An interaction aware trajectory prediction transformer for racing

Finn Lukas Busch
3037535950

Sean Maroofi
3037530997

Niklas Trekel
3037535937

***Abstract*—In this work, we propose a transformer based approach to predict trajectories of multiple race cars within complex scenarios in autonomous racing. Short-term historic data together with contextual semantic images were employed to encode wide-ranging information about multiple cars in one frame. We use a vision transformer trained in an unsupervised fashion to extract meaningful information from the semantic images. A multi-agent interaction aware transformer is used to make predictions accounting for both dynamics and vehicle interactions.**

I. Introduction

While autonomous racing is mainly concerned with lap-time optimization to this date, interactions such as overtakes between autonomous vehicles are still rare. This is partly due to the highly dynamic, fast and possibly adversarial interactions between the different vehicles. To act safely in these regimes, it is crucial to have a precise understanding of what actions other vehicles take in the near future. A system that aims to predict these well should account for all interactions between the involved vehicles. This makes the task especially challenging because the number of vehicles in different situations change and because it requires an understanding of how information changes and relates over two dimensions - the time and the different vehicles.

A. Related Work

The principle of combining contextual information from semantic maps with trajectory histories to achieve environmental-aware trajectory predictions has been widely explored in the context of traffic. In [2], a time sequence of semantic map representations of parking lots were used in two ways - to simultaneously infer about target parking spots and improve trajectory predictions for parking maneuvers of individual cars. General dynamical traffic situations increase the number of observed cars and therefore require jointly predicted trajectories for multiple cars [4], [5]. Here, static semantic representations proved valuable in enforcing feasibility of the predicted trajectories. As in many other recent applications which include sequential context, a large amount of recent approaches are based on the Transformer network structure, presented in [3]. Since the original structure was specifically designed for linking information along one dimension, [5] proposed a modified method for the Multi-Head Attention to account for awareness of multiple agents. Nowadays, CNNs are still mainly used for extracting meaningful features from semantic representations. However, novel attention-based mechanisms like the Vision Transformer (ViT) [6] are making their way to be used likewise [4].

B. Contribution

In this work, we propose a learning-based approach that makes short-term predictions of a varying number of vehicles only using short-term historic data. Note that here we do not aim to learn specific driver's policies but rather aim to obtain a model that can predict intentions based on very short observations. Together with the state history of the vehicles of interest, we construct semantic images of the historic data to provide context. Features are extracted from the semantic images using ViTs. To pre-train the ViTs, masked autoencoders (MAE) are used. These features and the state information are used by our proposed transformer network architecture attending over both time and the number of vehicles. The model is trained and evaluated on a dataset over all tracks driven by an experienced player versus AI in the *F1 2020* racing game.

II. Problem Formulation

The goal of this work is to provide a system that makes accurate short-term joint predictions of a varying number of vehicles n_{veh} in adversarial racing.

A. Inputs

The inputs consist of short-term historic information. A vehicle state of vehicle i at time t is defined as

$$\boldsymbol{x}^{(i,t)} = \begin{bmatrix} x^{(i,t)} & y^{(i,t)} & \gamma^{(i,t)} \end{bmatrix}^T$$

where x and y denote the position of the vehicle while γ depicts the orientation. A trajectory of one vehicle consists of multiple states. We define $\mathcal{X}_{\text{hist}}^{(t)}$ as the set of trajectories of the n_{veh} vehicles over n_{hist} time steps in the range from $t - (n_{\text{hist}} - 1)$ to the current time step t.

Furthermore, semantic images are supplied to provide contextual information of the situations. We generate images for each of the n_{hist} time-steps and for each vehicle. The images only differ in terms of coloring between the different vehicles. For each vehicle, we generate the image once where the vehicle of interest is colored differently from the others, as depicted in fig. 1. The set of semantic images for all vehicles for the n_{hist} steps up to the time step t is named $\mathcal{I}_{\text{hist}}^{(t)}$. Consequently, for a sample with $n_{\text{hist}} = 10$ and $n_{\text{veh}} = 4$, we generate $n_{\text{hist}} \cdot n_{\text{veh}} = 40$ images.

