Hybrid Control and Learning with Coresets for Autonomous Vehicles

Guy Rosman[1], Liam Paull[1] and Daniela Rus[1]

***Abstract*— Modern autonomous systems such as driverless vehicles need to safely operate in a wide range of conditions. A potential solution is to employ a *hybrid systems* approach, where safety is guaranteed in each individual mode within the system. This offsets complexity and responsibility from the individual controllers onto the complexity of determining discrete mode *transitions*. In this work we propose an efficient framework based on recursive neural networks and coreset data summarization to learn the transitions between an arbitrary number of controller modes that can have arbitrary complexity. Our approach allows us to efficiently gather annotation data from the large-scale datasets that are required to train such hybrid nonlinear systems to be safe under all operating conditions, favoring underexplored parts of the data.**

We demonstrate the construction of the embedding, and efficient detection of switching points for autonomous and non-autonomous car data. We further show how our approach enables efficient sampling of training data, to further improve either our embedding or the controllers.

I. Introduction

One key challenge facing real-world, safety-critical autonomous systems is the requirement for robust performance over a huge variety of environmental and operating conditions. This is particularly true for autonomous driving. Fully autonomous ("Level 5") driving is defined as when "the automated system can perform all driving tasks, under all conditions that a human driver could perform them" [26].

To handle road diversity and low error tolerance, we wish to better enable autonomous vehicles to adjust their driving style to the surrounding conditions. For example, driving on a congested urban center with unpredictable pedestrians and cars requires a much more conservative driving style than driving on a straight highway in the desert, and this is in turn different than driving on a winding narrow mountain, on a rainy night. Instead of one single autonomy solution, we could leverage the problem structure via a hybrid control approach [3],[21], by having library of autonomy solutions tuned for different driving situations. We call each potential solution a "driving mode", or system mode in the context of more general robotic systems. In this paper we develop a method for perception-driven matching of driving modes to environments. We efficiently learn from the visual data a representation for the driving modes and efficiently detect transitions between modes.

* Support for this work was given by the Toyota Research Institute (TRI). However, note that this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the DGX-1 used for this research. The authors wish to thank Andrea Censi for the Duckietown data.

[1]Authors are with the Computer Science and Artificial Intelligence Lab, Massachusetts Institute of Technology, Cambridge MA 02139, USA
rosman,lpaull,rus@csail.mit.edu

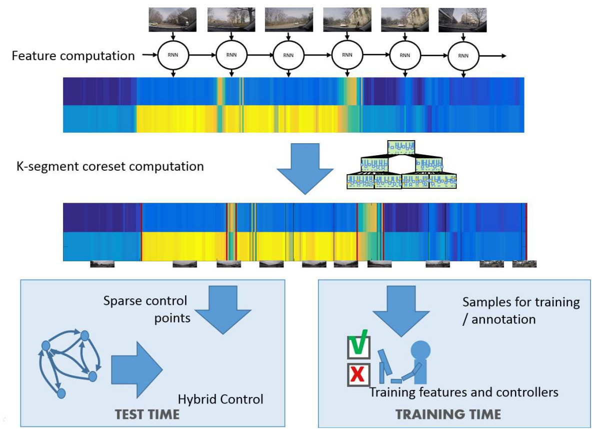

Feature computation
RNN
K-segment coreset computation
Sparse control points
Hybrid Control
TEST TIME
Samples for training / annotation
Training features and controllers
TRAINING TIME

Fig. 1. Overview of our method: RNNs are used to approximate possible control changes via an embedding space. The embedding vectors stream is then processed by a streaming coreset. Coreset segment transitions provide control points for controller switching, and coreset sampling allows selection of samples for further training and verification.

A key insight is that mode selection can often leverage the sensory history of the agent, in a conservative manner. We make minimal assumptions about the individual driving modes: each mode is able to output some measure that represents its estimate regarding the safety of its output (for example, probability of pedestrian misdetection).

Two key questions arise:

Q1: How to safely and efficiently switch modes given the system's sensory history?

Q2: How to leverage the massive amounts of data collected by a modern autonomous car to efficiently train the system, given that it is very costly to generate *negative* samples (failures).

To address these issues, we leverage: deep learning approach using recurrent neural networks [14] to capture the visual and temporal structure of the problem, and to encode the mode in a low-dimensional embedding space, and k-segment mean coresets [31], [36] for efficient data summarization, change detection, and querying. An overview of the method is shown in Fig. 1.

Specifically, since the sensory input of the car may be insufficient to efficiently determine the best controller, we shape the encoding mechanism so that the transient behaviors captured by the coresets help us determine critical points of possible mode changes. This allows us to limit the computations of mode identification by querying all modes for feasibility. Coreset computation over the encoded stream also allows us to query the data to obtain harder cases for training in a way that performs well with stochastic descent training methods.

We demonstrate this approach using driving data from Duckietown [28], a scaled-down prototype of an urban center, as well as driving data using over 60 hours of driving through different landscapes that transition between dense urban, suburban, and rural environments, as well as parking lots.

The paper contributes:

1) An encoding approach for the detection of possible driving mode switching based on the robot's recent history, using deep learning, RNNs and k-segment mean coresets.
2) Use of this encoding to improve the safety-vs-efficiency tradeoff (addressing *Q1*).
3) Use of coresets for life-long datasets where labeling is an expensive bottleneck (addressing *Q2*).

