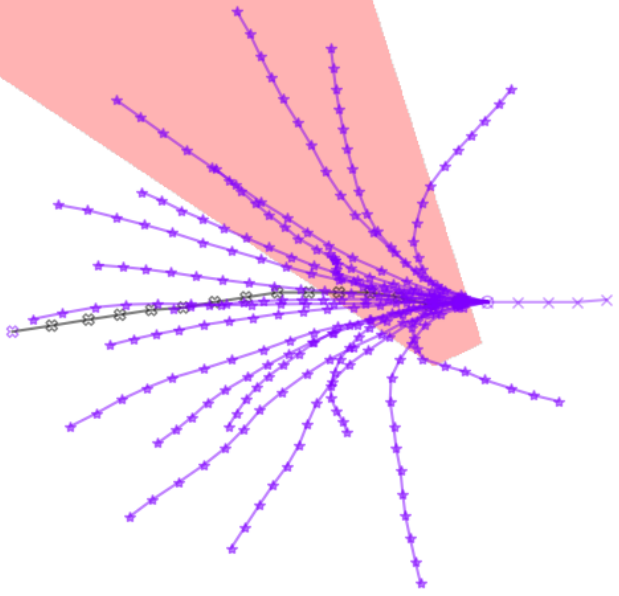
Toward Occlusion Capable Human Trajectory Prediction

Facilitating occlusion capability at the prediction stage of perception, with a TransFormer based trajectory prediction model

Paul Féry



Delft University of Technology

Toward Occlusion Capable Human Trajectory Prediction

Facilitating occlusion capability at the prediction stage of perception, with a TransFormer based trajectory prediction model

by

Paul Féry

student number: 4660625

Supervisors:	Dr. Ing. J. Kober
	A. Mészáros
Project Duration:	May, 2023 - October, 2024
Faculty:	Faculty of Mechanical Engineering, Delft

TU Delft

Summary

The following document is a report that discusses the research project which I conducted for my thesis, as part of my MSc in Robotics.

The majority of existing research within the trajectory prediction field makes the assumption that agents' past histories are perfectly observed. The focus of the research conducted in the project presented here is on the development of a trajectory prediction method which does not rely on this assumption, and actively accounts for the effects of occlusions on the prediction task. In the present research, the problem of trajectory prediction is reformulated in the following way: agents whose trajectories must be predicted may or may not be fully observed over the observation time window, and the prediction of agents whose most recent observations are *not current*, begins from their last available observation.

A prediction model has been developed in order to directly operate on the problem space considered (i.e., a prediction problem where complete observation of all present agents is not guaranteed). The proposed model, named OcclusionFormer, is based on an existing state-of-the-art prediction model, AgentFormer. This previous model is itself based on a TransFormer architecture, whose means of processing data is advantageous for handling occluded trajectories. Contrary to its predecessor, OcclusionFormer can be trained directly on occluded trajectory data. Notably, it predicts currently occluded agents' futures by beginning prediction from their last observed positions. An optional method of integrating knowledge of the *spatial configuration of occlusions* as an additional source of information to the prediction process has also been developed. The proposed prediction model can process an occlusion map as an additional input, extracting information about the visibility of the space, and using this information in its prediction.

In order to facilitate the study of occlusions and their effects on the prediction task, a simulator of occlusion events has been created. It relies on the placement of a virtual ego-perceiver, and a virtual occluding object, separating the workspace into two (complementary) regions: one visible to the ego, the other occluded. Occlusions generated by the simulator are not only defined temporally, but also spatially, thereby enabling the generation of occlusion maps, alongside with their corresponding occluded trajectory data. The occlusion simulator can be applied onto any trajectory dataset that is commonly available within the research field.

With the use of a popular trajectory dataset, experiments were conducted to study the performance of the proposed model under different circumstances. The results obtained from experiments reveal that OcclusionFormer is fit to operate on the prediction problem specified, with minimal performance decrease from AgentFormer under idealized, fully observed conditions. A study of the effects of occlusions on the model's performance is also provided, measuring the degree of degradation that occlusions induce on performance metrics. Experiments surrounding the integration of occlusion maps into the prediction model show that the approach taken results in performance improvements over some metrics, and deterioration over others.

Toward Occlusion Capable Human Trajectory Prediction

Facilitating occlusion capability at the prediction stage of perception, with a TransFormer based trajectory prediction model

Paul Féry
Department of Cognitive Robotics,
Faculty of Mechanical Engineering,
Delft University of Technology
Delft, the Netherlands

***Abstract*—A widely held assumption within the field of Trajectory Prediction is the perfect and complete observation of agents' pasts. While this assumption allows for a simpler representation of the prediction problem, it no longer holds true when prediction models are expected to operate on histories generated by upstream perception systems, which are susceptible to fail. Occlusions are a particularly important cause of perception failures. They fragment tracked agents' trajectories, and can often hide their most recent position(s) from the perception system. While most prediction models that are currently being researched cannot account for the possible incompleteness of agents' past histories, we devise a prediction model that is designed to directly operate on partially missing histories caused by occlusions. We present OcclusionFormer, a TransFormer based prediction model, which predicts agent's futures from their *last observed position*, without requiring imputation of missing past positions. Experiments show that our design is *occlusion capable*, as it can predict from trajectories with partially missing data, while remaining performant in the ideal, fully observed scenario. We also conduct research on the integration of an occlusion map within our model, which could help narrow the region of plausible prediction for occluded agents. We observe that, while the addition of such a map does improve the coherence of predictions with respect to the configuration of the occluded space, it results in a degradation of prediction performance.**

I. Introduction

In order to improve the safety of autonomous vehicles and robotic systems designed to operate in environments shared with other agents, it is important to ensure that the decision making processes of those systems is based on accurate and reliable perception of their environment. Trajectory prediction plays an important role in the overarching task of machine perception. It is a challenge that requires the consideration of many factors affecting the motions and behaviours of agents, such as the configuration of their environment, or the social interactions they might have with their neighbours.

One particular aspect that has received little attention from the research community is the possibility that past trajectories, which are the primary resource that prediction models can use to predict the future motions of agents, can be partially fragmented due to the imperfect detection and tracking of agents. Occlusions are a prominent cause of such incomplete histories. The great majority of the existing approaches within the trajectory prediction field neglect occlusions and the impact they may have on the prediction problem entirely. One important factor that participates in the retention of the assumption that trajectories are perfectly observable can be found in the datasets which present the prediction problem (such as the ATC Dataset [4], the ETH/UCY Dataset [17, 10], or the Grand Central Station Dataset [24]). Indeed, the trajectory datasets used within the field are designed to provide non-corrupt, fully observed trajectories. Occlusions are not well represented, and consequently, the responsibility to handle or mitigate them at the prediction stage of perception is a problem that remains largely unattended.

Our work is split into two main contributions. First, we choose to tackle the occlusion problem directly by redefining the prediction task, accounting for the possibility that agents' past histories may be incomplete, and notably, that the prediction of agents *currently* being occluded should start from their *last observed* state. We devise a prediction model to address this problem, by building on top of the AgentFormer model introduced by Yuan et al. [23], a state-of-the-art TransFormer based model, capable of accounting for social interactions among observed agents. Our implementation leverages positional encodings, which are used by TransFormer networks to inform the temporality of sequence data, and which facilitate the processing of partially missing input sequences. We show through our experiments that our model is capable of operating on partially missing input histories directly, predicting occluded agents' trajectories from their last observed timesteps, without requiring imputation. Our design also performs on par with its predecessor when deployed on completely observed trajectories, showing that occlusion capability can be implemented without necessarily inducing performance penalties in the idealized, fully observed case. Additional experiments reveal the impact that occlusions can have on the performance of prediction models, as well as highlight some problems that may arise from handling occlusions by means of imputation.

Our second contribution focuses on the potential benefits of informing our prediction model of the *spatial configuration* of occlusions. We recognise that occlusions are the product of a *partially unobserved workspace*, rather than an arbitrarily random source of trajectory corruption. In order to generate representations of occlusions, we devise an occlusion simulator, which can operate on any commonly available trajectory dataset. The simulator generates (currently occurring) occlusion instances by separating the workspace into a visible and an occluded region. We hypothesize that informing a prediction model about the visibility of the workspace could help narrow the field of estimation for the unknown states of occluded agents, thereby possibly improving prediction performance. We design an extension of our model, in which an occlusion map is provided as an additional input, and is processed alongside trajectory data through a specialized cross-attention process. This occlusion map is accompanied by an occlusion loss, which incentivizes the model to maintain predictions of unobserved past points within the occluded portion of the workspace. The experiments we conduct on the integration of the occlusion map show that our method is effective in improving the ability of the model to maintain predictions of occluded agents' unobserved pasts inside the occluded region of the workspace. However, our results show that this modification of the model degrades prediction performance.

II. Related Work

Trajectory Prediction is the last component of the machine perception pipeline. The task is most often formulated as a pattern recognition problem, where the objective is to anticipate agents' short term future motions, using their tracked past movements [19]. Important aspects of the problem include the capacity for models to account for social interactions among agents [1, 7] or environmental constraints and features of the agents' workspace [8, 16], or the formulation of prediction as a multi-modal task, allowing for multiple acceptable predictions for one given situation [7, 8, 9].

Occlusions are perhaps the greatest challenge within the machine perception field. They increase the difficulty of the detection and tracking tasks, resulting in potentially poor quality input data for the prediction task. Occlusions constitute a particularly dangerous risk factor for the perception of Vulnerable Road Users (VRU's) due to their size. Though the research fields of object detection and tracking assign a high priority to the problem of occlusion handling, occlusions receive minimal attention within the trajectory prediction field. The majority of existing work in trajectory prediction relies on the assumption of a perfectly observed environment, in which agents' past histories are not subject to occlusions [1, 7, 8, 16, 9]. The lack of extensive research on occlusions within the field can be partly explained by the datasets that are used as a basis for prediction [17, 10, 15, 18, 24, 4], as none of them explicitly present partially missing trajectories or occlusions, showing instead perfect data that is unlikely to be representative of tracklets generated online from an imperfect object detection/tracking system. Expecting perfect tracking performance is unreasonable. Most importantly, occlusion handling at the tracking level of perception focuses on re-identification after agent reappearance, with little capacity to address currently occurring occlusions. Such occlusions should therefore be addressed at the prediction level. Some of the existing research on "occlusion capable" trajectory prediction aims at enabling RNN architectures to manipulate irregular histories [3, 5]. Transformers [22] seem a more promising avenue, as they have the natural advantage of being directly fit to manipulate such sequences, thanks to their positional encoding [6, 23]. Mitigating the impact of occlusions can also be done by reducing the observation period [11, 20, 21], though this does not allow predictions for *currently occluded agents*. End-to-end approaches focus on encompassing all perception tasks in one system [14, 13, 12], which lowers the risks of inconsistency in the data being carried from one task to the next. However, end-to-end approaches tend to be inherently complex, making them both difficult to optimize and interpret.

AgentFormer [23] is a state-of-the-art, multi-modal prediction model capable of accounting for social interaction among agents. Based on a Transformer architecture, it is a promising candidate for an occlusion capable prediction model. The authors process input histories of all agents as one single sequence, specialising the attention mechanism of the network to let the model differentiate agent identities. This allows for modelling of social and temporal aspects of an input sequence simultaneously. In their supplementary material, the authors make note of the potential for their model to handle a time-varying number of agents, by omitting missing sequence elements from the input sequence. However, the authors' proposed approach does not account for the eventuality of generating predictions for *currently occluded agents*. Indeed, their model autoregressively decodes agents' future trajectories, starting from their current state, observed at timestep $t = 0$. Consequently, is is not fit to perform predictions on agents whose last observation was made before the current instant, as their current state is unavailable.

Our work focuses on the development of a method that directly accounts for occlusions as a part of the trajectory prediction problem. The design we propose is an expansion from AgentFormer, in which we enable the model's direct operation on trajectories which have been partially occluded. Notably, our design can perform prediction on occluded agents, whose current position is unknown by the perceiver system, by beginning predictions from agents' last observed state.

III. Methodology

This section will begin with a definition of the prediction task in Subsection III-A. We will then present the model we devise in order to address this problem in Subsection III-B.