B. Outputs

As we aim to predict trajectories of vehicles to use them to act safely, positional and orientation information of the vehicles in the future is sufficient. Thus, the outputs are n_{pred} future vehicle states for each vehicle, again consisting of position and orientation in 2D. The corresponding set of n_{pred} future states for all vehicles from time step $t+1$ up to $t+n_{\text{pred}}$ is called $\mathcal{X}_{\text{pred}}^{(t)}$.

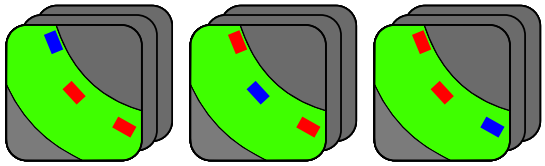

Fig. 1: Visualization of the input sequences for different vehicles of one sample. Every driver's trajectory should be predicted. To be distinguishable, one of the involved vehicles is colored different in the chosen sample over the time sequence.

III. Dataset

Currently, there is still limited public availability of real-world telemetry data of races. Therefore, and considering the close to reality experience of video games nowadays, we decided to gather our data from the *F1 2020* racing game by *Codemasters*. Offering an interface via User Datagram Protocol (UDP), the game provides access to telemetry data on demand. To generate meaningful semantic images, telemetry data together with the track layout is required. For the former, we utilize the dataset from [1], consisting of telemetry data of 20 drivers from 22 races each about half an hour. To reconstruct the track layout, we drove along the track boundaries of all tracks and frequently sampled the positional data. Further postprocessing and interpolation was done as a final step to yield accurate maps of the tracks used in the game.

A. Data Preprocessing

The collected data is interpolated and sampled with a fixed frequency of $10\,\text{Hz}$. Each dataset sample consists of n_{hist} time steps for the historical data and n_{pred} time steps for the future. To make sure that the data samples used for training contain a sufficient proportion of interaction situations, we determine for each sample whether it contains an overtake and measure the minimum time difference between racing drivers as a heuristic for estimating the likeliness of interactions. This allows to prefer interaction-heavy samples during training and evaluation. Situations where the vehicles are not on the track (e.g. for pit-stops) are filtered out as well. The historical data of each sample contains then n_{hist} generated images and states for each driver involved in the current situation, whereas the predicted data just contains the states to be predicted for each driver. As a last preprocessing step, we ensure that there are no discontinuities in heading angle in the dataset which are caused by angle wrapping.

B. Image Generation

To generate the semantic bird's-eye-view images for contextual information, we manually reconstruct the tracks from the *F1 2020* racing game and use it together with the dataset from [1] to generate semantic maps of the situations. These maps contain a local cutout of the track together with the vehicles involved in the situation. We decided to choose a resolution of $H \times W = 224 \times 224$ pixels for the image samples over 3-channels of RGB-data. A sequence of n_{hist} images displays a situation of the race where a certain number of n_{veh} are involved. For each sequence of images we chose a cutout which is fixed over time. It is large enough to capture both the historic data and information about the track to support the understanding of the movement of the vehicles. The cutout size is dynamically generated, based on the current situation. While iterating through the timeline of a race, we define a fixed-size initial frame centered on one of all the vehicles. The visible vehicles determine the number of vehicles involved in the current situation. Then, the cutout is inflated to capture the whole history of positions of all involved vehicles. This ensures that the cutout adapts to the current speed of the vehicles. For every time step, each vehicle only appears once in all generated samples. This aims to reduce redundancy in the data. The vehicles are represented by small rectangles with sizes according to the Formula 1 rules. The difference in color of the vehicle of interest was chosen to simplify relating vehicles in the images to the states provided to the predictor. The track is fully coloured in green to depict the drivable area. The semantic images are generated using OpenGL. Without a GPU, 100 images are generated in less than $0.1\,\text{s}$ which is sufficiently fast to generate these images in real-time. Image generation is expected to be even faster if a GPU is used.

All together, we generated a dataset consisting of $363\,401$ training samples which includes $5\,312\,640$ images. For evaluation, a full track is excluded from training containing $37\,351$ samples.

IV. Network Architecture

Generally, our model contains two sub-modules

1) A vision transformer feature extractor that extracts useful information $\mathcal{F}_{\text{hist}}^{(t)}$ from the historic semantic images $\mathcal{I}_{\text{hist}}^{(t)}$
2) A transformer trajectory predictor that processes the extracted features $\mathcal{F}_{\text{hist}}^{(t)}$ and historic state information $\mathcal{X}_{\text{hist}}^{(t)}$ to predict future trajectories $\mathcal{X}_{\text{pred}}^{(t)}$

The general structure is depicted in fig. 2.

