Four different intersections were observed by a drone multiple times, defining the data-batches that hold the set of trajectories to be analyzed. Equation (2) describes more appropriately the data that compose each trajectory.

$$\boldsymbol{tr}_i = \left[(\vec{s_x}, \vec{s_y}, \vec{v_x}, \vec{v_y}, \vec{a_x}, \vec{a_y})_j \right], j \in \mathbb{N} \tag{2}$$

Fig. 2: Example of an intersection from InD dataset (Obtained from [13])

With the process and the information that needs to be extracted defined, the next step is to find a method to cluster the data correctly. K-Means is a fairly popular and robust method, with implementations available in multiple programming languages, so it was the chosen method. However, there is an important characteristic that needs to be accounted: the trajectories are time series with different sizes, which means that the Euclidean distance, the usual metric to calculate the proximity between data points in a cluster, cannot be used in a straightforward way.

To be able to compare two time series with different sizes one has to find a relation between the points of each series, which is done using an algorithm called dynamic time warping (DTW). With it, a similarity score (analogue to the Euclidean distance) can be calculated, and thus, the K-Means can be executed data points that have different sizes.

A. Dynamic time warp (DTW)

DTW was first introduced in the speech processing domain as a way to compare two time series that have different phases. Even though the trajectories studied here were sampled at the same frequency, they have different sizes and might correspond to the same maneuver in an intersection. Consider two discrete time series, represented by (3) and (4), with different sizes n and m, where $K = \{k_0, k_1, \ldots, k_n, \ldots, k_m, \ldots\}$ represents the sampled periods:

$$R[K] = r[k_0], r[k_1], \ldots, r[k_n] \tag{3}$$

$$S[K] = s[k_0], s[k_1], \ldots, s[k_m] \tag{4}$$

The goal of the DTW is to calculate the optimal sequence of pairs of point indexes, one from each time series. This is done by minimizing the euclidean distance between the points pointed by the index pair, from $(r[k_0], s[k_0])$ to

[1]It represents the maneuvers executed by a single trajectory which may be more than one typical atomic maneuver.

$(r[k_n], s[k_m])$, using a certain set of increments to walk from the former to the latter. In the standard implementation (equation 6) three steps are tested: advance one for the index 1, advance one for index 2 or advance one in both. Equation (5) defines the DTW from R and S as the calculated sum of distances, which are determined by the recursion in equation (6), for $0 \leq i \leq k_n$ and $0 \leq j \leq k_m$.

$$DTW(R, S) = \gamma(k_n, k_m) \tag{5}$$

$$\gamma(i, j) = d(r[k_i], s[k_j]) + \min\left[\gamma(i-1, j), \gamma(i, j-1), \gamma(i-1, j-1)\right] \tag{6}$$

There are multiple DTW variants, some changing the walk used in the recursion (6) (constraint DTW [14]) or adopting restriction on the elements to be considered by equation (6) (Sakoe-Chiba band [15]; Itakura parallelogram [5]). In [16], the DTW implementation is constrained to avoid a degenerate comparison between two series (when the path moves in using the vector $(i-1, j)$ or $(i, j-1)$ multiple times) that have very different sizes while [17] implements a differentiable measure to be used as kernel for machine learning methods. A continuous DTW version was implemented by [18]. During the clustering operations discussed here the DTW version used will be the one displayed by equation (5) since continuity and differentiability is not important for the task.

Usually, when the Euclidean metric is used, the centroid of a set of series can be calculated simply by summing all the elements and dividing by the number of series in the set. To do so with series of different lengths the Dynamic Barycenter Averaging (DBA) can be used, producing an average inspired series that minimizes the sum of squared DTW distances to each element of the set [19]. In [20] the same procedure was expanded to calculate a weighted average of a set of time series.

B. K-Means clustering

The K-Means clustering is an unsupervised learning method that tries to separate a set of data into k pre-defined clusters given a measure of similarity. It is fairly well known today for its versatility and simplicity. In the instances where the k-means clustering is used in this work, the Euclidean metric, popularly used to determine the dissimilarity between elements inside a cluster will be replaced by the DTW distance measure. There are some possible problems with this substitution given that the DTW is not a metric (it does not assure the triangular inequality property [11]), but the conclusion of [3] guarantees that, if equation (6) is solved according to the *Bellman's* optimal principle, the triangle inequality is loosely satisfied, thus avoiding the necessity to change from the K-Means to C-Medoids [11]. The algorithm initialization is done using the k-means++ algorithm [21] except when said otherwise.