The remainder of the paper is structured as follows: we summarize the related literature in Sec. II, we describe the assumptions and the model in Sec. III, describe how this model can be used for control and for learning in Sec. IV, and demonstrate results in Section V.

II. Related Works

Our work ties into several fields of active research. Hybrid systems have been used for a long time in control, from multiple linear controllers systems [3], through to more general theory [22]. Moreover, while we use autonomous cars as our main motivating example, a hybrid systems approach is relevant for many other robotic systems, such as the humanoid robots used in the DARPA robotics challenge [8] or for quadrotors performing complicated aerial manuevers [13]. However, we are focused on application cases where the environment and task are suffeciently complex that it is hard to generate the switching functions from simple heuristics or close-form solutions. For example, in the case where linear control is sufficient, the optimum control law can be chosen in an online fashion [3]. However, as agents' tasks and environment become more complex, deciding the optimal mode is more challenging and requires significant resources. Autonomous driving is a clear example for this, containing a vast array of potential scenarios and operating conditions.

We are not placing any assumptions or limitations on the choice of control, perception and planning algorithms that are operating in each mode. For example, individual control modes could employ an end-to-end approach as in [38], [5], allowing simpler individual networks that are working in harmony over a broader set of conditions, with our system sitting at a higher level to arbitrate them. Other approaches are more specific and structured, such as lane following [30], platooning [4], or other modes. Another important automotive example is the bimodal case that combines an human driver and an autonomy solution. Some recent works have focused on defining the switching function between human and autonomous system modes through a minimal intervention approach. For example, formal methods approaches are able to provide rigorous theoretical guarantees that a dynamical system (e.g. an automobile) will not crash by ensuring that it does not enter the set of states that will result in crashes (referred to as inevitable collision states [12], capture sets [10], [16], or target sets [23]). These sets tend to be very costly to compute in all but the simplest real-world scenarios, often reasoning about geometry alone, and ignoring sensor limitations and complicated uncertainty models. An alternative approach is to consider this as a shared control problem and minimize the discrepancy between the human and autonomous system input subject to safety constraints [2], [32]. However, solving these optimizations can also be costly in practice, and may not generalize well to other system modes. Here, we are considering a more general formulation, where there are N control modes and we are using sensory inputs to select which is the most appropriate mode for any given scenario.

As our decision takes as input the system's sensory history, we need an efficient representation for this vast space that leverages its invariance and structure. In recent years, learning approaches such as RNNs [14] and Long-Short-Term-Memory (LSTM) networks [15] have proven very useful for audio, video and other temporal or spatial signals. In our work, we use RNNs as our function approximator, but other temporal architectures are also possible. The link between deep learning features and embedding spaces has been considered before in the neural network literature, see for example [18].

Finally, as our work deals with efficient modeling of piecewise behavior, it ties in to a large work of probabilistic reasoning over switching system [11], [24], [19], [27]. However our problem is a co-design problem, where we both design the embedding space, and track variations in time.

A. *k-Segment Mean Coresets*

We leverage k-segment mean coresets for temporal signal summarization [31]. These coresets have found numerous uses for localization [36], tracking [7], and medical video analysis [35]. Coresets techniques in general facilitate big-data analysis for a variety of clustering and data modeling problems. Coresets are constructed to represent the data by a subset of elements, selected such that computing a function on the reduced data returns approximately the same result as the original data, with provable approximation bounds. See for example [1], [31], [29] and references therein.

k-segment mean coresets approximate a temporal stream of high-dimensional vectors by a set of linear representatives, with respect to the l^2 norm fitting of any k-segment. The coreset is computed in streaming mode, where for each block of data 1) the data complexity bound is estimated via the *Bicriteria* algorithm. Then 2) the data is divided into a fine partition of segments (coreset segments) via the *BalancedPartition* algorithm, dividing the data according to on-line fitting error estimation. See [31] for additional details. Aside from efficiently computing segmentation, k-segment mean coresets have proven useful for reasoning about lifelong visual streams [36]. Utilizing a coreset allows us to both efficiently detect changes, and perform retrieval and sampling from points of interest in the stream.

III. Model Definition

We assume a hybrid controller for the system, and we are collecting data in order to optimize it. We denote by $x(t)$ and $o(t)$ the state and observation at time t, taken respectively from state space $\mathcal{X}$ and observation space $\mathcal{O}$. $o(t)$ denotes the observation history at time t, with some finite horizon. We assume a hybrid system that includes individual modes $\{C_i\}_{i=1}^{C}$. While C isn constant, in a life-long learning setting, we may discover additional modes that should be added. Our design accomodates such an eventuality – a good arbitration design should be robust to minor controllers additions or removals, and not require processing of the existing data.

In order to ensure safe execution in a generic setting, we assume there is always a system mode that is safe to use, and that each mode can reason about whether it is safe to use at time t. We assume safety of each mode i is a binary-valued function of the state, $y_i : \mathcal{X} \to \{-1, 1\}$. Since the state is not directly observable, and safety regions are mode-dependent, we assume the system mode defines a function $g_i(o(t))$ that estimates based on recent observations and the robot state.

Assumption 1: Each system mode is safe to use in some region of state-space.

Assumption 2: For each state x there is at least one safe system mode.

Assumption 3: Each system mode outputs a function g_i that represents its confidence relative to its output.