A. Trajectory Prediction

Our goal is to perform multi-agent trajectory prediction with incomplete information caused by the partial occlusion of agents' trajectories. In a scene containing N individual agents, the state $\boldsymbol{x}_a^t \in \mathbb{R}^4$ describes the position and velocity on the ground plane for agent a, observed at timestep t. Observations of agents' pasts are made over a temporal window $t \in [-T_{\text{obs}}+1, 0]$. We assume the workspace $W \subseteq \mathbb{R}^2$ to be divided into two complementary regions: a visible region W_V and an occluded region W_O. When an agent lies within W_O, the state of that agent for that timestep is *unobserved*, making it unavailable as part of the input set $\boldsymbol{X}$. The input set is then the set of all non-occluded states over the past time window:

$$\begin{aligned} \boldsymbol{X} = \{\boldsymbol{x}_a^t \quad | \quad \texttt{Pos}\left(\boldsymbol{x}_a^t\right) \in W_V\}, \\ \forall t \in \{-T_{\text{obs}}+1, -T_{\text{obs}}+2, ..., 0\}, \\ \forall a \in \{1, 2, ..., N\} \end{aligned} \tag{1}$$

where $\texttt{Pos}\left(\bullet\right)$ returns the position. A "fully observed" environment would yield a complete set of input states for all N agents, as $W_V \equiv W$. Occlusion phenomena result in varying observation patterns among agents' histories. Notably, for each agent, we denote the **l**ast **o**bserved (i.e., non-occluded) timestep as $t_{\text{LO},\,a}$.

The objective of the task is to predict $\boldsymbol{Y}$, the sequences of unobserved positions of agents $\boldsymbol{y}_a^t \in \mathbb{R}^2$, from their last observed timestep $t_{\text{LO},\,a}$ up to an amount of future timesteps T_{pred}:

$$\begin{aligned} \boldsymbol{Y} = \left\{\boldsymbol{y}_a^t\right\}, \\ \forall t \in \{t_{\text{LO},\,a}+1, ..., T_{\text{pred}}\}, \\ \forall a \in \{1, 2, ..., N\} \end{aligned} \tag{2}$$

Note that, as agents may be undergoing occlusions with different durations and different last observed timesteps $t_{\text{LO},\,a}$, it is possible for $\boldsymbol{X}$ and $\boldsymbol{Y}$ to partly overlap temporally with each other (when some of the prediction sequence elements $\boldsymbol{y}_a^t$ exist within the past window, $[-T_{\text{obs}}, 0]$). The input states $\boldsymbol{X}$, target states $\boldsymbol{Y}$ and arrangement of the workspace into W_V and W_O constitute an "instance" which we can use to train our prediction model.

B. OcclusionFormer

The model we devise for our prediction scheme is based on AgentFormer by Yuan et al. [23]. As described in Section II, this model possesses promising characteristics for the prediction of occluded trajectories. However, their implementation of the model cannot operate on occluded trajectories, as their design relies on a strictly defined structure of the input/output data, which prevents processing of incomplete sequences. Adapting AgentFormer such that our research can be conducted imposes a significant overhaul of the model's implementation. We perform this adaptation of the architecture's internal operations, in order to enable the deployment of this model onto a problem space with partially missing input trajectories (more detailed explanations of our modification of the AgentFormer source code can be found in Appendix A). With our model capable of accepting irregular trajectories, further alterations of the model were conducted, which will now be discussed.

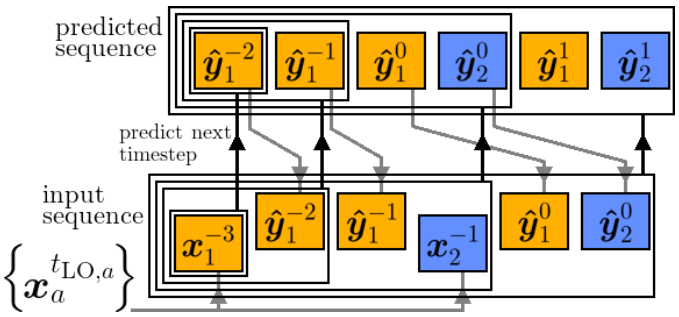


Fig. 1: OcclusionFormer's autoregressive decoding framework. Colours are used to highlight sequence elements belonging to different agents. Black upward arrows represent a pass through the decoder, generating from an input sequence the corresponding predicted sequence. Gray arrows indicate how individual input sequence elements are retrieved: each agent's first element in the sequence is their last observed state $\boldsymbol{x}_a^{t_{\text{LO},\,a}}$. Subsequent elements are retrieved from the predicted sequence.

1) Asynchronous Predictions from Irregular Histories: We modify the *decoder* module of the architecture by beginning the prediction for each individual agent from their last observed timestep $t_{\text{LO},\,a}$. Prediction begins at the earliest $t_{\text{LO},\,a}$ among agents; the decoder autoregressively completes the unobserved portion of the past sequence, inserting to its input sequence the last observed states of agents as it predicts subsequent timesteps (as shown in Figure 1). The prediction progresses over the future until reaching T_{pred}. Finally, this modification of the decoder module, and this new framework for performing predictions asynchronously imposes another modification on the reference frame used for the temporal encoding. The positional encodings of AgentFormer's different modules follow inconsistent temporal reference frames, encoding past and future sequence elements with disagreeing time codes. The temporal overlap between observations and predictions in our case, imposes that we adopt a consistent temporal reference frame throughout the entirety of the model. We do this by performing a *shift* in the assignment of time codes, such that elements at $t = -T_{\text{obs}}+1$ are encoded with

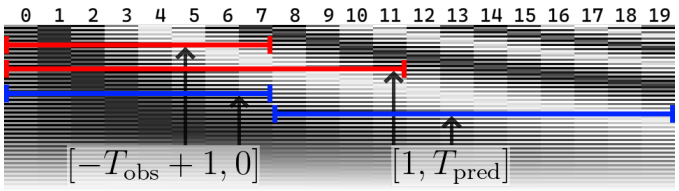


Fig. 2: How AgentFormer and OcclusionFormer respectively apply temporal encodings to a fully observed trajectory (the numbers above the displayed encoding range are the corresponding positional arguments *pos*).

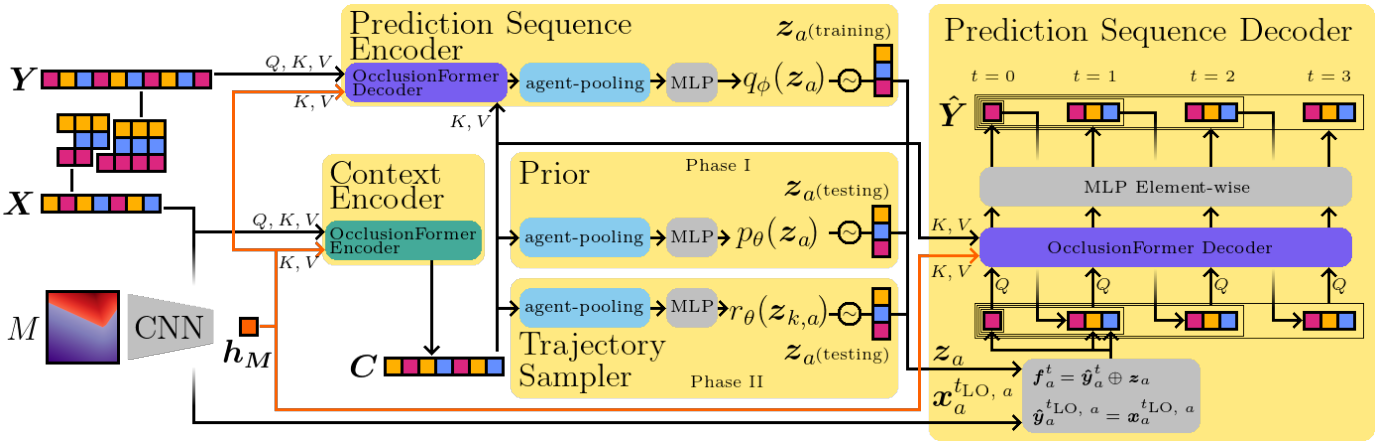


Fig. 3: An overview of the OcclusionFormer prediction model. The (optional) contribution of the occlusion map to the attention process of OcclusionFormer is highlighted by the orange arrows.

positional argument $pos = 0$ (see Figure 2). This shift prevents the generation of positional codes with a negative positional index (which, due to the usage of sine and cosine functions, are identical to their positive counterpart over exactly half of their dimensions, potentially increasing the difficulty of the learning process).

2) Learning with an Occlusion Map: The previously described changes will make our model "occlusion capable" (i.e., fit to operate on trajectory data as we define it in Subsection III-A). We would like to further investigate how the model's capacity to process occluded agents can be improved by leveraging additional information. More specifically, we assume that while an autonomous system might not have access to perfect information of surrounding agents due to limited visibility in its workspace, such a system could access (or generate a representation of) the visible space in its immediate vicinity. We hypothesise that a prediction model making use of such information could obtain better predictions for currently occluded agents, as knowledge of the occluded region W_O could be used to constrain the predictions over the unobserved past $[t_{\text{LO},\,a}, 0]$.

We devise our model to receive a representation of the workspace W in the form of an occlusion map, M, as an optional additional input. Similar to how Park et al. [16] encode their scene input, we define this occlusion map as the *distance transform* of a binary raster map, whose binary state assigns 0 to pixels within W_O, and 1 to those within W_V. As the goal of this addition is to help restrict the field of possible movement for currently occluded agents, the model must be capable of relating trajectory data to this new map data, as well as be encouraged to use such information efficiently.

The occlusion map M is first compressed by a CNN into a set of features $\boldsymbol{h}_M$. We project $\boldsymbol{h}_M$ to a key/value pair $\{\boldsymbol{K}_M, \boldsymbol{V}_M\}$, and project the trajectory features to a set of map-focused queries $\boldsymbol{Q}_M$, in addition to their already existing agent-aware attention related keys, queries and values $\{\boldsymbol{Q}_{\text{self}}, \boldsymbol{K}_{\text{self}}, \boldsymbol{Q}_{\text{other}}, \boldsymbol{K}_{\text{other}}, \boldsymbol{V}\}$, as they are defined in [23]. Concatenating the map-agent attention scores with the agent-aware attention scores, as well as the map value $\boldsymbol{V}_M$ with the trajectory values $\boldsymbol{V}$ allows us to compute output features for every input trajectory element. The combined map-agent and inter-agent attention process is illustrated in Figure 4. The interaction between occlusion map and trajectory features throughout this attention process allows for the model to account for the global occlusion configuration of the workspace.

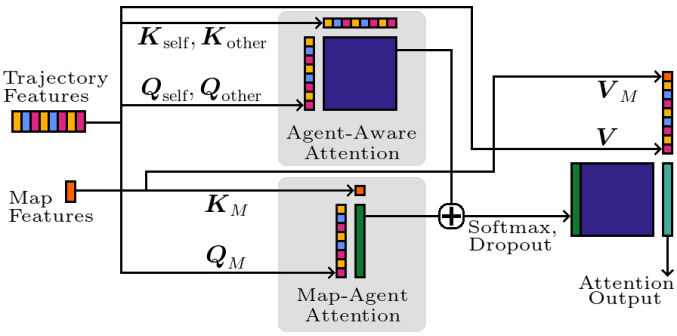


Fig. 4: The attention process we devise for OcclusionFormer allows for trajectory features to extract information about the occluded space from map features.

The resulting Map-Agent-Aware attention mechanism can simply replace Yuan et al.'s Agent-Aware attention when desired, by providing the set of occlusion map features $\boldsymbol{h}_M$ to all three attention modules of our OcclusionFormer model, as can be seen in Figure 3.