III. Finding the optimal number of clusters

Throughout the years many different methods have been used to measure the validity and quality of a cluster classification [22]. Some of them are: the Davies-Bouldin (DB) index [23], which calculates a score that measures the biggest similarity between two clusters (from the idea that it is desirable that each cluster be as different as possible from the others); the rand index, which measures the changes in classification for pairs of elements according to the number of clusters used [24]; and the gap statistic, that compares the within-cluster dispersion from a certain clustering operation with what would be expected from a null reference distribution [25].

To determine the optimal number of behaviors observed in the clustering of the longitudinal information, the Davies-Boudin index was chosen mostly because it takes into account the distances between elements inside a cluster and the inter-cluster distances as well, without the need of any further information. The gap statistic needs a null distribution to compare to and the rand index needs the ground truth for its calculation (which for the maneuver detection is possible but not the case for the longitudinal detection).

A. Davies-Boudin score

The DB index is defined based on the dispersion of a cluster C_i, indicated as s_i, and the dissimilarity between two different clusters, namely d_{ij} [22]. The calculation is displayed in equation (7), where n_c refers to the number of clusters being used.

$$\mathrm{DB}_{n_c} = \frac{1}{n_c} \sum_{i=0}^{n_c} \left[\max_{j=1,\ldots,n_c, i \neq j} R_{ij} \right] \tag{7}$$

$$R_{ij} = \frac{(s_i + s_j)}{d_{ij}} \tag{8}$$

In this case, the dispersion s_i within each cluster is calculated by the sum of the DTW distances between the DBA centroid and each element, divided by the number of pairs detected. The distance between clusters is obtained by the DTW distance from the respective DBA centroids.

IV. Clustering procedure

The procedure is based on three steps: first, executing the clustering using the trajectory position coordinates, for each data-batch in each scenario present in the dataset; second, each obtained cluster previously, which represents an expanded maneuver, are unified into a single cluster for each expanded maneuver; and third, the clustering using the longitudinal velocity and acceleration is done on the respective clusters from step two.

A. Position clustering

The K-means clustering can be executed on the trajectories directly, which is costly in computational terms. One way to speed up the calculation is to execute the clustering operation using a row from the dissimilarity matrix, defined by 9, where n indicates the number of trajectories being considered

and the $d_{i,j}^{DTW}$ represents the DTW distance between the elements indexed as i ad j.

$$\mathcal{D}_{DTW} = \begin{bmatrix} d_{0,0}^{DTW} = 0 & \cdots & d_{0,n}^{DTW} \\ d_{1,0}^{DTW} & \cdots & d_{1,n}^{DTW} \\ \vdots & \ddots & \vdots \\ d_{n,0}^{DTW} & \cdots & d_{n,n}^{DTW} = 0 \end{bmatrix} \tag{9}$$

Executing the K-Means onto one dimension instead with having to recalculate distances to the current centroid in an iteration vastly decreases the computational time in comparison with the alternative. Algorithm 1 shows the entire procedure.

Algorithm 1: Clustering the DTW dissimilarity matrix

Input : $\mathcal{D}_{DTW}$; nk, number of clusters
Output: $clust$, set of clusters

```
1  counter = 0;
2  c_arr = [from 0 to length(D_DTW)];
3  for counter < n_cores - 1 do
4      best_idx = −1, min_inert = 0;
5      for each c_idx in c_array do
6          labels, centers = kmeans(D(c_idx, :), nk);
7          min_idx = argmin(μ_idx);
8          if centers[min_idx] ≤ min_inert then
9              best_idx = min_idx;
10             min_labels = labels;
11             min_inert = centers[min_idx];
12         end
13     end
14     for each elem in min_labels do
15         if elem == best_idx & idx ∈ cand_idx then
16             clust[counter] ← idx;
17             cand_idx.pop(elem);
18         end
19     end
20     counter++;
21 end
```

Considering that each line (row or column) from $\mathcal{D}_{DTW}$ represent the comparison of one trajectory with all the others, line 2 establishes the index array of all trajectories. For each element still being considered, the respective line of the dissimilarity matrix is clustered (line 6) to obtain the labels (which allocates each element of the row to a cluster) and the centers of each cluster (which will be the average difference between elements on the cluster). Then, using the classification with the smallest center possible (line 8) all the elements of the same cluster are grouped in the final structure (line 16) and removed from consideration on the next round (line 17). The same procedure could be done sequentially according to the order of the matrix $\mathcal{D}_{DTW}$, which would be faster. However, iterating over all elements results in a more robust result, avoiding any initialization problems. Finally, the last cluster receives the remaining data still in c_idx.