Controllers are designed with an operational envelope, and with a good estimation of when the control state may exceed that envelope. For example, an inference engine may be able to output an information-theoretic measure in addition to a belief (e.g., [17]).

However, computing this function may be costly as it requires processing of a significant history, for example dependencies on previously seen traffic signs, or even telling apart road lanes on a rainy night. We therefore prefer to be judicious about the computation of $g_i(o(t))$. We define an embedding [6], [25] $f(o(t))$ such that changes in $g_i(o(t))$ (for any i) are captured by changes in $f(o(t))$. In other words, for every pair of time points t_j, t_k, $\|f(o(t_j)), f(o(t_k))\|^2$ should be small if $y_i(x(t_j)) = y_i(x(t_k))$ for every controller i, and sufficiently large otherwise. Given such a mapping f, jumps in $f(o(t))$ correspond to switching in controller safety, and should be examined – these are *transition candidates* and no other time points would require such checks.

This view leads to an embedding criteria similar to metric learning [37]. While many metric learning approaches can be used, we chose a penalty formulation

$$F(\Theta) = \mathrm{E}_{(t_j,t_k)}\, s_i(t_j, t_k) \|f_\Theta(o_{t_j}) - f_\Theta(o_{t_k})\|^2 \qquad (1)$$

where f is parameterized by a vector Θ, denoted as f_Θ for emphasis. $s_i(t_j, t_k) = \mathrm{sign}\left(g_i(o_{t_j}) g_i(o_{t_k})\right)$ is a ± 1 indicator whether observation histories t_j, t_k agree on the safety of controller i. Here we use $g_i(o(t))$ as y_i is never available – we assume that g_i must be perfected to approximate y_i to ensure safety. We use g_i as a surrogate for y_i when running the system. It is the task of a system designer to have sufficient examples of state and observations history to improve both the controller and the surrogate function g_i. This includes as few as possible off-policy examples [33], for safety reasons (since these require actually causing the system to fail). Detecting the changes in the temporal stream $f(o(t))$ allows us to compute $g_i(o(t))$ sparingly, and to reason about places where we are not sure about which controller to employ. Reasoning about safety and collision sets, especially in the presence of other agents, and partial sensor models, is a field in itself, and is beyond the scope of this paper. We note that the dimensionality of the embedding space need not equal the number of sub-controllers – in fact, we do not need to assume a fixed number of sub-controllers. During system construction we may add sub-controllers, and yet we do not wish to recompute the embedding space from scratch.

Next, we require a way to reason about the changes in $f(o(t))$. The embedding f is important in two respects. First, it allows us to detect transitions between modes. Secondly, its behaviour allows us to reason about possible points of failure from the data, and guide sampling of examples to improve both f and the controller of each system mode. From a safety perspective, we need to handle the large scale data required to train such controllers over diverse environments and with small margins of errors, requiring the right tools.

In order to represent the data in a way that affords temporal reasoning for both short and long time spans, we feed the the stream $f(o(t))$ into a k-segment means coreset to obtain a representation that enables such efficient reasoning about the stream. See Subsection II-A for more details about this specific structure. Utilizing the coreset enables detection of possible changes in controllers, while minimizing the number of times $g_i(\cdot)$ needs to be computed in order to decide on a controller change, addressing (Q1). It also enables easy collection of time points that should be important for training, such as transition and uncertainty region, addressing (Q2). This is all done with minimal assumptions on the underlying controllers set, and while scaling to infinitely large datasets by sampling.

While storage and computations become increasingly cheap, accurate training feedback is still costly for complex robotic systems such as cars. Unlike standard reinforcement learning literature, in autonomous driving the cost of policy failure are often unacceptable. This requires indirect approaches such as training with human annotation, simulators, or other approaches. While semi-supervised approaches to controller training can help [9], there is still a significant cost for every supervised training point, and these must be chosen judiciously from the data. Unlike the standard stationarity assumed in stochastic training methods, we want to be able to sample from the data while giving preference to cover possible failure cases when the system is uncertain or transitions occur. Using k-segment mean coreset allows us to do so, in an efficient manner.

IV. Methods

We now demonstrate two algorithms based on our approach. The first is a method to detect transitions at a smaller computational cost based on k-segment mean coresets, addressing (Q1). The second one allows training controllers in a more efficient way (with respect to required annotations), addressing (Q2).

A. An Embedding Approximation for Controllers

To detect transitions, k-segment mean coresets are constructed as piecewise approximations to the data, as part of the BalancedPartition algorithm [31]. The algorithm creates coreset segment bounds when a large change in the data is detected. It is those changes that we would like to capture. The coreset algorithm runs in streaming mode, a small latency (depending on coreset leaf size), and allows us to detect transition candidates in the data, as we describe in Alg. 1.