We also define an occlusion loss to incentivise the model to make productive use of the map. The occlusion loss is derived from the occlusion map M, and encourages the predictions' past segments to remain inside W_O, such that predictions remain coherent with the field of possible states constrained by the occluded space. Points predicted over the past are assigned a loss, equal to the square of the distance to the nearest point

inside W_O:

$$\mathcal{L}_{\text{occl.}} = \frac{\alpha_{\text{occl.}}}{\left|\hat{\boldsymbol{Y}}_{t\leq 0}\right|} \sum_{a=1}^{N} \sum_{t=t_{\text{LO, }a}+1}^{0} \text{distance}\left(\hat{\boldsymbol{y}}_a^t, W_O\right)^2 \quad (3)$$

where $\alpha_{\text{occl.}}$ is a weighing factor applied to the loss, and $\left|\hat{\boldsymbol{Y}}_{t\leq 0}\right|$ is the amount of points predicted over the past. The way the distance between prediction $\hat{\boldsymbol{y}}_a^t$ and the occluded region W_O is obtained is by computing the squared distance occlusion map M_{d^2} from M, and to retrieve the corresponding value at the location of the predicted point using bi-linear interpolation. When training our model to learn with the occlusion map, we apply the occlusion loss alongside the other loss components responsible for minimizing the difference between prediction sequences and ground truth. Those are the MSE loss, and the best-of-K samples loss:

$$\mathcal{L}_{\text{MSE}} = \frac{\alpha_{\text{MSE}}}{|\boldsymbol{Y}|} \sum_{a=1}^{N} \sum_{t=t_{\text{LO, }a}+1}^{T_{\text{pred}}} \gamma(t) \cdot \left\|\hat{\boldsymbol{y}}_a^t - \boldsymbol{y}_a^t\right\|^2 \quad (4)$$

$$\mathcal{L}_{\text{sample}} = \frac{\alpha_{\text{sample}}}{|\boldsymbol{Y}|} \sum_{a=1}^{N} \min_k \sum_{t=t_{\text{LO, }a}+1}^{T_{\text{pred}}} \gamma(t) \cdot \left\|\hat{\boldsymbol{y}}_{k,a}^t - \boldsymbol{y}_a^t\right\|^2 \quad (5)$$

with:

$$\gamma(t) = \begin{cases} \gamma_1 & \text{if} \quad t \leq 0 \\ \gamma_2 & \text{if} \quad t > 1 \end{cases}$$

We modify the MSE and sample losses such that individual points' errors can be weighed by a factor γ depending on whether the corresponding timestep is in the past, or in the future. This allows for control of the severity of ground truth penalties attributed to past points, as they are also being penalised by the occlusion loss. Following the procedure defined by Yuan et al. [23], our training process is performed in two subsequent phases; the first for training of the model itself, and the second for training of a latent code sampler, to improve prediction diversity. The total losses the model is subjected to during each phase are:

$$\begin{aligned} \mathcal{L}_{\text{phase I}} = \mathcal{L}_{\text{MSE}} + \mathcal{L}_{\text{occl., MSE}} + \\ \mathcal{L}_{\text{sample}} + \mathcal{L}_{\text{occl., sample}} + \mathcal{L}_{\text{KL}} \end{aligned} \quad (6)$$

$$\mathcal{L}_{\text{phase II}} = \mathcal{L}_{\text{sample}} + \mathcal{L}_{\text{occl., sample}} + \mathcal{L}_{\text{KL}} + \mathcal{L}_{\text{diverse}} \quad (7)$$

where $\mathcal{L}_{\text{KL}}$ denotes the Kullback-Leibler loss between prior and posterior latent codes distributions, and $\mathcal{L}_{\text{diverse}}$ denotes the diversity loss which penalizes small pairwise distances between the K predictions made, as they are defined by Yuan et al. [23].

IV. Experimental Setup

A. Occlusion Simulator

Commonly available datasets for trajectory prediction tasks only present fully observed trajectories. In order to enable the study of our specific formulation of the problem, we must be capable of representing trajectories subject to occlusions, which are defined both temporally and spatially.

Fig. 5: An example of a scene generated from our occlusion simulator. The red area constitutes W_O, and is obtained from computing the visibility polygon of the virtual ego, obstructed by a virtual occluder (respectively shown as a red dot located below W_O, and a dark blue segment at its lower boundary). Trajectories are shown as sequences of points: full lines connect trajectory points which are in the past, and dashed lines indicate the future part of the trajectory.

We build a simulator of occlusion events (see Figure 5), which can be applied on any available trajectory prediction dataset. The core principle of this simulator is the placement of two objects in the scene: a "virtual ego" (modelled as a point), whose visibility of the scene is partially hindered by a "virtual occluder" (modelled as a segment delimited by two points). Focus is placed on generating *currently occurring* occlusions among the present agents (see Appendix D for a detailed description of the simulation process).

B. Dataset

We evaluate our method on the Stanford Drone Dataset (SDD) [18], as it is a large trajectory dataset, focusing primarily on VRU's (which are most susceptible to be subjects of occlusions). SDD contains around 20000 trajectories, obtained from 60 aerial videos captured at 30 Hz over 8 different places on the campus of Stanford.

C. Stanford Drone Occlusion Dataset

We remove non-VRU trajectories from the dataset (keeping only *pedestrians*, *cyclists* and *skateboards*). Following the procedure indicated by Andle et al. [2], we also remove "lost" annotations and keep the first portion of fragmented trajectories. We convert trajectory coordinates from pixels to metres (see Appendix C). We define train/validation/test sets by randomly splitting the 60 videos with assignment proportions 42/9/9. We sample trajectories at a frequency of 2.5 Hz. We execute the occlusion simulator over the dataset 3 separate times, increasing the effective size of our dataset, and allowing for different occlusion scenarios for each sampled time window $[-T_{\text{obs}}; T_{\text{pred}}]$. Some summary statistics about the resulting "Occlusion Simulation" dataset can be seen in Table I (under the **"Complete"** column).

TABLE I: Summary statistics of our different dataset splits.

		Complete	Difficult
Training Split	split size	85776	2926
Validation Split	split size	19653	510
	effective size*	1250	510
Test Split	split size	11751	340
	trajectories	103191	3813 (353 difficult.)
	$t_{\text{LO},a} = -0$	91220	N/A
	$t_{\text{LO},a} = -1$	2336	172
	$t_{\text{LO},a} = -2$	2296	25
	$t_{\text{LO},a} = -3$	2458	18
	$t_{\text{LO},a} = -4$	2158	32
	$t_{\text{LO},a} = -5$	2228	83
	$t_{\text{LO},a} = -6$	495	23

The subset of *currently occluded trajectories* in the test split amounts to a total of 11971 (11.6% of all 103191 trajectories). To facilitate comparison between different modalities of trajectory visibility, we generate a copy of the test split, with the occluded trajectories remaining fully observed (which we call the "Fully Observed" dataset). Part of the evaluation will be focused on investigating the effects of imputation on prediction performance. For our purposes, we also generate a copy of the test set with occluded trajectories imputed according to the following strategy: linear interpolation of incomplete trajectory fragments bounded by two observed positions, and extrapolation of trajectories unobserved until $t = 0$, assuming constant velocity from their last observed position.

D. Difficult Occlusions

Initial experiment results led us to the conclusion that the vast majority of occlusion cases provided by our simulator might be restricting the visibility of agents' past trajectories in such a way that they might not sufficiently restrict the space of possible movement (Those results will be discussed in Subsection V-B). Occlusion cases with exceedingly large occlusion zones, where the possible movement space of occluded agents remains wide, present a poor amount of additional information that a prediction model could make use of when infering agent's motions up to their current state. The occlusion dataset containing a majority of such cases could be problematic for training our model, as a majority of uninformative occlusion zones might add difficulty to the model's capacity to learn from the occlusion map. We adapt to this reality by identifying a subset of occlusion cases, which are susceptible to contain meaningful information to provide to our prediction model. Such a subset is obtained by running a Constant Velocity predictor over our occlusion dataset, and finding the occlusion cases for which the predictor fails to place the current state of the occluded agent $\hat{\boldsymbol{y}}_a^0$ inside W_O. Those cases are susceptible to present an informative configuration of the occlusion space, as predictions making use of an occlusion map are expected to produce coherent predictions for occluded agents, maintaining their positions inside the occlusion zone. We describe those occlusion cases as "difficult occlusions", and the subset of our dataset containing all of those cases as the "Difficult Subset". Statistics about the difficult subsets obtained from our dataset splits are available in Table I. The difficult subset of our test split contains a total of 600 occluded trajectories, of which 353 are difficult occlusion cases.

With all variants of our dataset presented, the following abbreviations will be used in the following sections to facilitate the identification of experiments' train/test environments:

- **FO** : **F**ully **O**bserved dataset
- **OS** : **O**cclusion **S**imulation dataset
- **OS, I** : **O**cclusion **S**imulation dataset, **I**mputed
- **DS** : **D**ifficult **S**ubset

E. Experiments

We establish different versions of our model in order to evaluate how individual components and alterations affect its behaviour and performance. Each row in Table II describes an individual instance of a model, run using the indicated training data. To facilitate referencing, every model instance is identified by a unique roman numeral.

TABLE II: The list of experiments conducted. The **"Train Data"** column indicates which dataset variant was used for training.

#	Model	Train Data	Description
I	AgentFormer	FO	The original implementation by Yuan et al. [23].
II	OcclusionFormer	FO	Our implementation of OcclusionFormer.
III	OcclusionFormer	OS	The same model as #II.
IV	OcclusionFormer	DS	The same model as #II.
V	O.Former w/ occlusion map *A*	DS	The occlusion loss is applied as an additional penalty to points predicted in the past (see Appendix B).
VI	O.Former w/ occlusion map *B*	DS	The loss over the past is equally split between the occlusion losses and the ground truth losses (see Appendix B).

The Fully Observed / Occlusion Simulation datasets are significantly larger than the ETH/UCY dataset originally used by Yuan et al. Consequently, we choose to adapt the training process: when training on the "FO" or "OS" datasets, we perform 3 epochs (of 85776 batches, see Table I), performing validation and model checkpoint saving every 5000 batches, with a validation split of size 1250 (defined by sampling a fixed subset of the total validation split), in order to ensure a training/validation ratio of $80/20$. When training with the "DS" dataset, our number of epochs is 90 (with 2926 batches, see Table I), and our validation split size is 510. After completion of the training procedure, we keep the model checkpoint with the best validation loss for evaluation. We keep hyperparameter values constant across all experiments. We set the observation and prediction time windows to $T_{\text{obs}} = 8$ and $T_{\text{pred}} = 12$. The experiments ran for the investigation of the potential benefits of our occlusion map processing scheme are models #V and #VI (both described as "O.Former w/ occlusion map"). Every other model we prepare for our experiments is *not* making use of the occlusion map + occlusion loss combination. When

making use of the occlusion map, we require a constant resolution. We fix the scene side length to 80 m, and an occlusion map resolution of 800 px, with the center of the scene located at the mean of all present agents' last observed position. During training, we perform random rotation on the scene. We follow Yuan et al's [23] maximum number of agents per scene of 32, pruning away surplus agent trajectories by prioritising those lying furthest from the mean of all agents' last observed position.

F. Performance Metrics

ADE and FDE are the most commonly used metrics when it comes to evaluating the performance of coordinate-based prediction models. For a given prediction $\hat{\boldsymbol{Y}}$ made over a future time window $[1, T_{\text{pred}}]$, the scores are obtained by comparison against the ground-truth $\boldsymbol{Y}$ in the following way:

$$\text{ADE} = \frac{1}{T_{\text{pred}}} \sum_{t=1}^{T_{\text{pred}}} \left\| \hat{\boldsymbol{y}}^t - \boldsymbol{y}^t \right\|_2 \tag{8}$$

$$\text{FDE} = \left\| \hat{\boldsymbol{y}}^{T_{\text{pred}}} - \boldsymbol{y}^{T_{\text{pred}}} \right\|_2 \tag{9}$$

As we generate multiple prediction modes per agent, we denote the average ADE and FDE value over those K modes by MEANADE and MEANFDE, respectively, while denoting the scores for the best mode by MINADE and MINFDE.