The operation in algorithm 1 is repeated for different numbers of cores, and the optimal choice is determined by the minimal DB index calculated from all those clustering operations. One of the possible problems of this method is the calculation of the dissimilarity matrix, which is $O(N^2)$. This can be mitigated by splitting a bigger dataset in smaller ones and then using the next method to unify all the results.

B. Unifying the maneuvers from each data-batch

This process actually allows all data-batches from a single scenario to be bundle together while it also mitigates the $O(N^2)$ problem of calculating the dissimilarity matrix. In consequence, one can cluster using small sets of data during the position clustering and then bring them together in clusters without repetition. Having all the data batches clustered, their centroids can be compared with each other using the DTW distance. Defining a simple difference threshold is enough in the following case to add data to its analogue cluster or to establish a new one, if none are similar.

C. Longitudinal clustering

Now having all the trajectories classified into maneuvers, the longitudinal set of data from each maneuver can be recovered and clustered into different behavioral profiles. This is done by recovering the longitudinal velocity and acceleration for each element on the cluster: for a determined number of cores, execute the clustering and determine the optimal cluster number by minimizing the DB index calculated.

Algorithm 2: Longitudinal maneuver clustering

Input : long_data, nx2 matrix
Output: $\text{lab}_{opt}, \mu_{opt}$, optimal labels and centers

```
1  for nk < n_cores do
2      lab_nk, μ_nk = kmeans(long_data, nk);
3      db_result ← DB(lab_nk, μ_nk, long_data);
4  end
5  best_idx = argmin(db_result);
6  lab_opt, μ_opt ← lab_best_idx, μ_best_idx
```

The method of clustering the dissimilarity matrix was tested with the longitudinal data as well, but it did not produce adequate results because the differences between each element can be too small to be significant in the final distance. The longitudinal velocity is contained around a positive finite range in this case between $20m/s$ and 0 and acceleration fluctuates around zero, differently to the position that usually do not have a common baseline. Hence, the DTW distances calculated for the position of different trajectories are clearly different, but not so much for the longitudinal data of the same expanded maneuver.

V. Results and Discussion

A. Methodology

All data-batches and scenarios from the InD dataset [13] (7 for scenario 1, 10 for scenario 2, 12 for scenario 3 and 3 for scenario 4) were used to obtain the results displayed

in the figures 5, 6 and 7 and table II. The only alteration done was to remove the following trajectories: first filter all those related to pedestrians, cyclists or trucks-buses; second, to eliminate incomplete traces, notably if they had already started before the beginning of the data recording or if they did not end before its end. Also, there was some trajectories inside the dataset that suffered a tracking loss during their calculation and were incomplete. All those trajectories were not used in the clustering presented in V-B.

Table I shows the number of clusters explored during the position and longitudinal clustering to determine the best number of clusters. These ranges were determined by visual inspection for the position clustering, according to the possible maneuvers in the map and defined empirically for the longitudinal clustering. Since there was no indication as to how many clusters might exist for the longitudinal clustering, a wider search range was used.

TABLE I: Number of clusters explored

Clust. type	Sce. 0	Sce. 1	Sce. 2	Sce. 3
Position	7 ± 2	10 ± 3	10 ± 3	8 ± 3
Longitudinal	11 ± 9			

All the calculations were done in a Dell Inspiron 5570 notebook with a Intel Core i9-12900H containing an NVidia Quadro A2000 and using the functions available from the Python library TSlearn [26].

B. Results

1) Position clustering: Applying the algorithm 1 for each data-batch available in the dataset results into clusters similar to figures 3 and 4, which are clusters for scenario 0 (not using the same color code as in figure 5; the cluster numbers are not related to the maneuvers defined in the next subsection).