Algorithm 1 Coresets-based hybrid control

```
1: for Every new observation $o(t)$ do
2:     Obtain embedding vector $f_\Theta(o(t))$ for the time frame.
3:     Update the coreset stream in streaming mode [31]
4:     When a new coreset segment is created, compute
       $g_i(o(t))$ for all $i$, switch mode.
5: end for
```

For sufficiently large changes, these can be detected instantenously after computing f, with a verification from the coreset segments arriving within a short latency, allowing to rule out transient outliers. Such an instantenous approach is close in spirit to dead-reckoning algorithm over feature space, which was compared to k-segment mean coresets in [31] and found to result in a worse compression ratio. Our specific choice of embedding function f is a recursive neural network, as these allow us to capture features in the temporal visual history of the car, without committing to a specific temporal window. We use a transfer-learning approach, fine-tuning visual classification neural net (such as Googlenet [34]) by replacing the top classification layer with a linear layer, feeding into an 8-state RNN. An additional linear layer applied to the RNN's internal state forms f. The network topology is illustrated in Fig. 2. The parameters of the fine-tuning network define Θ in our case. While more complex network topologies can be used, we found this topology sufficient for all of our datasets, as we show in section V.

B. Coreset Sampling for Embedding Learning

Next, we show how to efficiently sample from the data in the limit of a large scale data streams such as the ones needed to detect and address rare yet dangerous events in the data. Coresets are approximation methods that summarize the data with respect to a family of analysis functions, representing it in a compact way. k-segment coresets allow us to represent an incoming data stream with guarantees on the quality of later segmentation analysis [31].

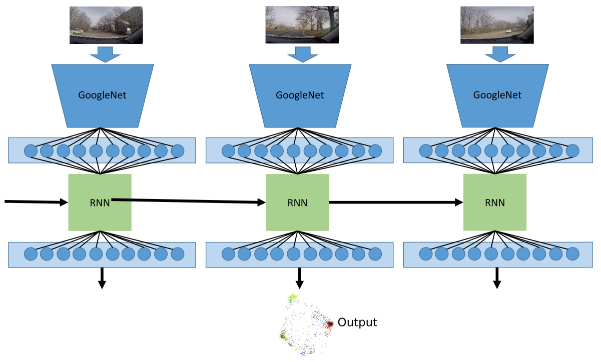


Fig. 2. The network topology: a trimmed GoogleNet, connected to a 10-neuron layer, is input into an RNN with a state of 8 elements. The RNN output is fed into another 10-neuron layer, and output as the embedding vectors.

Coreset segments are by construction segments of few changes in the data stream (for a piecewise-constant model). If the data is piecewise-constant, which is expected according to our optimization criteria in Eq. (1), the transitions between coreset segments should capture the transitions in the data. It is therefore of interest to sample these points in the stream more intensely.

In order to sample from the full data while favoring endpoints of coreset segments, we sample first a coreset leaf, then a segment within that leaf. Finally, we sample points within that segments with a Beta$(0.5, 0.5)$ distribution. This favors sampling with emphasis on transition points, as shown in Fig. 3.

Algorithm 2 Coreset sampling for metric learning

```
1: for Samples $s = 1, 2, \dots$ do
2:     Sample coreset segments $c_j, c_k$ based on coreset tree
       sampling ([36], Alg. 1)
3:     Sample time points $t_j, t_k$ from within $c_j, c_k$ using a
       Beta$(0.5, 0.5)$ distribution.
4:     Query (approximate) annotations for $y_{t_j}$, $y_{t_k}$
5:     compute $\nabla_\Theta f(\Theta), f(\Theta_)$, and gradients/values for $g_i$
       and their parameters.
6:     Update controller parameters and safety indicator
       function $g_i$ (controller-specific)
7:     Update $\Theta$ by stochastic optimization
8: end for
```

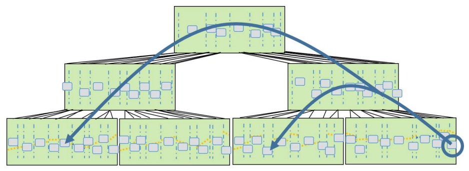


Fig. 3. An illustration of coreset tree sampling between coreset tree nodes. Random sampling is done from the last (rightmost) keyframe according to the tree edges, in a way that balances variability in the stream and in time, see [36] for details.

In our implementation we used ADAM [20] to optimize Eq. (1) with respect to Θ. However, other algorithms can be used as well. In order to reason about the sampling points, we look at the case where $y(X(t))$ is a piecewise-constant vector signal that has sufficient variability per time interval (i.e., there are ample controller changes). Furthermore, let us assume that f, g faithfully approximate g, y respectively in most of the data, and that places where this approximation fails appear independently between the signals. The following assertion can be made:

Lemma 1: Given two points t_b, t_g such that: 1) in the neighborhood of t_g, f faithfully represents g and g faithfully represents y, and 2) there is no controller switch. Furthermore, 3) at t_b, one of the three conditions does not hold. Then, f changes at t_b, but not in t_g.

Proof: At t_g, y is a constant signal (2), therefore g is constant (1), and therefore f is constant (1). Conversely, at t_b one of the three conditions fails (3). If theres a controller switch but, f, g are faithful – g switches and therefore f changes. If g isn't faithful but y is constant – g changes, therefore f changes. If f doesn't approximate g – g doesnt change but f changes. This means that around t_b there is a large change in f ■

An overview of the sampling algorithm is shown in Alg. 2. By definition of the BalancedPartition algorithm in k-segment coresets, change points in a piecewise constant stream $f(o(t))$ tend to have coreset segment boundaries around them. Since in Lines 2,3 of Algorithm 2 we sample more frequently around coreset segments boundary, intuitively there is a neighborhood of t_g that will be sampled less than t_b. While this is a limit case which requires already a good deal of training for the controllers and embedding function, we note that this is the behaviour observed in practice in our experiments in Section V.