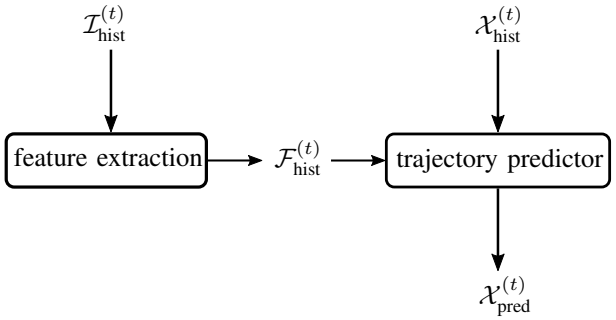

Fig. 2: Dataflow through the two models.

A. Feature Extractor

The feature extractor is supposed to extract useful information from the semantic images. To reduce the model size in training, it is favorable to train the feature extractor separately from the trajectory predictor. In recent works, vision transformers [6] have been shown to outperform traditional CNNs. Furthermore, in [7] masked auto encoders were proposed as a method for unsupervised pre-training of vision transformers. Here, an encoder-decoder transformer architecture is employed to reconstruct images from incomplete parts of the original image. Ideally, the encoder will then be able to extract meaningful features of the original images so that they can be used for different tasks. This is very useful for our application as hand-designing features for supervised training of the vision transformer is difficult and end-to-end-training of the complete model is computationally expensive and very time consuming. In [7] different ViT models are proposed. For our needs, the base version with patch size 16 seemed to be sufficiently complex to extract useful features while allowing for fast training. The ViT will process batches of images of size $H \times W \times C$ to batches of feature vectors of size n_{feat}.

B. Trajectory Transformer

The trajectory transformer is the core part of the prediction framework. The underlying structure of the problem poses one main challenge - available information needs to be processed considering relations over two dimensions:

1) Time: understanding the evolution of states over time both for prediction and history is crucial to make meaningful predictions
2) Number of vehicles: every action taken by a driver is dependent on the other drivers, the other drivers' histories and the other drivers' future actions

For best results, it is crucial to not separate these two dimensions but rather consider both when making predictions. Flattening these two dimensions into one dimension with variable sequence length (scaling with number of vehicles and time history) results in suitable inputs for the attention mechanisms. Intuitively, this also allows the transformer to simultaneously attend and reason over both dimensions. The key challenge here is to still keep track of the affiliation of single entries in time and agent domain. Therefore, we make adaptations to the original transformer architecture as shown in fig. 3.

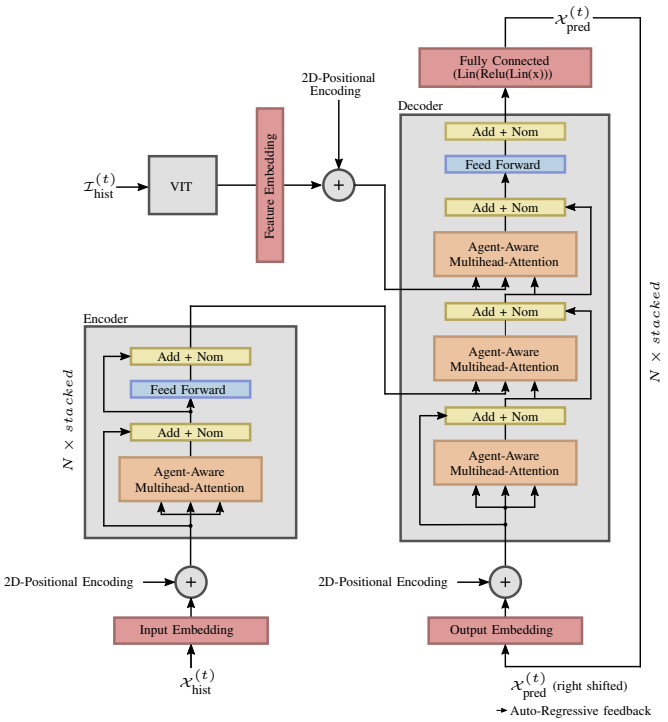

Fig. 3: Network structure: The Decoder and Encoder combined with the ViT. Historic state information $\mathcal{X}_{\text{hist}}^{(t)}$ together with features from the ViT are fed into the transformer which produces the predicted future trajectory $\mathcal{X}_{\text{pred}}^{(t)}$.