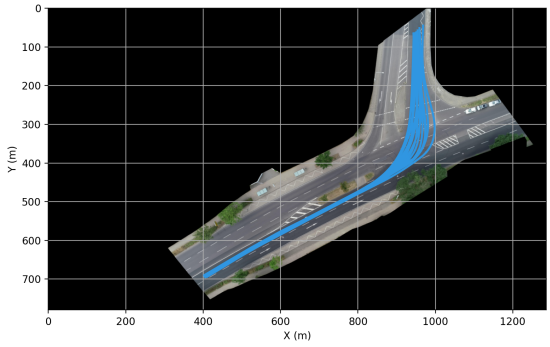


Fig. 3: Cluster 0 from data batch 2 of scenario 0

All the modifications discussed in the methodology section allow the algorithm 1 to be robust enough to cluster all the trajectories in the dataset without mistakes. Table II shows the difference in execution time between the default clustering method and the dissimilarity method usage proposed in algorithm 1. The timings were calculated as the average time for the clustering of all cores displayed in table I and all data-batches for each scenario, in one pass only. The

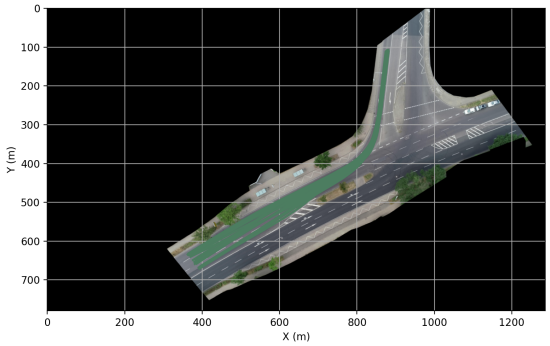


Fig. 4: Cluster 2 from data batch 2 of scenario 0

standard deviation in table II refers to the time differences in clustering each data-batch, that have different number of trajectories.

Given the performance difference by calculation time, the dissimilarity matrix method is superior. However, the error rate was negative because the direct K-Means approach miss-classified some trajectories while the dissimilarity method did not. This happens due to the initialization of the direct method, which might start with the wrong trajectories in a cluster and throughout the clustering might not be able to shift to the correct ones (being stuck in a local minima).

TABLE II: Average position clustering time per data-batch

Scenario	Dissi. method	Direct Kmeans	Error rate
0	$31.48s \pm 13.69s$	$831.16s \pm 572.19s$	0%
1	$26.85s \pm 14.45s$	$267.34s \pm 167.08s$	-1.7%
2	$32.72s \pm 09.88s$	$257.28s \pm 098.87s$	-0.53%
3	$57.08s \pm 19.54s$	$682.91s \pm 290.26s$	-0.37%

As it can be seen there is a real gain using the dissimilarity method, even if it is necessary to calculate the dissimilarity matrix before the clustering and the clusters' centers afterwards (which is included in the calculated times). As said before, this method works as well as the direct k-means method, if not better, because all the maneuvers are distinct enough for the DTW distance to have a clear difference between clusters. This is not the case for the longitudinal velocity and acceleration, for example, where the division between maneuvers was already done, thus the differences are rather subtle and difficult to cluster.

2) Unifying data-batches: Following the details given in subsection IV-B, figures 5, 6, 7 and 8 represent all the trajectories detected for the first three intersections present in the InD dataset. The union of clusters determined earlier represents the maneuvers discovered in each scenario.

The only drawback of the unifying method presented earlier is that a threshold needs to be established based on the DTW distance score between two cluster centers. In the implementation a single value could be used for all scenario but there is no guarantee that one can be found every time. The ease of finding the correct threshold value depends on how different are the clusters being considered, which is the same characteristic that allows the dissimilarity matrix