V. Results

We now demonstrate our approach on two datasets. One dataset (Dataset I) we used comes from the Duckietown autonomous vehicles course [28]. In this setup, small-scale vehicles are able to navigate a small scale city. We have utilized logs recorded from 10 robots used in the course. We used the transitions between discrete controllers in these robots as annotation data to train the the embedding on the forward-facing camera raw footage. While we have allocated two dimensions to the embedding, the resulting embedding is 1D, and includes two tight clusters that correspond to these two modes. Embedding streams, and the transitions candidates detected via the coreset segments are shown in Fig. 6. In this case, the robot operates in one of two modes: LANE_FOLLOWING, or INTERSECTION_TRAVERSAL. In the system, the mode transitions are triggered by detecting stop lines and completing open-loop intersection traversal actions. However, our learns the transitions from the raw visual input without explicitly knowing about things like "stop lines" or "lanes" – more generally, it is unaware of the controllers' inner structure.

To simulate controllers in a more challenging visual environment, we have collected a dataset (Dataset II) of 130 car rides within a 20km radius, for a total of 60 hours. We used human annotation to approximate driving types according to a set of 4 scene types (Set I): "parking lot", "suburban driving", "urban area", "highway". The dataset contained 140 annotation of the different ride segments.

For the purpose of training the networks for Set I, we used 70% of the annotation, sampling $20K$ pairs of time points from the videos, and embedded the images into a 2D feature space. We used data augmentation with respect to the field-of-view (FOV) of the camera by subsampling windows from the camera's FOV, and found it sufficient for this nuisance factor.

The resulting feature streams are shown in Fig. 5, along with the coreset segments at the top layer of the coreset tree. The feature space is shown in Fig. 4, for six 45-minute drives. As can be seen, the same clusters form in all videos where they appear, corresponding to the four categories of "parking lot", "suburban driving", "urban area", "highway", and are robust to different illumination conditions and different locations. Utilizing the coreset segments, compared to uniform segmentation, we obtain improved fit of the detected transitions, as shown in Table I, where we computed a modified Rand index for both datasets, compared to manual annotations. More importantly, the coreset segments edges cover the transitions of the embedded stream, and the transitions in the locations in the videos.

We then utilized the resulting coresets to sample data points according to Alg. 2, to see their distribution and coverage of different regions in the data. The resulting samples are concentrated around transition regions, and regions of uncertainty in the embedding, which are the regions the often require more training data to cover well. This demonstrates the utility of the coresets for extracting training data for annotations from a set of large-scale datastreams.

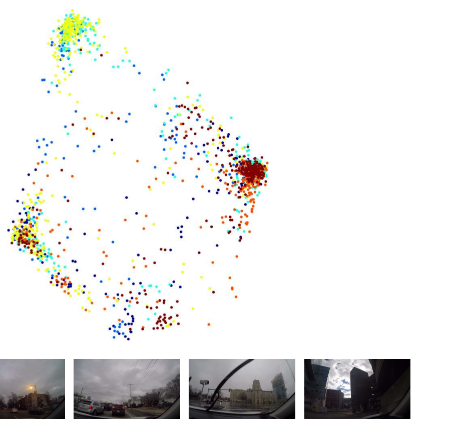
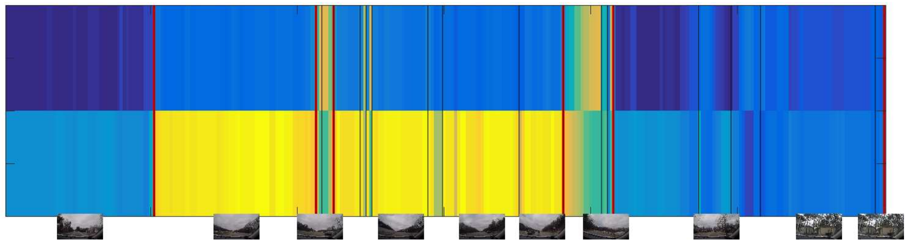
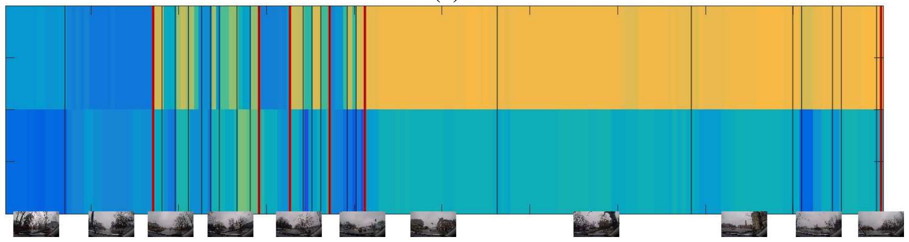


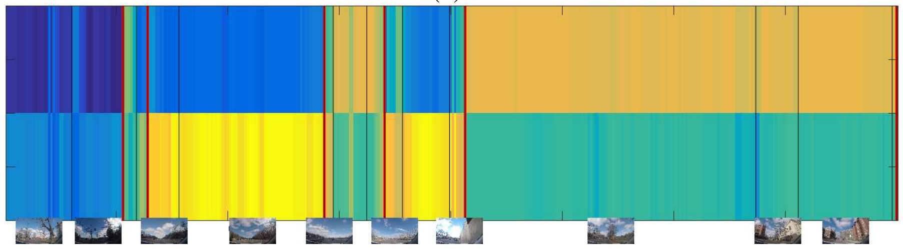
Fig. 4. Top: embedding space for different time points, sampled from 6 rides in our car dataset. Different colors show different rides at different times of daylight and different weather (sunny, slight rainy/clouds), with different colors representing separate rides. The embedding was train according to the controllers in Set I (parking, city, suburb, highway). Clusters in the data can be clearly seen to capture prevalent modes (Top - "highway", right - "city", bottom - "parking", left - "suburb"). Bottom row shows images from different rides.