Our particular framework generates prediction sequences which are partly defined in the unobserved past of occluded agents. Those past sections of the prediction are not tracked by the traditional ADE/FDE scores. Consequently, we also define new versions of those metrics, focusing on the past portion of the prediction:

$$\text{ADE}_{\text{past}} = \frac{1}{|t_{\text{LO, }a}|} \sum_{t=t_{\text{LO, }a}+1}^{0} \left\| \hat{\boldsymbol{y}}_a^t - \boldsymbol{y}_a^t \right\|_2 \tag{10}$$

$$\text{FDE}_{\text{past}} = \left\| \hat{\boldsymbol{y}}^0 - \boldsymbol{y}^0 \right\|_2 \tag{11}$$

Additionally, when it comes to studying the extent to which the past portion of predictions are coherent with respect to the disposition of the occluded region W_O, we devise the following metrics. First, the "Occluded Area Count" ($\text{OAC}_{t=0}$), measures the number of prediction modes which (correctly) place an occluded agent's position at $t = 0$ inside W_O. Second, the "Occluded Area Occupancy" (OAO), measures the proportion of predicted past points lying within W_O. The metrics proposed here are an adaptation of Park et al.'s DAO/DAC metrics [16], which aim at measuring the extent to which predictions for vehicles remain inside the driveable space.

$$\text{OAC}_{t=0} = \frac{K - m_{t=0}}{K} \tag{12}$$

with $m_{t=0}$ equal to the number of modes with predicted position at $t = 0$ outside of W_O.

$$\text{OAO} = \frac{\left|\text{traj}_{\text{px}}\right|}{K \cdot |t_{\text{LO, }a}|} \tag{13}$$
$$\forall \quad k \in [1, 2, ..., K],$$
$$\forall \quad t \in \{t_{\text{LO, }a} + 1, t_{\text{LO, }a} + 2, ..., 0\}$$

with:

$$\text{traj}_{\text{px}} = \left\{ \hat{\boldsymbol{y}}_k^t \quad | \quad \hat{\boldsymbol{y}}_k^t \in W_O \right\}$$

Higher $\text{OAC}_{t=0}$ and OAO scores indicate a better compliance with the occlusion map.

V. Results

The evaluation of our model, and of the aspects affecting its "occlusion capability", will focus on 4 questions:

- Are there costs associated to enabling our model to predict under occluded conditions?
- How impactful are occlusions on the prediction performance of our model?
- How does imputation affect the prediction task?
- Can prediction performance be improved by learning with an occlusion map?

A. Are there costs associated to enabling our model to predict under occluded conditions?

The design modifications brought to the AgentFormer model allow for its deployment in environments containing partially observed trajectories, without having to rely on any imputation preprocessing step. Naturally, expanding the operational design domain of any system is likely to incur some penalties to that system's performance, or to impose certain restrictions or limitations on its operation.

TABLE III: Models ran on fully observed trajectories, ranked by lowest MINFDE. Values are expressed in meters.

		MIN (K=20)		MEAN (K=20)	
#	**Model**	**ADE**	**FDE**	**ADE**	**FDE**
I	AgentFormer	0.405	0.712	2.171	4.735
I	AgentFormer	0.409	0.720	2.184	4.704
II	OcclusionFormer	0.407	0.720	2.262	4.849
I	AgentFormer	0.407	0.725	2.142	4.674
I	AgentFormer	0.411	0.726	2.212	4.755
I	AgentFormer	0.413	0.727	2.245	4.872
II	OcclusionFormer	0.412	0.732	2.231	4.815
II	OcclusionFormer	0.415	0.733	2.207	4.744
II	OcclusionFormer	0.420	0.748	2.234	4.788
II	OcclusionFormer	0.418	0.749	2.281	4.949
I	*Avg. AgentFormer*	*0.409*	*0.722*	*2.191*	*4.748*
II	*Avg. OcclusionFormer*	*0.414*	*0.736*	*2.243*	*4.829*
	Relative increase (%)	*+1.33*	*+1.99*	*+2.40*	*+1.71*

As will be observed throughout the following subsections, OcclusionFormer is an occlusion capable prediction model. In order to study the effects of the design differences between AgentFormer and OcclusionFormer which are responsible for the expansion of the problem space, we compare models #I and #II (see Table II), running 5 copies of each with different RNG seeds to obtain a baseline of their performance. The results are reported in Table III.

TABLE IV: The performance summary table of our different experiments. Results shown are obtained by aggregating scores on the subset of agents which were subject to currently occurring occlusion events within our Occlusion Simulation dataset (11971 trajectories). ADE/FDE scores are expressed in meters.

	#	Model	Train Data	Test Data	MIN ADE	MIN FDE	MEAN ADE	MEAN FDE	MIN_{past} ADE	MIN_{past} FDE	$\text{MEAN}_{\text{past}}$ ADE	$\text{MEAN}_{\text{past}}$ FDE	OAO	$\text{OAC}_{t=0}$
a.	*I*	*AgentFormer*	*FO*	*FO*	*0.495*	*0.868*	*2.384*	*5.160*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*
b.	*II*	*OcclusionFormer*	*FO*	*FO*	*0.503*	*0.885*	*2.465*	*5.312*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*
c.	*II*	*OcclusionFormer*	*FO*	*OS*	*1.482*	*1.503*	*4.487*	*7.671*	*0.797*	*0.993*	*1.396*	*1.532*	*0.826*	*0.849*
d.	III	OcclusionFormer	OS	OS	0.830	1.285	4.096	7.315	0.147	0.183	0.497	0.827	0.900	0.924
e.	IV	OcclusionFormer	DS	OS	0.943	1.434	4.540	7.756	0.210	0.241	0.721	1.143	0.874	0.879
f.	V	O.Former w/ map A	DS	OS	1.030	1.529	4.853	8.302	0.356	0.368	0.833	1.248	0.910	0.904
g.	VI	O.Former w/ map B	DS	OS	0.985	1.499	4.017	7.049	0.415	0.471	0.642	0.911	0.979	0.981
h.	*II*	*OcclusionFormer*	*FO*	*OS, I*	*0.866*	*1.182*	*2.870*	*5.650*	*0.297*	*0.476*	*0.297*	*0.476*	*0.950*	*0.966*

When it comes to the differences between AgentFormer and OcclusionFormer, we notice a slight performance loss from our implementation with respect to the original. The increase in the average of ADE/FDE scores from AgentFormer to OcclusionFormer is in the order of a few percents. Multiple factors could play a role in this difference in performance. The implementation of AgentFormer and OcclusionFormer differ in the following way:

- Certain internal operations within the architecture have been changed to adapt to a new input/output data paradigm (see Appendix A).
- OcclusionFormer's autoregressive decoder performs predictions asynchronously, by appending agents' last observed states $\boldsymbol{x}_a^{t_{\text{LO},\,a}}$ appropriately to the prediction sequence (see Subsubsection III-B1).
- The temporal encodings used to embed input trajectory features are different (see Subsubsection III-B1).
- Some of the loss components' normalization has been altered (i.e., due to a varying number of predicted timesteps per agent, ground truth losses are no longer normalized by number of agents), which has been mitigated by a proper adaptation of loss weights (see Appendix B).

All of those aspects could participate in a minimal change in the models' behaviour while training. Aside from those differences at the architecture level, the possibility of chance playing a role in the performance difference cannot be eliminated completely: performing an unequal variances t-test on the MINADE/MINFDE/MEANADE/MEANFDE scores of the AgentFormer results versus the OcclusionFormer results only yields a statistically significant difference for the MEANADE scores (the p-values of those t-tests being: 0.09097/0.054518/0.045035/0.132679). Running additional instances of the models would help in improving the statistical power of those tests. The apparent performance penalty caused by the adaptation of the model architecture remains minimal.

B. How impactful are occlusions on the prediction performance of our model?

In our evaluation of OcclusionFormer's performance, we would like to dedicate some of our attention specifically to the model's occlusion capability. Namely, we would like to investigate in detail the effects that occlusions can have on our model's prediction performance. To this end, we focus our attention on the subset of agents who are subject to *currently occurring occlusion events* within our Occlusion Simulation dataset. We evaluate our OcclusionFormer model, with different training and testing conditions, and report the results under Table IV.

Effects of occlusions on ADE/FDE scores. We observe an average increase of MINADE/MINFDE by $+65\%/+45\%$ for OcclusionFormer when moving from a train/test environment with trajectories fully observed, to a train/test environment where trajectories are occluded (i.e., comparing models #II and #III, rows *b.* and *d.* in Table IV). Similar score increases are observed for MEANADE/MEANFDE: $+66\%/+38\%$. The performance degradation naturally comes from the lower quantity of available information for the model to leverage for its prediction, and from the extension of the prediction duration. Notably, with our simplest version of the model operating on occluded data (i.e. model #III, row *d.*), the best prediction mode on average will *misplace* occluded agents by 18.3 cm when estimating their current position (the error in the estimation of the current state is directly measured by $\text{MINFDE}_{\text{past}}$/$\text{MEANFDE}_{\text{past}}$).

Prediction coherence with respect to occlusions. When it comes to compliance with the occlusion map, we remark that OcclusionFormer naturally performs quite well, showing $\text{OAO}/\text{OAC}_{t=0}$ scores of $90.0\%/92.4\%$, without relying upon an occlusion map (see row *d.*). This indicates that our simulator has a tendency to produce occlusions that are "poorly informative" for prediction; without the help of the occlusion map to enforce good map compliance, the model already performs very well in keeping agents within the occlusion zone. A majority of poorly informative occlusion cases is a problem for training models intending to leverage information from an occlusion map. Indeed, the learning process of such models is susceptible to being hindered by a training environment where the usefulness of the occlusion map is not made sufficiently salient. Consequently, the decision has been made to facilitate the learning process of our occlusion map processing model(s) (i.e., models #V and #VI), by training them on a set of more informative occlusion cases, prompting the specification of the Difficult Subset (as explained in Subsection IV-D). Discussions on how to better address this issue of poorly informative occlusion cases in future work can be found in Subsection VI-B.

Fully observed and occluded trajectories: a distributional shift. In our evaluation, we also consider the case of transferring OcclusionFormer from a training environment containing no occlusions to a deployment environment with occluded trajectories (i.e., observing model #II, tested on the "OS" dataset, Table IV, row *c.*). This particular case shows very poor performance over every metric, caused by the significant change in the distribution of trajectory data when transitioning from the fully-observed domain to the occluded one. Indeed, the decoder module(s) of a Transformer model trained exclusively on fully observed trajectories will be subject to never-before-seen input positional embeddings when generating predictions for occluded agents, naturally resulting in an ill-fitted model for this data. Training Transformer models to predict from agents' *last observed states* requires a sufficient proportion of unobserved past states in the training data, in order to properly facilitate the training of the decoders. Generating occlusions by random dropout of elements within $\boldsymbol{X}$ is a way of generating occlusions which could provide better control over this proportion than our simulator approach.

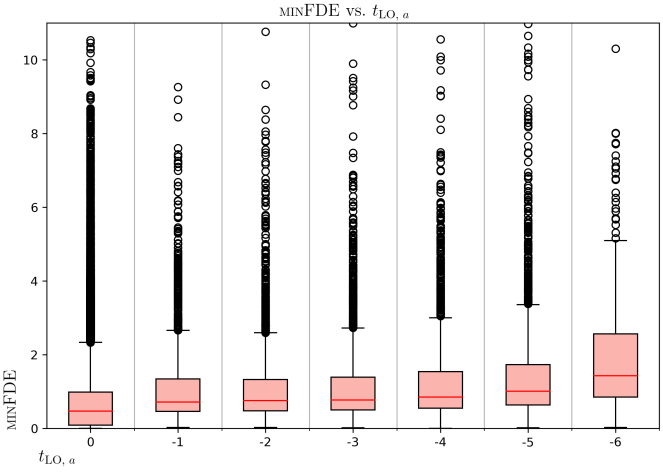


Fig. 6: Boxplots showing the evolution of the minFDE score for OcclusionFormer (model #III).