1) Agent-aware multi-head attention: As the standard attention mechanism proposed in [3] only models dependencies over one dimension using 1D-positional encoding, we extended it to 2D positional encoding as proposed in [6]. As opposed to 1D-positional encoding, here two encodings are applied - one for time and one for the vehicle id. Each of these two encodings is applied to half of the entries in the feature vector ultimately leading to one half of the feature vector encoding time and one half encoding the vehicle id. Secondly, the standard multi-head attention in each sublayer of the transformer is replaced by the agent aware attention mechanism proposed in [5]. The mechanism treats the attention between the same vehicle and pairwise different vehicles separately. This is achieved by using two separate weight matrices in combination with two corresponding masks, which are inverse to each other. The first mask only allows entries to attend to other entries belonging to the same vehicle but to arbitrary entries along the time dimension. Conversely, the second mask allows attention to all time steps and only other vehicles. This ideally allows the network to separately optimize parameters towards predicting meaningful trajectories, while also gaining awareness of other vehicles.

2) Encoder: Our encoder structure is shown in fig. 3 and matches the original transformer structure introduced in [3], apart from the previously described changes. The input $\mathcal{X}_{\text{hist}}^{(t)} \in \mathbb{R}^{N \times n_{\text{veh}} \times n_{\text{hist}} \times n_{\text{state}}}$ is augmented to the transformer dimension n_{traf} by the linear input embedding. Here, N is the batch size

and n_{state} the state dimension of the vehicles. After applying the 2D-positional encoding, it is flattened to the dimension $\mathbb{R}^{N\times(n_{\text{veh}}\cdot n_{\text{hist}})\times n_{\text{traf}}}$ and fed to the encoder.

3) Decoder: In general, the decoder structure has one additional multihead-attention layer in each decoder-sublayer compared to the decoder from [3], which integrates the features provided by the ViT. The input is the right shifted, predicted future trajectory $\mathcal{X}^{(t)}_{\text{pred}} \in \mathbb{R}^{N\times n_{\text{veh}}\times(n_{\text{pred}}+1)\times n_{\text{state}}}$. The right shift is done by prepending the current state $\boldsymbol{x}^{(i,t)}$ of each vehicle i to the time sequence. Note that we use auto-regressive prediction of the trajectories not only during evaluation time, but also during training time. Similar to [5], we experienced better results at test-time which in our case resulted in noticeably smoother trajectories and a more consistent evolution over time. However, since one forward pass during training time requires to feed the outputs $n_{\text{pred}}+1$ times through the decoder, the improvement is achieved at the expense of higher training time. This means, that the initial input to the decoder is the current state $\boldsymbol{x}^{(i,t)}$ of all vehicles and the final output of the decoder is the predicted full trajectory $\mathcal{X}^{(t)}_{\text{pred}} \in \mathbb{R}^{N\times n_{\text{veh}}\times(n_{\text{pred}}+1)\times n_{\text{state}}}$. Here, the last entry of the time sequence is a self-defined end token. Similar to the encoder, we at first perform an output-embedding to transform the state-dimension n_{state} to the transformer dimension n_{traf}. After adding the 2D-positional encoding, the input is again flattened and fed into the decoder. Since we don't feed the ground truth of the predicted trajectory into the decoder, the first self-attention layer does not require a mask to prevent looking into the future. As usual, multihead-attention using the encoder output is performed with the output of the self-attention layer. The second multihead-attention layer in each decoder-sublayer uses the given contextual features of the ViT $\mathcal{F}^{(t)}_{\text{hist}} \in \mathbb{R}^{N\times n_{\text{veh}}\times n_{\text{hist}}\times n_{\text{feat}}}$. Another embedding layer adapts the dimension of the features n_{feat} to the transformer dimension n_{traf}. Positional encoding is used here as well to support the transformer in relating the vehicle-specific features to the current representation used as the values in the multihead-attention. After the feed-forward layer, two fully connected layers generate the output prediction $\mathcal{X}^{(t)}_{\text{pred}}$.