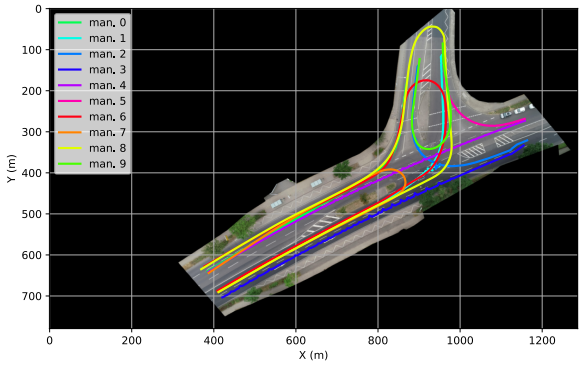


Fig. 5: Maneuvers for scenario 0 from InD dataset

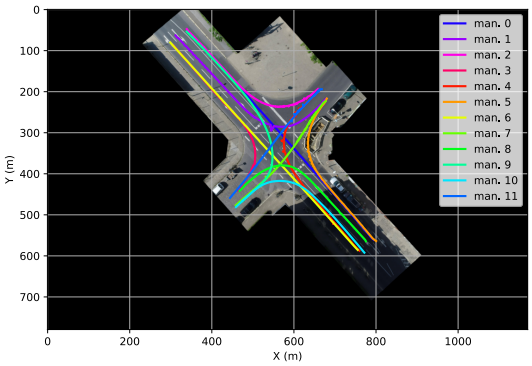


Fig. 6: Maneuvers for scenario 1 from InD dataset

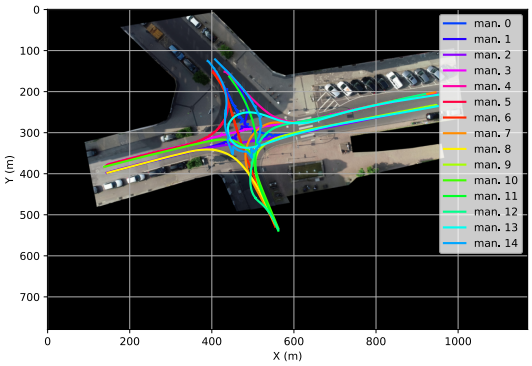


Fig. 7: Maneuvers for scenario 2 from InD dataset

clustering to give the correct results.

3) Longitudinal clustering: Given that some maneuvers have a small number of samples, a minimal threshold of 10 trajectories was adopted to execute the clustering (in red on table III). Also, a restriction of the number of clusters tested according to the number of trajectories was used: $\min(20, n_{\text{traj}}/2)$.

As initialization, the algorithm 1 was used to select the cluster seeds instead of the usual k-means++ in the following way: from the clusters obtained with the dissimilarity matrix calculated with the longitudinal data, one sample was randomly selected from each cluster to serve as initial seed. Table III gives a complete overview concerning the number of samples per trajectory in each of the scenarios available.

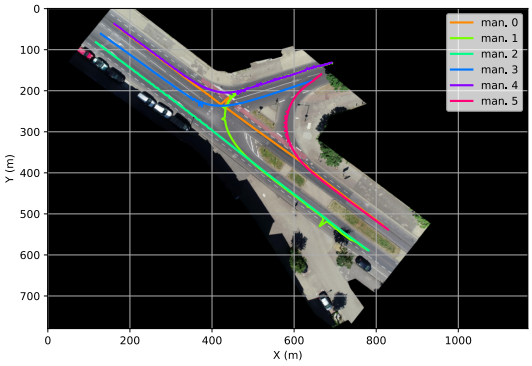


Fig. 8: Maneuvers for scenario 3 from InD dataset

TABLE III: Number of trajectories per maneuver

Man.	Sce. 0	Sce. 1	Sce. 2	Sce. 3	Traj.	Sce. 0	Sce. 1	Sce. 2	Sce. 3
0	199	712	245	405	8	1	21	15	-
1	146	231	162	37	9	1	13	19	-
2	21	118	724	452	10	-	12	7	-
3	609	9	357	85	11	-	20	14	-
4	803	216	324	47	12	-	-	2	-
5	21	257	133	44	13	-	-	4	-
6	2	662	21	-	14	-	-	4	-
7	2	19	20	-	Total	1805	2290	2051	1070

To explain the results obtained from the clustering of the longitudinal data, figure 9 displays the average velocities and 10 shows the average acceleration for the maneuver 1 of scenario 0 (which are the data from the maneuvers in figure 3). In this case the optimal number of clusters, hence behavior profiles, detected was 12. It can be seen that there is a stratification of velocity between cluster centers, with two of them reaching the velocity equal to zero in the middle of the maneuver (clusters 1 and 2), represented by figures 11a and 11c respectively and other two others that did not interact with anyone (clusters 3 and 4), represented by figures 11b and 11d.