Occlusion durations and prediction performance. Of course, the extent to which occlusions impact the performance of OcclusionFormer is directly dependent on the value of $t_{\text{LO},\,a}$. In order to further examine how occlusion duration impacts prediction performance, we observe the behaviour of performance scores along prediction groups defined by $t_{\text{LO},\,a}$, as can be seen in Figure 6 (additional boxplot figures can be found in Appendix F). We can observe the distribution of minFDE scores progressively increase as the last observed timestep $t_{\text{LO},\,a}$ recedes further into the past. two particular phenomena seem to make themselves apparent on this figure: an unusually large "performance shift" between $t_{\text{LO},\,a} = \{0, -1\}$, and another between $t_{\text{LO},\,a} = \{-5, -6\}$. In order to properly assess the effects of occlusion duration on prediction performance, we must isolate any eventual extraneous factor that might play a role in these shifts. An investigation of such factors revealed that a partial cause for the shift between $t_{\text{LO},\,a} = \{0, -1\}$ is the fact that, by design of our occlusion simulator, all still-standing agents are inherently fully observed, and therefore fall within the $t_{\text{LO},\,a} = 0$ category. No clear cause could be identified for the $t_{\text{LO},\,a} = \{-5, -6\}$ shift, but we suspect that the occlusion simulator might be implicitly selecting only certain trajectories to fall within the $t_{\text{LO},\,a} = -6$ category (the investigations undertaken surrounding these performance shifts are discussed in more detail in Appendix F). We acknowledge the implicit partiality with which the occlusion simulator might assign occlusion patterns to trajectories, and choose to narrow our analysis on the effects of occlusion duration on prediction to the range $t_{\text{LO},\,a} \in [-5, -1]$. We perform linear regressions of the average ADE/FDE scores obtained over trajectory sets categorized by $t_{\text{LO},\,a}$ within this range for OcclusionFormer (model #III), and show the results in Table V. From this, we observe that performance scores degrade fairly linearly with respect to the duration of the occlusion event, with every additional unobserved timestep increasing minADE/minFDE scores by 9 cm, and meanADE/meanFDE scores by 41cm. We also remark that an increase in occlusion duration by 1 timestep increases $\text{minADE}_{\text{past}}$/$\text{minFDE}_{\text{past}}$ by 4cm/6cm, and $\text{meanADE}_{\text{past}}$/$\text{meanFDE}_{\text{past}}$ by 13cm/29cm, respectively.

TABLE V: linear fit of the average performance scores of OcclusionFormer (model #III), aggregated over identities categorized by $t_{\text{LO},\,a}$ within the range $[-1, -5]$.

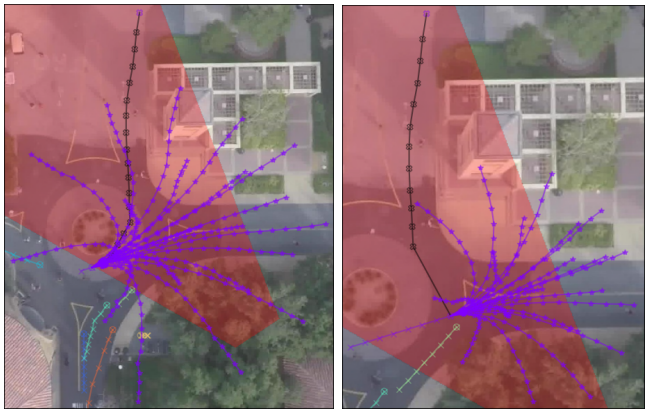
	$t_{\text{LO},\,a}$		**Linear Fit:** $t_{\text{LO},\,a} \in [-5, -1]$		
Score	-1	-5	m	b	R^2
minADE	0.6496	0.9992	0.0868	0.5483	0.9866
minFDE	1.1178	1.4786	0.0880	0.9944	0.9316
meanADE	3.2425	4.8900	0.4085	2.7707	0.9870
meanFDE	6.4624	8.1313	0.4091	5.9465	0.9598
$\text{minADE}_{\text{past}}$	0.0588	0.2122	0.0374	0.0241	0.9970
$\text{minFDE}_{\text{past}}$	0.0588	0.2948	0.0592	–0.0060	0.9969
$\text{meanADE}_{\text{past}}$	0.2332	0.7420	0.1272	0.0910	0.9945
$\text{meanFDE}_{\text{past}}$	0.2332	1.4016	0.2933	–0.1007	0.9935

Occlusions play an undeniably important role in the degradation of prediction models' performance. We argue that the trajectory prediction research field would benefit from focusing more on occlusions within the prediction task, and dedicate a greater amount of the research effort towards finding means to address occlusions directly at the prediction stage of the autonomous perception process.

C. How does imputation affect the prediction task?

Partially missing trajectories caused by occlusions do not necessarily impose on prediction models the responsibility of being compatible with fragmented data, in order to ensure the proper handling of occlusions within the prediction stage. Another typical approach to any problem involving missing data is imputation: rather than making the model capable of directly processing missing data, the incompatibility between model and data can be solved by reconstructing the missing parts of the input data before feeding it to the model. The imputation process then carries the responsibility of making

the solution occlusion capable, allowing the problem definition of the prediction task to remain in its simpler form (i.e., maintaining the assumption of complete past histories). We recognise that accounting for occlusions directly within the prediction problem's definition (Subsection III-A) will increase the complexity of possible solutions, and will reduce the amount of suitable approaches (e.g., preventing the use of regular RNNs, which cannot directly handle missing data).



(a) #III: Occluded trajectory. (b) #II: Imputed trajectory.

Fig. 7: Qualitative comparison of predictions made by our model, using differently preprocessed input trajectory data. Predictions are shown for the purple agent only, as small stars connected by colored lines. All trajectory sequences which are part of the input set $\boldsymbol{X}$ are shown as crosses connected by colored lines. The hollow crosses connected by a black line correspond to the ground truth sequence of the purple agent.

Consequently, we are interested in evaluating how a simple imputation process might impact our model's performance. We test OcclusionFormer's performance against imputed trajectories, using the imputation scheme we described in Subsection IV-C. The results we obtain from this experiment (i.e. model #II, tested on "OS, I", Table IV, row *h.*) reveal that, while OcclusionFormer performs similarly between imputed and occluded conditions in terms of MINADE/MINFDE scores (comparing rows *d.*, *h.*), MEANADE/MEANFDE show significantly lower values than those obtained from the model operating on occluded trajectories. It is important not to interpret those results as evidence that imputation here is an objectively better approach, and that reframing the prediction task as an operation that begins from agents' last observed state should be dismissed: indeed, lower MEAN scores for the imputation case are at least partly explained by the effective reduction of the prediction horizon (i.e., every prediction starts from $t = 0$, instead of $t = t_{\text{LO, }a}$). Consequently, imputation effectively induces a bias on the prediction: the model is denied the permission to explore the field of possible unknown states of occluded agents. This can be observed clearly in Figure 7: the imputation misplaces the current state of the agent far away from its last observed timestep, while also reducing the space the model can explore for its predictions.

D. Can prediction performance be improved by learning with an occlusion map?

We will now focus on the performance of our model when provided with the additional information of an occlusion map. We devised two different configurations of OcclusionFormer making use of the occlusion map. Version *A* (model #V) applies the occlusion loss *in addition* to AgentFormer's traditional losses, while version *B* (model #VI) effectively splits the "loss budget" for points predicted over the past equally between occlusion losses and ground-truth losses (the loss configuration of both versions is provided in more detail in Appendix B). As those two models were trained with the "Difficult Subset" (as explained under Subsection IV-D and Subsection V-B), we also produce model #IV, a run of OcclusionFormer, without any occlusion map processing, trained under the same conditions (i.e., an exact copy of #III, trained on the "DS" dataset). With this model, we can measure eventual performance changes induced by training on the "DS" dataset, as well as provide a baseline for a fair comparison against the two "occlusion map aware" models (i.e., #V and #VI).

Performance effects of the Difficult Subset, and of the parameterization of the occlusion loss. Comparing the scores of models #III and #IV (Table IV, rows *d.* and *e.*), it can be remarked that training on the Difficult Subset implies a significant performance drop across all scores, likely caused by an impoverishment of diversity in the training data, due to the reduction in dataset size. This significant difference in performance shows the importance of using model #IV as the baseline for studying the effects of introducing the occlusion map as an additional input within our model. When it comes to comparing both occlusion map aware model versions against one another, we can see from Table IV that model #VI is superior to #V, across all measured performance scores (see rows *f.* and *g.*). Qualitatively, model #VI also seems to have better internalised the objective of maintaining past predictions inside W_O, as can be observed from Figure 8. This could be explained by the occlusion loss being proportionately more important in #VI than in #V. The difference in performance between both versions is significant; perhaps even better results could be obtained from further fine-tuning of the model's hyperparameters. However, for reasons that will be discussed over the course of this subsection, we choose not to conduct an extensive hyperparameter search. The continuation of our discussion will focus on the performance of model #VI, the better of the two occlusion map aware experiments.

Learning with an occlusion map: improvements and degradations. Comparing model #VI against its "occlusion map unaware" homologue #IV (see Table IV, rows *e.* and *g.*), we observe that the processing of the occlusion map does not result in an overall improvement of OcclusionFormer's performance. The introduction of the occlusion map did result in an improvement of $\text{OAO}/\text{OAC}_{t=0}$ scores by $\approx 10\%$, and a positive reduction of MEANADE/MEANFDE scores by 0.52 m, and 0.71 m, respectively. Those positive changes in the

(a) #IV: OcclusionFormer (b) #V: O.Former w/map A (c) #VI: O.Former w/ map B

Fig. 8: Comparisons of predictions from models #IV, #V and #VI, generated for an occluded agent. The past portions of predicted sequences (where $t \leq 0$) are highlighted in opaque.

model's performance seem to indicate that the model is successfully relating the information from map and trajectory data sources in order to produce predictions for occluded agents that are respectful of the configuration of the occluded region. A model that integrates the information of the occlusion map perfectly should, in theory, be capable of producing prediction sets of higher accuracy, by narrowing the field of occluded agents' likely motions. In terms of performance, improved MEAN scores can be an indicator of improved accuracy. Interestingly, MEAN performance only improved over the set of occluded agents: when making the same comparison using the subset of agents who have been fully observed, both MIN and MEAN scores of the model making use of the map are worse than those of the model not using it (see Table IX, in Appendix E). As intended, the occlusion map is specifically affecting the prediction of occluded agents.

In contrast to the improved MEAN scores and occlusion map compliance scores, the effects of the occlusion map processing on the MINADE/MINFDE scores is negative, increasing by 0.04 m, and 0.07 m, respectively. Our application of the occlusion map and occlusion loss to the prediction framework is not without drawbacks. The occlusion map processing strategy was designed with the underlying intuition that, as the movements of occluded agents must have remained within the unobserved portion of the workspace W_O, it would be beneficial to design a prediction model that "prefers" those possible scenarios over the (less likely) set of trajectories that should have resulted in the occluded agent becoming visible again. However, in effect, it would seem that our implementation did not result in the model developing a preference for a subset of fitter trajectory candidates from the distribution of all possible predictions. Instead, the effects on the model's predictive behaviour are better interpreted as a significant change in the prediction distribution, in order to conform to the added objective of occlusion compliance imposed by the occlusion loss. Figure 9 illustrates clearly the change in prediction behaviour induced by our occlusion map processing framework. The occlusion loss incentivises predictions to remain within the occlusion area. Consequently, a portion of the workspace W that should remain worthy of consideration for the model's prediction is penalized, and remains unexplored.

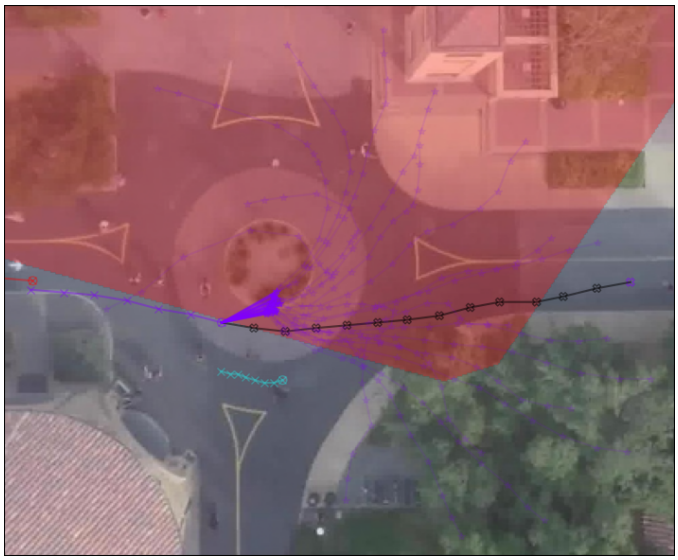


Fig. 9: A set of predictions made using model #VI. The application of the occlusion loss "pushes" the model away from W_V when predicting the past portion of the prediction (displayed as opaque colored lines).