(a)



(b)



(c)

Fig. 5. Example embedding vector streams from the videos, each rows represents a 30-60 minutes driving sequence, the horizontal and vertical axes represent time and feature space elements respectively. Black vertical lines represent coreset segments boundaries, and red vertical lines demonstrate the resulting segmentation into $k = 6$ via the dynamic programming approach, which the coreset approximates. As can be seen, coreset segments capture the transitions, with multiple segments in regions of uncertainty.

TABLE I

SEGMENTATION MEASURES – MODIFIED RAND INDEX RESULTS WITH OUR APPROACH, COMPARED TO THOSE OF UNIFORM INTERVALS OF THE SAME AVERAGE FREQUENCY. AS CAN BE SEEN, THE CORESET SEGMENTS PROVIDE A GOOD PARTITION OF THE STREAM THAT NATURALLY FOLLOW TRANSITION POINTS.

Dataset	Rand index, our approach	Rand index, Uniform Intervals
Dataset I (Duckietown)	**0.9357** $\pm$ **0.0231**	0.6862 $\pm$ 0.1156
Dataset II	**0.8198** $\pm$ **0.0328**	0.5798 $\pm$ 0.0784

VI. CONCLUSIONS

In this paper we developed a deep embedding approach to allow switching of driving modes based on the car's visual feed, and showed how k-segment coresets allow us both easy transition detections and data collection for training the embedding and the controllers. This work raises important questions as to the unsupervised case where both control modes and embeddings are learned in parallel, as well as the optimal controller and embedding structure for complex autonomous robots in order to improve safety and performance for these systems.

REFERENCES

[1] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan, "Geometric approximation via coresets," *Combinatorial and computational geometry*, vol. 52, pp. 1–30, 2005.

[2] J. Alonso-Mora, P. Gohl, S. Watson, R. Siegwart, and P. Beardsley, "Shared control of autonomous vehicles based on velocity space optimization," in *ICRA*, May 2014, pp. 1639–1645.

[3] M. Athans, D. Castanon, K.-P. Dunn, C. Greene, W. Lee, N. Sandell, and A. Willsky, "The stochastic control of the f-8c aircraft using a multiple model adaptive control (mmac) method–part i: Equilibrium flight," *IEEE Transactions on Automatic Control*, vol. 22, no. 5, pp. 768–780, 1977.

[4] C. Bergenhem, S. Shladover, E. Coelingh, C. Englund, and S. Tsugawa, "Overview of platooning systems," in *Proceedings of the 19th ITS World Congress, Oct 22-26, Vienna, Austria (2012)*, 2012.

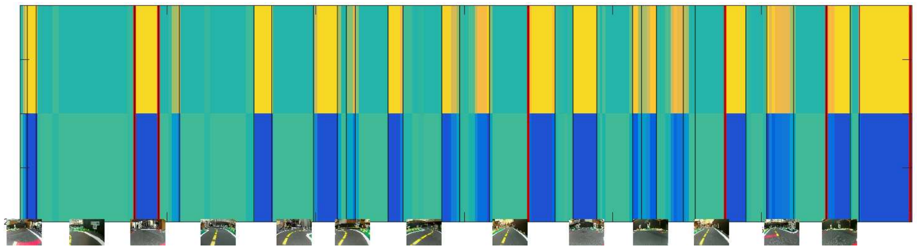
[5] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," *CoRR*, vol. abs/1604.07316, 2016. [Online]. Available: http://arxiv.org/abs/1604.07316

[6] C. J. C. Burges, "Dimension reduction: A guided tour," *Foundations and Trends in Machine Learning*, vol. 2, no. 4, pp. 275–365, 2010. [Online]. Available: http://dx.doi.org/10.1561/2200000002

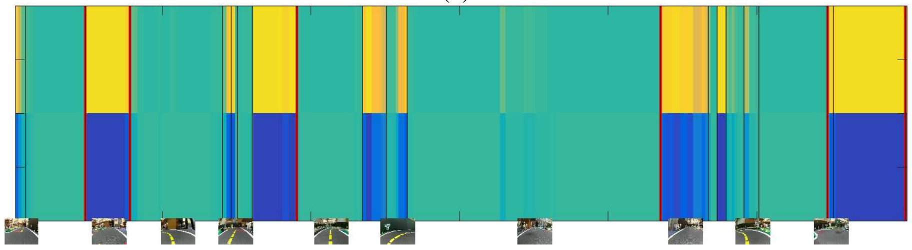
[7] A. Dubey, N. Naik, D. Raviv, R. Sukthankar, and R. Raskar, "Coreset-based adaptive tracking," *CoRR*, vol. abs/1511.06147, 2015. [Online]. Available: http://arxiv.org/abs/1511.06147

[8] M. Fallon, S. Kuindersma, S. Karumanchi, M. Antone, T. Schneider, H. Dai, C. P. D'Arpino, R. Deits, M. DiCicco, D. Fourie, *et al.*, "An architecture for online affordance-based perception and whole-body planning," *J. Field Robotics*, vol. 32, no. 2, pp. 229–254, 2015.