V. Experiments

A. Training

Training consists of multiple steps. First, the vision transformer is pre-trained as part of the masked autoencoder. Then, using the pre-trained ViT, its head is trained together with the trajectory transformer on all samples generated and lastly fine-tuned on a dataset consisting of mostly interaction-heavy samples. Ideally, a final end-to-end fine-tuning training step would be beneficial but was not suitable for us as the hardware we had access to does not allow to train both the ViT and trajectory transformer at the same time. For all training steps, we early-stop on convergence. We predict $n_{\text{pred}} = 10$ future states using $n_{\text{hist}} = 10$ historic time-steps. Training was done on a desktop with a NVIDIA RTX 3080 with 10 GB of VRAM and a Intel Core i7-10700k with 32 GB of RAM.

1) Vision Transformer: The vision transformer is trained on a subset of all the images generated yielding a dataset of 300000 images. The chosen hyperparameters are shown in table I. This is exactly the ViT-Base model proposed in [6] with a final affine layer from 768 to 1000. The final affine layer is not trained in pre-training but rather trained together with the trajectory transformer. The ViT is trained using the standard masked autoencoder procedure with the training params described in II until convergence.

Model name	ViT Base
Layers	12
Hidden size	768
MLP size	3072
Patch size	16
Heads	12
Output size	1000

TABLE I: ViT Parameters

Optimizer	AdamW
Batch size	32
Mask ratio	0.75
Learning rate	10^{-4}

TABLE II: ViT Pre-Training Params

Fig. 4 shows samples of reconstructed images. It is clear to see that the masked autoencoder is able to reconstruct the track boundaries as well as some of the vehicles indicating that the encoder extracts information sufficient to understand the context. However, there are also a lot of artifacts remaining. We think that with more data and training time, those should become less but decided to keep it as it is as it clearly already understands the track boundaries. After pre-training, the output for all images of the encoder without the head are extracted and saved.

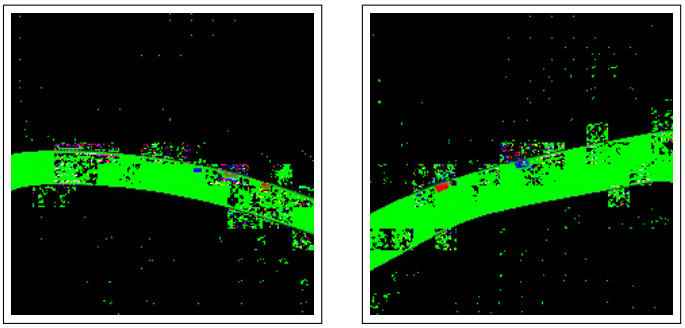

Fig. 4: Sample reconstructed ViT images

2) Trajectory transformer: The chosen settings for the trajectory transformer are outlined in tab. III. Additionally, table IV shows the chosen training parameters. The trajectory transformer is trained on all except one race tracks together with the ViT head.

Encoder Layers	6
Decoder Layers	6
Heads	8
Encoder/Decoder Model size	128
Feedforward Layer size	2048
Decoder Output Hidden size	256

TABLE III: Trajectory Transformer Parameters

Optimizer	AdamW
Batch size	32
Dropout	0.3
Learning rate	$5 \cdot 10^{-5}$

TABLE IV: Trajectory Transformer Training Parameters

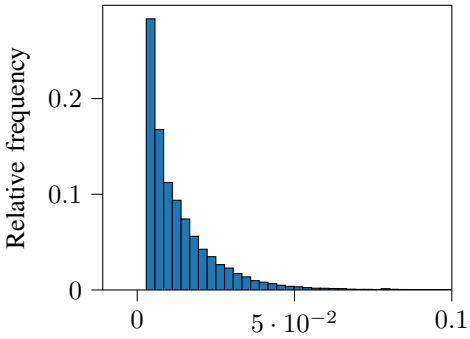

Fig. 6: Testset squared heading error histogram.

B. Results

To evaluate the performance of our model, we investigate the mean squared positional error in the predictions as well as the mean squared error in the heading angle over all predictions of the evaluation set. The errors are shown in table V. Considering the high speeds, we achieve satisfying positional and heading accuracy. Moreover, evaluation performance is close to training performance.

	Squared positional errors $[\mathrm{m}^2]$	Squared heading errors $[\mathrm{rad}^2]$
Training	2.1022	0.00987
Evaluation	2.6337	0.01118

TABLE V: Training and evaluation errors

Furthermore, fig. 5 and fig. 6 depict the relative squared error frequency. While there are a lot of errors that lie below the average error, we can observe that there are some samples where the positional error is very large. This indicates that for some samples, the predictions are very off which is what we investigate in the next part.