Two different prediction strategies. Our implementation led to improvements over certain performance metrics, and degradation over others, posing difficulty to the definite assessment of our approach as being beneficial or detrimental to the prediction task. The main issue concerning our manner of applying the occlusion loss is that it essentially teaches two separate and different prediction behaviours, depending

on whether the points to be predicted are in the past, or in the future. Indeed, as we can see from Figure 9, the model initiates its prediction by a sudden turn into the occlusion zone, in order to comply with the occlusion loss. This initial "jump" ends up misdirecting the prediction away from the ground truth. It would seem that there is a misalignment between the objective of complying with the arrangement of W_O as it has been specified by the occlusion loss, and the objective of remaining close to the ground truth when generating prediction sequences. When considering the occlusion capability of prediction strategies, constraints can be applied to the prediction of past portions of occluded trajectories. However, the implementation of such constraints should be performed in such a way that it does not come at the expense of the consistency of the overall prediction. In that respect, it would seem that designs that encourage significantly different model behaviours between predictions made over the past and the future are undesirable, and should be avoided. With further research into the inclusion of information about the occluded space W_O within the prediction task, a prediction method that does not suffer from the negative performance effects we observed with OcclusionFormer might be attainable. Subsection VI-B covers some potential investigation avenues.

Our attempt at learning to predict with an occlusion map showed that, while it is possible for a model to relate map data with trajectory data, the incentive to maintain occluded trajectories within the bounds of the occluded region resulted in a degradation of the model's behaviour and performance. We acknowledge the limitations and inadequacies that our approach produced, and recommend for future research focusing on enabling occlusion capability within the prediction task, to remain cautious when designing systems accounting for the arrangement of the occluded space.

VI. Conclusion

The conclusion of our research will be first presented in Subsection VI-A. Recommendations for the continuation of work will then be provided in Subsection VI-B.

A. Toward Occlusion Capable Human Trajectory Prediction

This project investigated the effects of occlusions on the trajectory prediction task. In order to facilitate our research on the potential benefits of integrating spatial information of occlusions into the prediction framework, we created an occlusion simulator, which can be deployed to provide representations of occlusion scenarios on any widely available trajectory dataset. We reformulated the trajectory prediction task as the continuation of agents' past histories, starting from their *last observed state*. Our approach to addressing this prediction problem is based on the existing work of Yuan et al.'s AgentFormer model [23], a TransFormer based architecture, with promising characteristics for our specific definition of the prediction problem. We adapted the model into OcclusionFormer, a model that is specifically designed to operate under occluded circumstances, and investigated the effects of occlusions on its performance, as well as the possibility for the model to improve by receiving information about the spatial arrangement of the occluded space from an occlusion map.

Our evaluations of OcclusionFormer revealed that our model is fit to operate under occluded conditions, and that our adaptation of the original AgentFormer architecture incurs very minimal penalties on the performance of the model (when operating on fully observed trajectories). Transformer based prediction networks are very suitable when it comes to handling partially missing trajectory data. Deploying our model on occluded trajectories, we discovered that missing trajectory fragments increase the difficulty of the prediction task, resulting in an average degradation of MINADE/MINFDE scores by $+65\%$/$+45\%$, increasing linearly with respect to the predicted trajectory's last observed timestep. When investigating the option of endowing the model with the ability to use an occlusion map in order to leverage additional information about the space constraining occluded agents, we find that our model is capable of extracting information from the map. However, doing so does not result in a meaningful improvement of performance, as the model fails to harmoniously combine our occlusion loss with ground truth reconstruction losses. There are limitations to our analysis, as we observed that, when generating sets of occluded trajectories using our simulator, trajectory sets grouped by last observation timestep are not identically distributed. We also discovered that the majority of occlusion cases produced by the generator were susceptible to not contribute meaningfully to the prediction process, as they provided very loose constrains on the possible motions of occluded agents. Our findings nonetheless highlights that occlusions have a significant impact at the prediction stage of machine perception, and that this impact must not be neglected when developing the design of autonomous perception systems.

We believe that the trajectory prediction field would greatly benefit from dedicating more effort towards handling occlusions. We hope that the work we presented here provides a meaningful contribution to the problem of handling of occlusions at the prediction stage, and that it will encourage more people to pursue this exciting challenge.

B. Future Work

We recommend that future research into the exploration of *occlusion handling* within the trajectory prediction field focuses on two primary aspects.

Rendering Occlusions. The first recommended direction for future work is the development of fair and accurate representations of occlusions for a proper render of the problem of occlusions in the prediction problem. When it comes to this aspect, further improvements of our occlusion simulator could be sought. Notably, the improvement in the level of control over the degree of "informativity" that occlusions can provide for prediction. Indeed, currently, as our simulator seems to generate a great majority of occlusion cases which do not severely restrict the motions of occluded agents, it remains somewhat impractical to conduct highly detailed analyses

of the benefits of approaching the occlusion problem by accounting for the spatial configuration of occlusions. A larger pool of informative occlusions could allow us to extract further insights into how much performance can be gained from our approach. This could be achieved by altering the constraints on the creation of occlusion cases (e.g., by reducing the set of last non-occluded point candidates to a temporal window that terminates a few timesteps after $t = 0$, or by accounting for turns, stops and other changes in the motion of agents, in a way that guarantees "difficult" occlusion cases). Additionally, further research into the informativity of occlusions as they manifest themselves in reality would also be helpful, as such research could serve towards providing better design recommendations for the improvement of the simulator. Along with this potential research avenue, other efforts into closing the gap between simulation and reality could also be made, such as developing the simulator to account for multiple occluded areas within the workspace, and/or generating occlusion cases that are dynamically changing over time. Alongside this striving towards rendering as accurate a representation of the prediction problem as possible, future research might benefit from a more in depth investigation of the interface between prediction frameworks and upstream perception processes of detection and tracking, particularly in the description of the corruption that trajectories are subject to due to occlusions. Indeed, the current standard approach to trajectory prediction remains highly focused on perfectly observed, clean trajectory datasets. While they provide a simplified representation of the prediction problem (and thereby facilitate research without immediately pushing complexity to a paralysing extent), they implicitly hide all forms of incompatibilities that a perception system may have between the tracking stage and the prediction stage. The improvement of the interface between those stages is consequently under-researched, and will continue to be if no effort is dedicated to closing the gap between the idealized representation of trajectory data that is currently being used within the prediction field, and the imperfect trajectory data that is actually produced by tracking systems.

Improving Occlusion Capability. The second aspect which should attain a high priority in the research towards better occlusion capable trajectory prediction is in the development of models directly designed and intended for deployment under occluded conditions. The capacity of our model, OcclusionFormer, to operate under such conditions, makes it a promising starting point for future research. Continuing our research on the integration of an occlusion map into the prediction framework should focus on finding a more suitable specification and application of the occlusion loss, in order to prevent the inadequacies in prediction behaviour we were confronted with. Perhaps one of the ways this could be achieved is by applying the occlusion loss only in the second phase of training (which is responsible for the training of a diverse trajectory sampler), so as to only modify the creation of latent codes $\boldsymbol{z}$ used by the model to generate its multi-modal predictions. However, we recognise that investigations into the redefinition of the occlusion loss can only increase the complexity of our design. Alternatively, the integration of an occlusion map within the prediction task could be considered without the implementation of an additional occlusion loss. The occlusion map could instead be used only in a retrospective way, contributing to the assessment of the likelihood of prediction modes after they have been generated. Once the problem of the misalignment of the occlusion loss with respect to the main objective of capturing the ground truth has been properly addressed, a more extensive hyper-parameter search could be conducted. Aside from the integration of the occlusion map, other promising characteristics of OcclusionFormer could be worth investigating. One particularity of our model (and of its precursor AgentFormer) is the ability for trajectory elements to communicate social information across agents and across timesteps (by means of agent-aware attention), possibly letting trajectory elements of agents access information about their neighbours in their *relative future*. In a context where occluded agents must be accounted for, this particular way of letting social information flow across timesteps could perhaps be beneficial to the prediction task (for example, it could help refine the prediction of occluded agents who are following the movements of a group of neighbour agents). The insights that would be gained from investigations on this aspect of the propagation of social information could lead to recommendations for the design of socially aware prediction methods.

REFERENCES

[1] Alexandre Alahi et al. "Social LSTM: Human Trajectory Prediction in Crowded Spaces". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). ISSN: 1063-6919. June 2016, pp. 961–971. DOI: 10.1109/CVPR.2016.110.

[2] Joshua Andle et al. *The Stanford Drone Dataset is More Complex than We Think: An Analysis of Key Characteristics*. Mar. 22, 2022. DOI: 10.48550/arXiv.2203.11743. arXiv: 2203.11743[cs]. URL: http://arxiv.org/abs/2203.11743 (visited on 05/03/2023).

[3] Stefan Becker et al. "Handling Missing Observations with an RNN-based Prediction-Update Cycle". In: *Computer Analysis of Images and Patterns*. Ed. by Nicolas Tsapatsoulis et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2021, pp. 311–321. ISBN: 978-3-030-89128-2. DOI: 10.1007/978-3-030-89128-2_30.

[4] Dražen Brščić et al. "Person Tracking in Large Public Spaces Using 3-D Range Sensors". In: *IEEE Transactions on Human-Machine Systems* 43.6 (Nov. 2013). Conference Name: IEEE Transactions on Human-Machine Systems, pp. 522–534. ISSN: 2168-2305. DOI: 10.1109/THMS.2013.2283945.

[5] Ryo Fujii et al. *A Two-Block RNN-based Trajectory Prediction from Incomplete Trajectory*. Mar. 16, 2022. DOI: 10.48550/arXiv.2203.07098. arXiv: 2203.07098[cs]. URL: http://arxiv.org/abs/2203.07098 (visited on 12/07/2022).

[6] Francesco Giuliari et al. *Transformer Networks for Trajectory Forecasting*. Oct. 21, 2020. DOI: 10.48550/arXiv.2003.08111. arXiv: 2003.08111[cs]. URL: http://arxiv.org/abs/2003.08111 (visited on 12/07/2022).

[7] Agrim Gupta et al. *Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks*. Mar. 28, 2018. DOI: 10.48550/arXiv.1803.10892. arXiv: 1803.10892[cs]. URL: http://arxiv.org/abs/1803.10892 (visited on 02/17/2023).

[8] Vineet Kosaraju et al. *Social-BiGAT: Multimodal Trajectory Forecasting using Bicycle-GAN and Graph Attention Networks*. July 16, 2019. DOI: 10.48550/arXiv.1907.03395. arXiv: 1907.03395[cs]. URL: http://arxiv.org/abs/1907.03395 (visited on 02/17/2023).

[9] Namhoon Lee et al. *DESIRE: Distant Future Prediction in Dynamic Scenes with Interacting Agents*. Apr. 14, 2017. DOI: 10.48550/arXiv.1704.04394. arXiv: 1704.04394[cs]. URL: http://arxiv.org/abs/1704.04394 (visited on 02/17/2023).

[10] Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. "Crowds by Example". In: *Computer Graphics Forum* 26.3 (2007). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8659.2007.01089.x, pp. 655–664. ISSN: 1467-8659. DOI: 10.1111/j.1467-8659.2007.01089.x. URL: http://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2007.01089.x (visited on 02/16/2023).

[11] Kunming Li et al. "Attentional-GCNN: Adaptive Pedestrian Trajectory Prediction towards Generic Autonomous Vehicle Use Cases". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021 IEEE International Conference on Robotics and Automation (ICRA). ISSN: 2577-087X. May 2021, pp. 14241–14247. DOI: 10.1109/ICRA48506.2021.9561480.

[12] Ming Liang et al. *PnPNet: End-to-End Perception and Prediction with Tracking in the Loop*. June 27, 2020. DOI: 10.48550/arXiv.2005.14711. arXiv: 2005.14711[cs]. URL: http://arxiv.org/abs/2005.14711 (visited on 12/08/2022).

[13] Katie Luo et al. "Safety-Oriented Pedestrian Occupancy Forecasting". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). ISSN: 2153-0866. Sept. 2021, pp. 1015–1022. DOI: 10.1109/IROS51168.2021.9636691.

[14] Wenjie Luo, Bin Yang, and Raquel Urtasun. *Fast and Furious: Real Time End-to-End 3D Detection, Tracking and Motion Forecasting with a Single Convolutional Net*. Dec. 22, 2020. DOI: 10.48550/arXiv.2012.12395. arXiv: 2012.12395[cs]. URL: http://arxiv.org/abs/2012.12395 (visited on 12/08/2022).