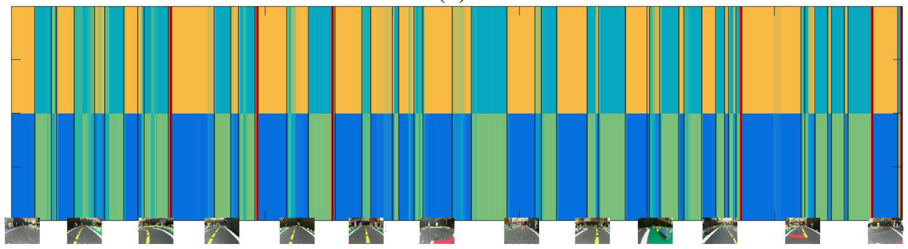
[9] C. Finn, T. Yu, J. Fu, P. Abbeel, and S. Levine, "Generalizing skills with semi-supervised reinforcement learning," *CoRR*, vol. abs/1612.00429, 2016. [Online]. Available: http://arxiv.org/abs/1612.00429



(a)



(b)



(c)

Fig. 6. Embedding results obtained on data from Duckietown, trained using the controllers from the system's ROS nodes. The resulting coreset segments boundaries are shown as black vertical lines. Note that in this case the distribution in embedding space is approximately one dimensional, capturing intersections (yellow-blue columns) and lane-following (light blue columns) modes.

[10] M. Forghani, J. M. McNew, D. Hoehener, and D. D. Vecchio, "Design of driver-assist systems under probabilistic safety specifications near stop signs," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 1, pp. 43–53, Jan 2016.

[11] E. Fox, E. B. Sudderth, M. I. Jordan, and A. S. Willsky, "Nonparametric bayesian learning of switching linear dynamical systems," in *Advances in Neural Inf. Proc. Sys.*, 2009, pp. 457–464.

[12] T. Fraichard and H. Asama, "Inevitable collision states. a step towards safer robots?" in *IROS*, vol. 1, Oct 2003, pp. 388–393 vol.1.

[13] J. H. Gillula, H. Huang, M. P. Vitus, and C. J. Tomlin, "Design of guaranteed safe maneuvers using reachable sets: Autonomous quadrotor aerobatics in theory and practice," in *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 1649–1654.

[14] C. Goller and A. Kuchler, "Learning task-dependent distributed representations by backpropagation through structure," in *Neural Networks, 1996., IEEE International Conference on*, vol. 1. IEEE, 1996, pp. 347–352.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[16] D. Hoehener, G. Huang, and D. D. Vecchio, "Design of a lane departure driver-assist system under safety specifications," 2016. [Online]. Available: http://hdl.handle.net/1721.1/101596

[17] G. Hoffmann and C. Tomlin, "Mobile sensor network control using mutual information methods and particle filters," *Automatic Control, IEEE Transactions on*, vol. 55, no. 1, pp. 32–47, jan. 2010.

[18] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 2014, pp. 675–678.

[19] M. J. Johnson and A. S. Willsky, "Bayesian nonparametric hidden semi-markov models," *Journal of Machine Learning Research*, vol. 14, no. Feb, pp. 673–701, 2013.

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[21] J. Lygeros, C. Tomlin, and S. Sastry, "Multiobjective hybrid controller synthesis," in *International Workshop on Hybrid and Real-Time Systems*. Springer, 1997, pp. 109–123.

[22] ——, "Hybrid systems: modeling, analysis and control," *preprint*, 1999.

[23] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin, "A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games," *IEEE Transactions on Automatic Control*, vol. 50, no. 7, pp. 947–957, July 2005.

[24] K. P. Murphy, "Hidden semi-Markov models (HSMMs)," Nov. 2002.

[25] ——, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[26] NHSTA, "Federal automated vehicles policy accelerating the next revolution in roadway safety," Sep 2016.

[27] S. Niekum, S. Osentoski, C. Atkeson, and A. G. Barto, "Champ: Changepoint detection using approximate model parameters," Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-14-10, June 2014.

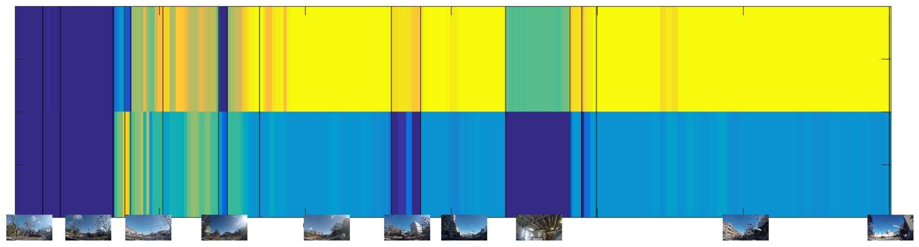
[28] L. Paull, J. Tani, H. Ahn, J. Alonso-Mora, L. Carlone, M. Cap, Y. F. Chen, C. Choi, J. Dusek, Y. Fang, D. Hoehener, S.-Y. Liu, M. Novitzky, I. F. Okuyama, J. Pazis, G. Rosman, V. Varricchio, H.-C. Wang, D. Yershov, H. Zhao, M. Benjamin, C. Carr, M. Zuber, S. Karaman, E. Frazzoli, D. D. Vecchio, D. Rus, J. How, J. Leonard, and A. Censi, "Duckietown: an open, inexpensive and flexible platform for autonomy education and research," in *ICRA*, 2017, pp. 1–8, Accepted.