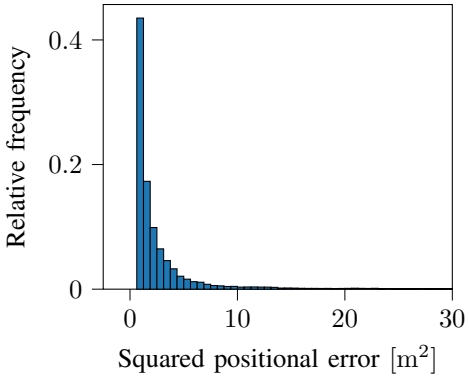

Fig. 5: Testset squared positional error histogram.

In the following, we will show some samples from the test set that depict accurate predictions and others that capture the most common situations where the predictor falls short. The blue lines indicate the ground truth future driven path while red denotes the predictions made by our model. Note that some frames show vehicles without any predictions. This is because those vehicles are not of interest for the interactions between the other vehicles.
Fig. 7 shows good prediction behavior. It becomes apparent that the predictor understands the dynamics of the vehicles well and also predicts drivers taking the inside of the curve in an attempt to follow the optimal race-line. Furthermore, it predicts non-intersecting trajectories as it learnt that collision-free trajectories are much more likely.

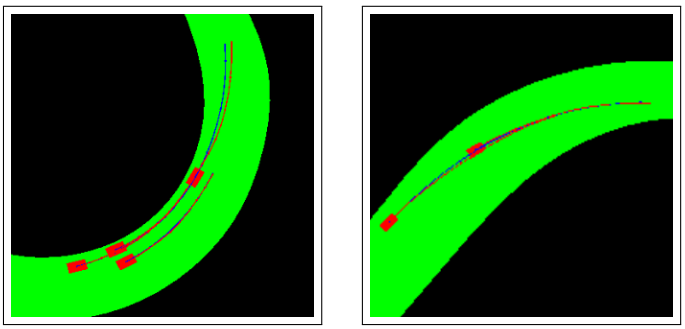

Fig. 7: Samples with good predictions

Fig. 8 shows samples where our model does not predict well. We identified two main issues captured here. Firstly, the contextual features sometimes seem to not be understood well by the predictor. Thus, it predicts vehicles leaving the track, as shown in the right picture. We hypothesize that this is an issue that could be addressed by end-to-end-training the whole system which was not possible for us for the reasons mentioned earlier.

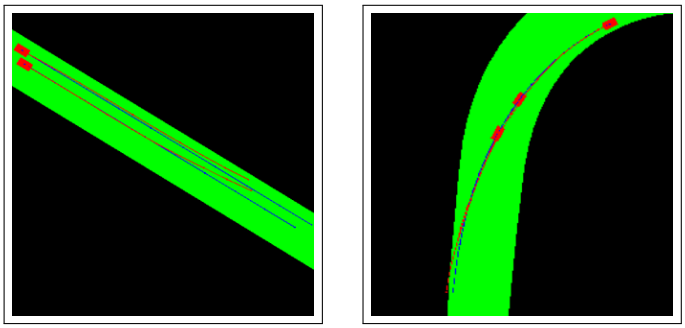

Fig. 8: Samples with inaccurate predictions

Secondly, the predictor sometimes makes seemingly valid predictions as shown in the left image that do nevertheless not match the ground truth. This could be caused by a number of reasons. Investigating multiple samples, we identified two likely reasons. First of all, in many situations, multiple outcomes are equally likely and our system is not able to capture multi-modalities. For example, the leading vehicle could try to block the overtaking attempt by the following vehicle or decide to maintain speed and going straight. Both moves could be valid here but our model is not able to capture that. Approaches as the one discussed in [5] aim to tackle this by explicitly modeling multi-modalities but introduce further complexity to the system. Secondly, for example for long straights, the contextual images do not provide enough information to predict where the vehicles will go. For instance, consider a very long straight followed by a turn. Here, the semantic images will most likely not contain the turn but vehicles will prefer to stick to the side of the track yielding the best entry to the turn. Without knowing the track, there is no way for the predictor to accurately predict the vehicle behavior. To solve this, more information about the track such as the pre-computed optimal race-line could be supplied.