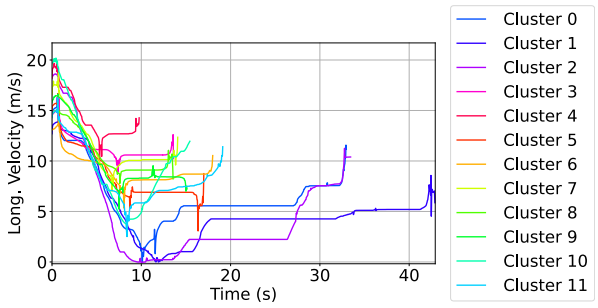


Fig. 9: Average velocities for maneuver 1 of scenario 0

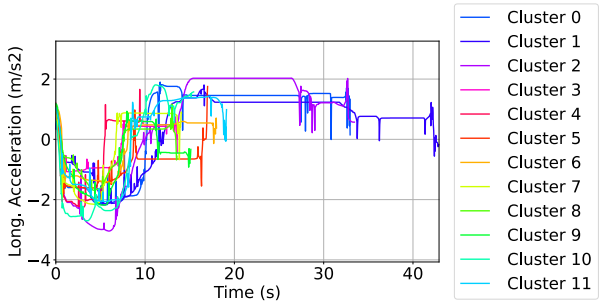
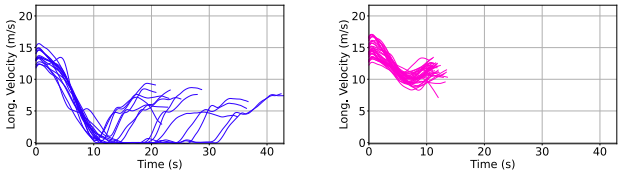
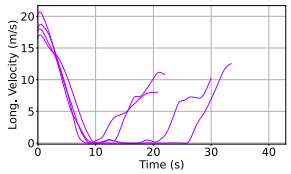


Fig. 10: Average accelerations for maneuver 1 of scenario 0

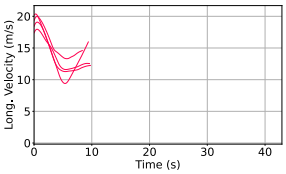


(a) Behavior profile 1

(b) Behavior profile 3



(c) Behavior profile 2



(d) Behavior profile 4

Fig. 11: Four behavior prof. for maneuver 1 from scenario 0

Comparing the four plots in figure 11, it can be said that some clusters represent the vehicles that interacted with other road users and some represent those that did not. Moreover, the difference between 11a and 11c and between 11b and 11d can be explained by the starting velocity (from 20 to 15 m/s for the former and 15 to 10 m/s for the latter), since the turning velocity (at the middle of the trajectory) is almost equivalent (around 10m/s).

The other clusters, not shown here, represent velocity profiles that indicate different levels on interaction with other road users, some with higher starting velocities and/or with higher decelerations, to wait other road users or just to execute the turn more lowly. The results from all the scenarios point to the conclusion that two of the most determinant factors to explain the behavior profiles obtained are the aggressiveness of the driver (accounts for deceleration and the starting velocity) and if and how it interacted with other road users during the trajectory. Therefore, classifying each behavior profile with these two parameters can represent the entire range of behavior profiles for an intersection.

However, there is no right answer about the number of clusters, differently from the position clustering case. Figure 12 shows the DB score curve, and if one uses the minimal approach suggested in algorithm 2 the optimal number of clusters should be 18, not the 12 shown in figures 9 and 10. This is because for higher number of clusters the DB score reach the realm of diminished returns, where only a few trajectories might get relegated to one cluster, decreasing the dispersion and thus the score itself.