[29] J. M. Phillips, "Coresets and sketches," *arXiv preprint arXiv:1601.00617*, 2016.

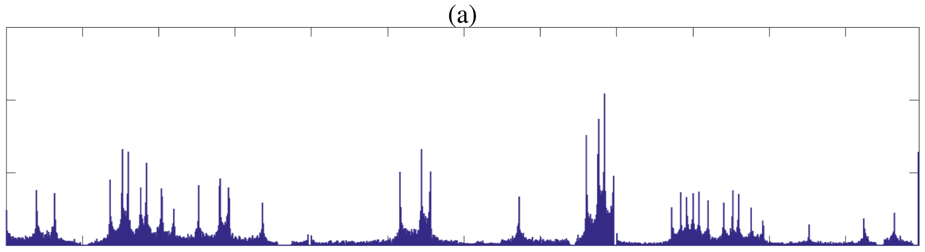
[30] D. A. Pomerleau, "ALVINN, an autonomous land vehicle in a neural network," Carnegie Mellon University, Computer Science Department, Tech. Rep., 1989.

[31] G. Rosman, M. V. Volkov, D. Feldman, J. W. Fisher III, and D. Rus, "Coresets for k-segmentation of streaming data," in *Advances in Neural Inf. Proc. Sys.*, 2014, pp. 559–567.

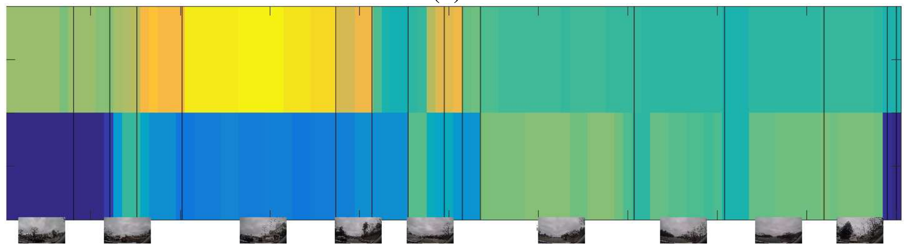
[32] W. Schwarting, J. Alonso-Mora, L. Paull, S. Karaman, and D. Rus, "Parallel autonomy in automated vehicles: Trajectory generation with



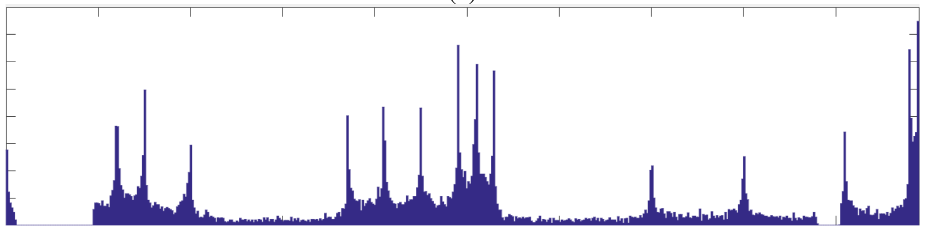
(a)



(b)

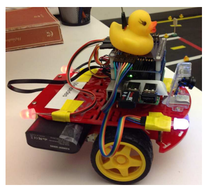


(c)

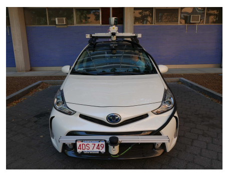


(d)

Fig. 7. Examples of sampled locations by the coreset. (a),(c): the original stream of embedded vectors. (b),(d): the histogram of sampled frames in the videos, computed over 10^6 samples. As can be seen, location of uncertainty and transitions are the parts that are most often sampled.



(a)



(b)

(c)

Fig. 8. Examples sensors used in our datasets: (a): A Duckiebot used for in Duckietown (dataset I). (b),(c) - autonomous vehicle w/ PointGrey Grasshopper3 6.0 MP, and dashboard-mounted GoPro Hero4 cameras used in dataset II.

real-time obstacle avoidance and human input optimization," in *ICRA*, 2017, pp. 1–8, accepted.

[33] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

[34] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[35] M. Volkov, D. A. Hashimoto, G. Rosman, O. R. Meireles, and D. Rus, "Machine learning and coresets for automated, real-time video segmentation of laparoscopic surgery," in *SAGES*, Boston, Massachusetts, Mar. 16–19 2016.

[36] M. V. Volkov, G. Rosman, D. Feldman, J. W. Fisher III, and D. Rus, "Coresets for visual summarization with applications to loop closure," in *ICRA*, 2015, pp. 3638–3645.

[37] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell, "Distance metric learning with application to clustering with side-information," in *Advances in Neural Inf. Proc. Sys.*, vol. 15, no. 505-512, 2002, p. 12.

[38] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," *arXiv preprint arXiv:1612.01079*, 2016.