[15] Barbara Majecka. "Statistical models of pedestrian behaviour in the Forum". In: 2009. URL: http://www.semanticscholar.org/paper/Statistical-models-of-pedestrian-behaviour-in-the-Majecka/6505d259758fc2fd4e60da018c35d687a2ddc250 (visited on 02/16/2023).

[16] Seong Hyeon Park et al. *Diverse and Admissible Trajectory Forecasting through Multimodal Context Understanding*. Aug. 31, 2020. DOI: 10.48550/arXiv.2003.03212. arXiv: 2003.03212[cs]. URL: http://arxiv.org/abs/2003.03212 (visited on 08/08/2023).

[17] S. Pellegrini et al. "You'll never walk alone: Modeling social behavior for multi-target tracking". In: *2009 IEEE 12th International Conference on Computer Vision*. 2009 IEEE 12th International Conference on Computer Vision. ISSN: 2380-7504. Sept. 2009, pp. 261–268. DOI: 10.1109/ICCV.2009.5459260.

[18] Alexandre Robicquet et al. "Learning Social Etiquette: Human Trajectory Understanding In Crowded Scenes". In: ed. by Bastian Leibe et al. Vol. 9912. Book Title: Computer Vision – ECCV 2016 Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 549–565. ISBN: 978-3-319-46483-1 978-3-319-46484-8. DOI: 10.1007/978-3-319-46484-8_33. URL: http://link.springer.com/10.1007/978-3-319-46484-8_33 (visited on 02/16/2023).

[19] Andrey Rudenko et al. "Human Motion Trajectory Prediction: A Survey". In: *The International Journal of Robotics Research* 39.8 (July 2020), pp. 895–935. ISSN: 0278-3649, 1741-3176. DOI: 10.1177/0278364920917446. arXiv: 1905.06113[cs]. URL: http://arxiv.org/abs/1905.06113 (visited on 12/08/2022).

[20] Christoph Schöller et al. *What the Constant Velocity Model Can Teach Us About Pedestrian Motion Prediction*. Jan. 22, 2020. DOI: 10.48550/arXiv.1903.07933. arXiv: 1903.07933[cs]. URL: http://arxiv.org/abs/1903.07933 (visited on 02/02/2023).

[21] Jianhua Sun et al. "Human Trajectory Prediction with Momentary Observation". In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE Computer Society, June 1, 2022, pp. 6457–6466. ISBN: 978-1-66546-946-3. DOI: 10.1109/CVPR52688.2022.00636. URL: https://www.computer.org/csdl/proceedings-article/cvpr/2022/694600g457/1H1n1HKcpvW (visited on 12/19/2022).

[22] Ashish Vaswani et al. *Attention Is All You Need*. Dec. 5, 2017. DOI: 10.48550/arXiv.1706.03762. arXiv: 1706.03762[cs]. URL: http://arxiv.org/abs/1706.03762 (visited on 01/05/2023).

[23] Ye Yuan et al. *AgentFormer: Agent-Aware Transformers for Socio-Temporal Multi-Agent Forecasting*. Oct. 7, 2021. DOI: 10.48550/arXiv.2103.14023. arXiv: 2103.14023[cs]. URL: http://arxiv.org/abs/2103.14023 (visited on 12/19/2022).

[24] Bolei Zhou, Xiaogang Wang, and Xiaoou Tang. “Understanding collective crowd behaviors: Learning a Mixture model of Dynamic pedestrian-Agents”. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. 2012 IEEE Conference on Computer Vision and Pattern Recognition. ISSN: 1063-6919. June 2012, pp. 2871–2878. DOI: 10.1109/CVPR.2012.6248013.

APPENDIX A
CODE MODIFICATION: UNLOCKING OPERATIONS ON PARTIALLY MISSING TRAJECTORIES

In their supplementary material, Yuan et al. [23] theorised a method for AgentFormer to operate on partially missing sequences. As the temporality of input elements is informed by positional encoding, and the social aspects are managed by the agent-aware attention mechanism, unobserved input elements could simply be omitted from the input sequence.

However, the implementation of their model does not allow for such processing of the input sequences. Although the model effectively reasons about temporality through positional encoding, it implicitly relies on the structure of the input data to assign elements their corresponding time code: input sequences are built from a list of agent sequences, which are stacked prior to being processed by the model. For example, an instance containing 3 agents will provide the model with the past/observed data as a list containing 3 sequences of 8 positions $[x, y]$. These sequences are then stacked into a tensor of shape $[3, 8, 2]$, then reshaped to $[24, 2]$, before being further processed by the model. This structuring of the data allows the internal components of the model to *implicitly infer temporality and agent identity* from the order of the input sequence. Consequently, the implementation effectively cannot process partially missing or fragmented trajectories, as such irregularities would disturb the order of the processed sequence data. As an example, the implementation of the function which generates the positional encodings for embedding of input sequences utilizes PyTorch's `repeat_interleave()` function, implicitly requiring a specific structure of the input data, which may not be satisfied when encountering partially missing trajectories:

```
def get_pos_enc(self, num_t, num_a, t_offset):
      pe = self.pe[t_offset: num_t + t_offset, :]
      pe = pe.repeat_interleave(num_a, dim=0)
      return pe
```

We remedy this problem by reworking the internal implementation of AgentFormer, reformulating its processes as operations making use of 3 separate sequences: a *features* sequence (e.g., the input states sequence $\{\boldsymbol{x}_a^t\}$), accompanied by a corresponding *identity* sequence $\{a\}$ and *timestep* sequence $\{t\}$. The identity and timestep sequences are used to inform the behaviour of operations on the features sequence. For example, our reworked implementation generates positional encodings by referring to the timestep sequence in order to allow for irregular time code sequences:

```
    # x: features sequence                                  [B, T, model_dim]
    # time_tensor: corresponding timestep sequence          [B, T]
    pos_enc = self.pe[time_tensor + self.t_index_shift] # [B, T, model_dim]
```

APPENDIX B
ADDITIONAL IMPLEMENTATION DETAILS

Attention Layers: we follow the same configuration as the original implementation of AgentFormer: layers have 8 attention heads, each one with dimension 32 (i.e., a layer dimensionality of 256). The inner layer of the fully-connected feed-forward network within the attention layer is 512. The dropout ratio is 0.1.

Occlusion Map CNN: the CNN processing our occlusion map is made up of alternating 2D-convolutional layers and 2D-maxpool layers. The input feature set of the occlusion map is of shape (1, 800, 800). The final layer of the CNN maps the convolved feature set into $\boldsymbol{h}_M$, with dimension 256.

TABLE VI: The configuration of layers for our occlusion map CNN.

layers	**output feature dimension (C, H, W)**
`Conv2D(out_channels=4, kernel_size=11, stride=4)`	156816 (4, 198, 198)
`MaxPool2D(kernel_size=2, stride=2)`	39204 (4, 99, 99)
`Conv2D(out_channels=8, kernel_size=9, stride=3)`	7688 (8, 31, 31)
`MaxPool2D(kernel_size=2, stride=2)`	1800 (8, 15, 15)
`Conv2D(out_channels=8, kernel_size=3, stride=1)`	1352 (8, 13, 13)
`MaxPool2D(kernel_size=2, stride=2)`	288 (8, 6, 6)

Encoder and Decoder modules: The Context Encoder, the Prediction Sequence Encoder and the Prediction Sequence Decoder all have 2 Attention Layers. The Multi-Layer Perceptrons present in the Prediction Sequence Encoder and Decoder are identical, containing each 2 layers, with dimensions (512, 256).

Temporal Encoding: The temporal encoding of trajectory features is done by concatenating those features with positional encodings generated from the timestep sequence $\{t\}$. Those concatenated features are then passed through a fully connected layer, to preserve the features' original dimensionality. This is an unusual way of performing encoding: usually, it is done by simply adding features and positional codes together. Our decision to follow this temporal encoding framework is motivated by the desire to remain as close as possible to the original implementation of AgentFormer.

Prior and Trajectory Sampler modules: The prior module and trajectory sampler module produce the latent codes $\boldsymbol{z}$, which are used as part of the input for our prediction sequence decoder. The chosen dimensionality of those latent codes is $d_{\boldsymbol{z}} = 32$. The number of multi-modal predictions generated for each agent is $K = 20$.

TABLE VII: The configuration of loss components for our different models. Each column provides loss component weight values for the indicated model versions.

			Agent-Former #I	**Occlusion-Former** #II, #III, #IV	**O.Former w/ map A** #V	**O.Former w/ map B** #VI
$\mathcal{L}_{\text{phase I}}$	$\mathcal{L}_{\text{MSE}}$	α_{MSE}	1	12	12	12
		γ_1	N/A	N/A	1	0.5
		γ_2	N/A	N/A	1	1
	$\mathcal{L}_{\text{sample}}$	α_{sample}	1	12	12	12
		γ_1	N/A	N/A	1	0.5
		γ_2	N/A	N/A	1	1
	$\mathcal{L}_{\text{occl., MSE}}$	$\alpha_{\text{occl., MSE}}$	N/A	N/A	12	6
	$\mathcal{L}_{\text{occl., sample}}$	$\alpha_{\text{occl., sample}}$	N/A	N/A	12	6
	$\mathcal{L}_{\text{KL}}$	α_{KL}	1	1	1	1
$\mathcal{L}_{\text{phase II}}$	$\mathcal{L}_{\text{sample}}$	α_{sample}	5	60	60	60
		γ_1	N/A	N/A	1	0.5
		γ_2	N/A	N/A	1	1
	$\mathcal{L}_{\text{occl., sample}}$	$\alpha_{\text{occl., sample}}$	N/A	N/A	60	30
	$\mathcal{L}_{\text{KL}}$	α_{KL}	0.1	0.1	0.1	0.1
	$\mathcal{L}_{\text{diverse}}$	α_{diverse}	20	20	20	20

Loss configuration: AgentFormer (and consequently OcclusionFormer) are both trained in two separate phases, each with a different loss configuration. The first phase is aimed towards training the prediction model itself (i.e., the context encoder, prediction sequence encoder and decoder, as well as the prior modules). The second phase trains a trajectory sampler module, which is used as a substitute to a pretrained model's prior module. Loss functions for phase 1 and 2 are defined according to Equation 6 and Equation 7, respectively. The loss weights for our different models were configured according to Table VII. We clip the maximum value of $\mathcal{L}_{\text{KL}}$ to 2 in the first phase of training, 10 in the second phase, and set the value of the scaling factor σ_d (used in $\mathcal{L}_{\text{diverse}}$ in the second phase) to 10, following the original training policy of AgentFormer.

We would like to clarify one important difference in the configuration of loss components between AgentFormer and OcclusionFormer: as we can see from Table VII, the values of α_{MSE} and α_{sample} for AgentFormer and OcclusionFormer

are 1 and 12, respectively. This is because those loss components are normalized by the number of agents being predicted, in the original AgentFormer implementation. As our method deals in varying prediction sequence lengths per agent, we instead normalize the loss components by the total length of the prediction sequence (which is equivalent to $T_{\text{pred}} \cdot N$ when not a single agent is being occluded). The changes in the weight values reflect this adaptation in the normalization of the losses.

Training Regime: For both phase I and phase II, we train using the Adam optimizer, with a starting learning rate of 10^{-4}. In the first phase, we halve the learning rate every 50000 batches. In the second phase, we halve the learning rate every 12500 batches.

APPENDIX C
STANFORD DRONE DATASET: COORDINATES CONVERSION

The annotations of the trajectories within the Stanford Drone Dataset are expressed in pixel coordinates. Unfortunately, the creators of the dataset [18] do not provide tools to convert trajectories into metric coordinates.

We express the trajectories in our work in meters, making use of the coordinate conversions available in Table VIII. The coordinate conversions were manually obtained, by measuring distances between geographic landmarks in each scene to establish estimates of px/m ratios. Metric distances were obtained using Google Maps.