VI. Conclusion

We proposed a systematic approach for interaction-aware trajectory prediction in autonomous racing using short-term historic data employing transformers. Semantic images were generated and used to capture contextual information and used by a transformer trajectory predictor that attends both over time and the number of agents ultimately allowing to consider all the relations between the data available. The preliminary results show that our approach conceptually works but falls short in certain situations. Future work should go into end-to-end-training of all modules as well as modeling multi-modalities as described in [5].

VII. Final Project reflection

This section is supposed to give an overview over the contributions of each group member as well as a short description of our experiences with the project. Conceptually, all design choices were made together as a group with no group member being responsible to make design choices alone. In general, we met up as the whole group for most of the project, so that crucial decisions could be immediately discussed and the workload was distributed equally.

In the initial phase of the project, a substantial amount of effort went into reading papers and finding existing approaches for our task. This was equally done by all of us. After, we made some preliminary design choices on the basic structure of the whole module, what the inputs and outputs of the system should be and searched for available datasets that matched our needs. Following the choice of our dataset a lot of work was required to generate a usable dataset for the actual training. We sampled the track boundaries using the actual game (Finn, Sean) to gather maps of the tracks. The telemetry data needed to be subsampled with a fixed frequency, labeled for interaction and outliers were sorted out (Niklas). For the generation of the images, a filter was required to split up the continuous race data into individual situations and label involved vehicles (Sean). The sampled track points from the game needed further processing, which is why we implemented a GUI tool to easily manipulate sample points (Finn) and preprocessed all tracks (Sean, Niklas). For efficient image generation and visualization, we further subsampled the tracks based on curvature (Niklas) and implemented a visualization in OpenGL (Finn) from scratch from which the images were generated. We aimed to make this step as computationally efficient as possible as a lot of images were needed. To our surprise, especially the data generation and data management required a lot of work. Most significantly, managing the huge amount of data ($\sim$ 5 million images), reducing the bottleneck induced by disk i/o and the hardware requirements posed by the complex models and generating the dataset efficiently in the first place took a very long time. Most of these issues are usually already dealt with if using publicly available datasets which is why we underestimated the amount of effort needed for this. At this point, we slightly split up the responsibility for the implementation of the main remaining points. Finn spend a lot of effort on efficient data generation and implementing dataloaders to handle the data required for the training of the feature extractor and the transformer. Sean and Niklas meanwhile implemented the network structures for feature extraction and the trajectory transformer, respectively. For the final training, evaluation and documentation we again worked collectively.

Lastly, we encountered the challenge to make training work in the first place with the hardware that was available to us. The limitation in computing power and machines to run training on forced us to omit some parts of our original code. For example, we implemented an additional Convolutional Neural Network (CNN) as a comparison to the ViT but were not able to involve it into our final results as we prioritized the ViT for our approach. In hindsight, we should have dealt with acquiring suitable hardware earlier as this was an issue that definitely limited our options in the end.

References

[1] N. Mine, *F1_2020_race_data*, 2020.
Accessed on: March 29, 2021. [Online].
Available: https://www.kaggle.com/datasets/coni57/f1-2020-race-data.

[2] X. Shen, M. Lacayo, N. Guggilla, and F. Borrelli, *ParkPredict+: Multimodal Intent and Motion Prediction for Vehicles in Parking Lots with CNN and Transformer*. arXiv, 2022.
[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, *Attention is all you need*, in Proceedings of the 31st International Conference on Neural Information Processing Systems, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., Dec. 2017, pp. 6000–6010
[4] E. Amirloo, A. Rasouli, P. Lakner, M. Rohani, and J. Luo, *LatentFormer: Multi-Agent Transformer-Based Interaction Modeling and Trajectory Prediction*. arXiv, 2022.
[5] Y. Yuan, X. Weng, Y. Ou, and K. Kitani, *AgentFormer: Agent-Aware Transformers for Socio-Temporal Multi-Agent Forecasting*. arXiv, 2021.
[6] A. Dosovitskiy et al., *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. arXiv, 2020.
[7] He, K., Chen, X., Xie, S., Li, Y., Dollár, P. & Girshick, R. Masked Autoencoders Are Scalable Vision Learners. (arXiv,2021), https://arxiv.org/abs/2111.06377