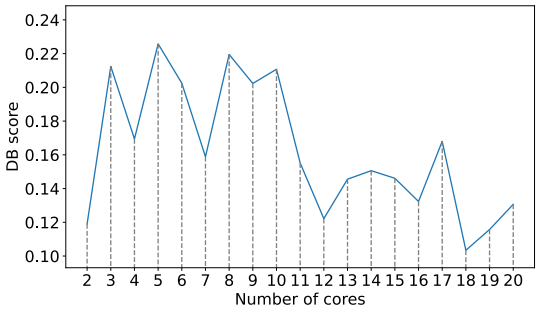


Fig. 12: DB scores for the test number of clusters

Using the k-means++ algorithm to seed the initial clusters (not the usual dissimilarity matrix method) and 5 repeated calculations per cluster (to avoid any initialization local minima) results in even lower DB scores, with a cost due to the calculation time (10 times higher than with the dissimilarity method as seed initialization). But, in this case, the optimal number of clusters is two, simply separating the trajectories that do interact and those that do not. The use of k-means++ selects two dissimilar seeds for initialization which, connected to the fact that the average calculated by the DBA is equal in extension to the biggest series, reduces the DB score in comparison with higher number of clusters. In the dispersion calculation for equation (7) the value is normalized using the number of pairing between each element and the average (the number of times equation (6) is executed), therefore artificially reducing the score.

Another problem with the DTW calculation for longitudinal data happens in the comparison of large periods where the velocity is equal to zero. In this case a big part of the trajectory might produce a distance equal to zero, thus keeping together trajectories that should form different clusters. It is what happens in figure 11a and 11c where curves that spent different amounts of time in zero should have been in different clusters.

Ideally, for the proposed use as input to train a simulation agent, it should exist a separation on the range of velocities displayed by the trajectories representing a maneuver as well. Thus, considering all these elements, the results displayed in table IV comes from the manual selection of the DB scores, taking into account still the local minima on the DB curve, but trying to obtain profiles that separate on the longitudinal velocity as well. The results in red represent an insufficient number of trajectories to apply the longitudinal clustering.

VI. Conclusion and Next Steps

It is essential for realistic simulations to have independent agents that behave and interact with each other in a natural way. The main goal of this article is to extract behavior profiles from real trajectories to, first, understand which kind of behaviors might arise from different maneuvers, and second, to be able to, in a later iteration, generalize over

TABLE IV: Discovered behaviors per scenario

Man.	Sce. 0	Sce. 1	Sce. 2	Sce. 3	Traj.	Sce. 0	Sce. 1	Sce. 2	Sce. 3
0	15	12	10	9	8	No	4	4	-
1	12	10	12	6	9	No	4	2	-
2	8	5	5	14	10	-	4	No	-
3	9	No	14	12	11	-	4	6	-
4	11	11	14	9	12	-	-	No	-
5	5	12	12	10	13	-	-	No	-
6	No	15	6	-	14	-	-	No	-
7	No	5	7	-					

such behaviors and reproduce interactions and trajectories with high fidelity.

The extraction of behaviors was done in a sequence of two clustering operations. First, the position coordinates of each trajectory as clustered to determine the existing maneuvers in one data-batch. This clustering operation was accelerated by using DTW dissimilarity matrix itself (algorithm 1) rather than the entire trajectories. Second, the detected maneuvers for all the data-batches in one scenario were fused together and clustered as a maneuver using the longitudinal velocity and acceleration, to determine the behavior profiles that might exist.

The next steps can be divided in three tasks: to test the algorithm with the pedestrian and cyclist data, not used here, and also to explore new datasets; test different dissimilarity measures and optimal cluster selection criteria for the longitudinal case; and finally to train the aforementioned agents in a simulation environment based on the profiles obtained.

REFERENCES

[1] O.-R. A. D. O. c. SAE, "Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles," USA, Standard, 06 2018.

[2] E. Vidal Ruiz, F. Casacuberta Nolla, and H. Rulot Segovia, "Is the dtw "distance" really a metric? an algorithm reducing the number of dtw comparisons in isolated word recognition," *Speech Communication*, vol. 4, no. 4, pp. 333–344, 1985. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0167639385900585

[3] E. Vidal, F. Casacuberta, J. M. Benedi, M. J. Lloret, and H. Rulot, "On the verification of triangle inequality by dynamic time-warping dissimilarity measures," *Speech Communication*, vol. 7, no. 1, pp. 67–79, 1988. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0167639388900222

[4] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *5th Berkeley Symp. Math. Statist. Probability*, 1967, pp. 281–297.

[5] F. Itakura, "Minimum prediction residual principle applied to speech recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 23, no. 1, pp. 67–72, 1975.

[6] H. Sakoe, "Dynamic-programming approach to continuous speech recognition," in *1971 Proc. the International Congress of Acoustics*. Budapest: ICA, 1971.

[7] C. A. Ratanamahatana and E. Keogh, "Everything you know about dynamic time warping is wrong," in *Third workshop on mining temporal and sequential data*, vol. 32. Citeseer, 2004.

[8] T. Warren Liao, "Clustering of time series data—a survey," *Pattern Recognition*, vol. 38, no. 11, pp. 1857–1874, 2005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320305001305

[9] M. Y. Choong, L. Angeline, R. K. Yin Chin, K. Beng Yeo, and K. T. Kin Teo, "Modeling of vehicle trajectory clustering based on lcss for traffic pattern extraction," in *2017 IEEE 2nd International Conference on Automatic Control and Intelligent Systems (I2CACIS)*. IEEE, 2017, pp. 74–79.

[10] B. T. Morris and M. M. Trivedi, "Trajectory learning for activity understanding: Unsupervised, multilevel, and long-term adaptive approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 11, pp. 2287–2301, 2011.

[11] R. G. Ribeiro and R. Rios, "Temporal gap statistic: A new internal index to validate time series clustering," *Chaos, Solitons & Fractals*, vol. 142, p. 110326, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0960077920307219

[12] S. Choi, J. Kim, and H. Yeo, "Trajgail: Generating urban vehicle trajectories using generative adversarial imitation learning," *Transportation Research Part C: Emerging Technologies*, vol. 128, p. 103091, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X21001121

[13] J. Bock, R. Krajewski, T. Moers, S. Runde, L. Vater, and L. Eckstein, "The ind dataset: A drone dataset of naturalistic road user trajectories at german intersections," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 1929–1934.

[14] M. Müller, *Information retrieval for music and motion*. Springer, 2007, vol. 2.

[15] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 43–49, 1978.

[16] M. Morel, C. Achard, R. Kulpa, and S. Dubuisson, "Time-series averaging using constrained dynamic time warping with tolerance," *Pattern Recognition*, vol. 74, pp. 77–89, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S003132031730328X

[17] M. Cuturi and M. Blondel, "Soft-dtw: A differentiable loss function for time-series," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17. JMLR.org, 2017, p. 894–903.

[18] M. Brankovic, K. Buchin, K. Klaren, A. Nusser, A. Popov, and S. Wong, "(k, l)-medians clustering of trajectories using continuous dynamic time warping," in *Proceedings of the 28th International Conference on Advances in Geographic Information Systems*, ser. SIGSPATIAL '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 99–110. [Online]. Available: https://doi.org/10.1145/3397536.3422245

[19] F. Petitjean, A. Ketterlin, and P. Gançarski, "A global averaging method for dynamic time warping, with applications to clustering," *Pattern Recognition*, vol. 44, no. 3, pp. 678–693, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S003132031000453X

[20] G. Forestier, F. Petitjean, H. A. Dau, G. I. Webb, and E. Keogh, "Generating synthetic time series to augment sparse datasets," in *2017 IEEE International Conference on Data Mining (ICDM)*, 2017, pp. 865–870.

[21] D. Arthur and S. Vassilvitskii, "K-means++ the advantages of careful seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, 2007, pp. 1027–1035.

[22] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "On clustering validation techniques," *Journal of Intelligent Information Systems*, vol. 17, no. 2, pp. 107–145, Dec 2001. [Online]. Available: https://doi.org/10.1023/A:1012801612483

[23] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 224–227, 1979.

[24] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical Association*, vol. 66, no. 336, pp. 846–850, 1971. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/01621459.1971.10482356

[25] R. Tibshirani, G. Walther, and T. Hastie, "Estimating the number of clusters in a data set via the gap statistic," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 2, pp. 411–423, 2001. [Online]. Available: https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/1467-9868.00293

[26] R. Tavenard, J. Faouzi, G. Vandewiele, F. Divo, G. Androz, C. Holtz, M. Payne, R. Yurchak, M. Rußwurm, K. Kolar, and E. Woods, "Tslearn, a machine learning toolkit for time series data," *Journal of Machine Learning Research*, vol. 21, no. 118, pp. 1–6, 2020. [Online]. Available: http://jmlr.org/papers/v21/20-091.html