TABLE VIII: Pixel to meter coordinate conversions for each video in the Stanford Drone Dataset.

scene	video	px	m	px/m	scene	video	px	m	px/m
bookstore	video0	663.40	24.85	26.70	hyang	video5	926.90	28.17	32.90
bookstore	video1	624.20	24.85	25.12	hyang	video6	515.10	28.17	18.29
bookstore	video2	569.20	23.17	24.57	hyang	video7	796.10	28.17	28.26
bookstore	video3	601.20	24.59	24.45	hyang	video8	853.00	29.77	28.65
bookstore	video4	778.00	27.66	28.13	hyang	video9	914.00	31.67	28.86
bookstore	video5	514.20	15.70	32.75	hyang	video10	1069.40	58.38	18.32
bookstore	video6	690.40	21.48	32.14	hyang	video11	668.10	37.16	17.98
coupa	video0	1318.60	41.11	32.07	hyang	video12	498.10	28.97	17.19
coupa	video1	1553.20	41.96	37.02	hyang	video13	372.20	20.20	18.43
coupa	video2	1013.00	27.17	37.28	hyang	video14	648.80	27.99	23.18
coupa	video3	966.10	26.95	35.85	little	video0	681.20	19.94	34.16
deathCircle	video0	421.90	17.54	24.05	little	video1	812.00	23.04	35.24
deathCircle	video1	435.20	17.54	24.81	little	video2	521.30	15.53	33.57
deathCircle	video2	426.30	17.54	24.30	little	video3	684.40	19.94	34.32
deathCircle	video3	601.80	17.54	34.31	nexus	video0	660.00	29.58	22.31
deathCircle	video4	442.70	17.54	25.24	nexus	video1	936.40	41.85	22.38
gates	video0	1033.80	42.07	24.57	nexus	video2	893.00	38.43	23.24
gates	video1	936.80	36.32	25.79	nexus	video3	573.60	26.96	21.28
gates	video2	690.00	25.26	27.32	nexus	video4	602.80	28.00	21.53
gates	video3	952.80	33.38	28.54	nexus	video5	930.20	43.53	21.37
gates	video4	752.70	33.38	22.55	nexus	video6	779.10	29.58	26.34
gates	video5	952.20	33.38	28.53	nexus	video7	1365.90	51.52	26.51
gates	video6	685.10	22.89	29.93	nexus	video8	723.30	38.43	18.82
gates	video7	1537.30	65.90	23.33	nexus	video9	858.80	38.43	22.35
gates	video8	916.20	42.18	21.72	nexus	video10	603.20	26.96	22.37
hyang	video0	812.30	28.17	28.84	nexus	video11	1087.80	48.73	22.32
hyang	video1	618.00	28.17	21.94	quad	video0	1591.40	68.58	23.21
hyang	video2	1038.70	57.69	18.00	quad	video1	1609.40	68.91	23.36
hyang	video3	712.60	43.38	16.43	quad	video2	1581.50	68.60	23.05
hyang	video4	811.50	28.17	28.81	quad	video3	1560.50	68.64	22.73

APPENDIX D
THE OCCLUSION SIMULATOR

A. Occlusion Simulator Design

As our research intends to study the capacity for a prediction model to reason about the spatial configuration of occlusions, we must be able to provide spatio-temporal representations of occlusions.

The underlying principle behind the occlusion simulation process is to separate the workspace into W_V and W_O by placing two objects in the scene: a "virtual ego" point $\boldsymbol{p}_{\text{ego}}$, and a "virtual occluder" segment $W_{\text{occl}} \equiv \left(\boldsymbol{p}_{\text{occl, 1}}, \boldsymbol{p}_{\text{occl, 2}}\right)$. Visibility of the ego stops at the boundary of the scene, and is also limited by the virtual occluder. The visible region W_V is then defined as the visibility polygon for $\boldsymbol{p}_{\text{ego}}$, with obstacles being the scene boundary, and the occluder W_{occl}. W_O can be obtained by subtracting W_V from the entire scene.

In order to ensure that the generated occlusion regions correctly limit the observation of agents, the placement of $\boldsymbol{p}_{\text{ego}}$ and W_{occl} is partially informed by agents' trajectories. Particularly, we focus the simulation process towards generating *currently* occurring occlusions (i.e., , occlusions where the most recent portion of an agent's past trajectory is entirely unobserved). Although neither the ego, nor the occluder are physical objects with which agents can interact and account for in their motion, we deem appropriate to reduce the potential interference between them, in order to prevent unrealistic occlusion arrangements. Consequently, the ego and occluder should not lie too close to and/or intersect with agent trajectories. Finally, it is also important for the simulation to not be deterministically defined by agent trajectories, as a prediction model could potentially implicitly learn deterministic rules defining occlusions, and infer some information about positions of occluded agents. To this end, randomness plays an important role in the simulation process. An example of the step-by-step occlusion simulation process is shown in Figure 10.

B. Occlusion Simulator: Limitations and Potential Improvements

The presented occlusion simulator can sometimes fail to generate occlusions in certain cases. For example, this can happen when all agents present in the scene are standing still, as their trajectories cannot be separated into an occluded and a visible part based on their disposition within the workspace. It can also happen due to an impossibility to satisfy multiple geometric constraints imposed on the placement of $\boldsymbol{p}_{\text{ego}}$ and W_{occl}. Reducing the failure rate of the occlusion simulator can be done by imposing more lenient geometric constraints: for example, by reducing the minimum distance factors d_R and d_O by which trajectories are "inflated" in order to produce limiting regions. The combination of geometric constraints imposed on the placement of the virtual ego could be considered as too restrictive. For example, the wedge shaped regions which are added to the restriction area for the placement of the ego, R_{ego} (in Figure 10b), or the restriction enforcing that no agent lies between the virtual ego and the target agent (i.e. the blue region, B_{ego}, also in Figure 10b), might perhaps be altered for a more lenient placement of the ego without affecting the realism of the occlusions being generated.

The definition of $\{W_V, W_O\}$ does not inherently have to be done by means of placing a virtual ego and occluder. The process we devise for the coherent placement of these objects within the workspace necessitates numerous computations involving visibility polygon calculations and polygon triangulations. More simple approaches could be investigated. For example, it could be possible to instead model occluder objects as unique points or as predefined polygons and to use such definitions to divide the space into W_V and W_O, accounting for a desired occlusion window over the target agent.

Due to the focus of each run of the simulation onto one single agent, the occlusions generated make up a small percentage of the entirety of the available trajectory data (as we can see from the number of occlusion cases present in our test split, which amounts to 11.6% of the trajectories present). Generating spatio-temporal representations of currently occurring occlusion cases makes the percentage of occlusion cases over the entire dataset a difficult property of the resulting dataset to control. Generating multiple virtual occluders W_{occl} per run would further complexify the operations of the simulator, especially in the case where potential interferences between multiple occluders must be accounted for and prevented.

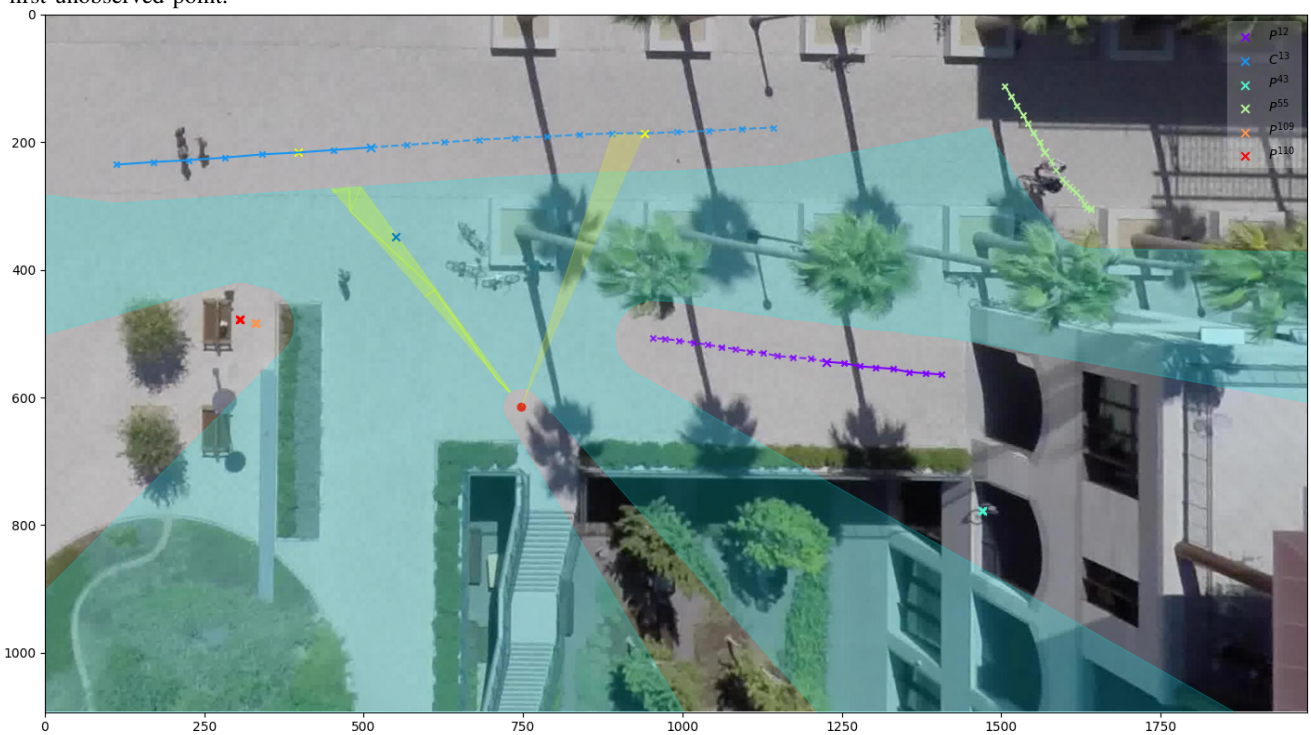
Another important limitation of the occlusion process lies in the inherent characteristic of being a simulation: as such, it is difficult to know whether the level of detail to which occlusions are rendered is sufficiently realistic for prediction models to be able to generalise to trajectory cases extracted in the real world when they are deployed. The simulator itself generates rather simplistic occlusion cases: always only one occluder is present in the scene, generating only one single occluded region (i.e., $\boldsymbol{M}_O$ is always equivalent to a single concave polygon). Additionally, the configuration of the occlusion space itself remains static over the entire time period $[-T_{\text{obs}} + 1, T_{\text{pred}}]$. More advanced forms of rendering the occlusion space could perhaps generate dynamic occlusions, which change over time. Another important remark to make with this simulation scheme is that, while it generates mutually coherent occlusion regions and occlusion patterns among the trajectories of agents, this may very well not be the case in a real life scenario, where the inference of the occluded region $\boldsymbol{M}_O$ and the tracking of agents' trajectories relies on upstream perception functions, which might generate "disagreeing" data. Further investigations to narrow the gap between simulation data and "real" data collected from an autonomous system deployed in the world could be worthwhile.

(a) The red regions, R_{ego}, restrict the placement of the virtual ego $\boldsymbol{p}_{\text{ego}}$ within the workspace. They correspond to the regions of points within a distance d_R from agent's complete trajectories over $t \in \{-T_{\text{obs}}; T_{\text{pred}}\}$, or a distance d_B from the boundaries of the scene. Similarly, orange regions circumvent agents' trajectories, with delimiting distance factor d_O, in order to limit the positioning of the points defining W_{occl}. We set $d_R > d_O$, in order to ensure that some free space exists between ego and trajectories, facilitating the placement of W_{occl}.

(b) An agent is randomly selected as the "target" for occlusion simulation. Eligible "target" candidates must be moving, the probability of an agent being selected as target is proportional to their distance travelled over $t \in \{-T_{\text{obs}}; T_{\text{pred}}\}$. The blue region, B_{ego}, is defined as the intersection of the visibility polygons for the first and last trajectory points of the target agent, with the past trajectories of other agents acting as visibility constraints, to prevent cases where an agent might lie between the target and $\boldsymbol{p}_{\text{ego}}$. Two yellow points in the target's trajectory are randomly sampled as the first and last non-occluded points (one in $[-T_{\text{obs}}; 0]$, one in $[1; T_{\text{pred}}]$). Mutually opposite wedge shaped red regions are defined from those first and last non-occluded points, and are added to R_{ego}, in order to minimize risk of aligning the ego directly with the trajectory of the target.

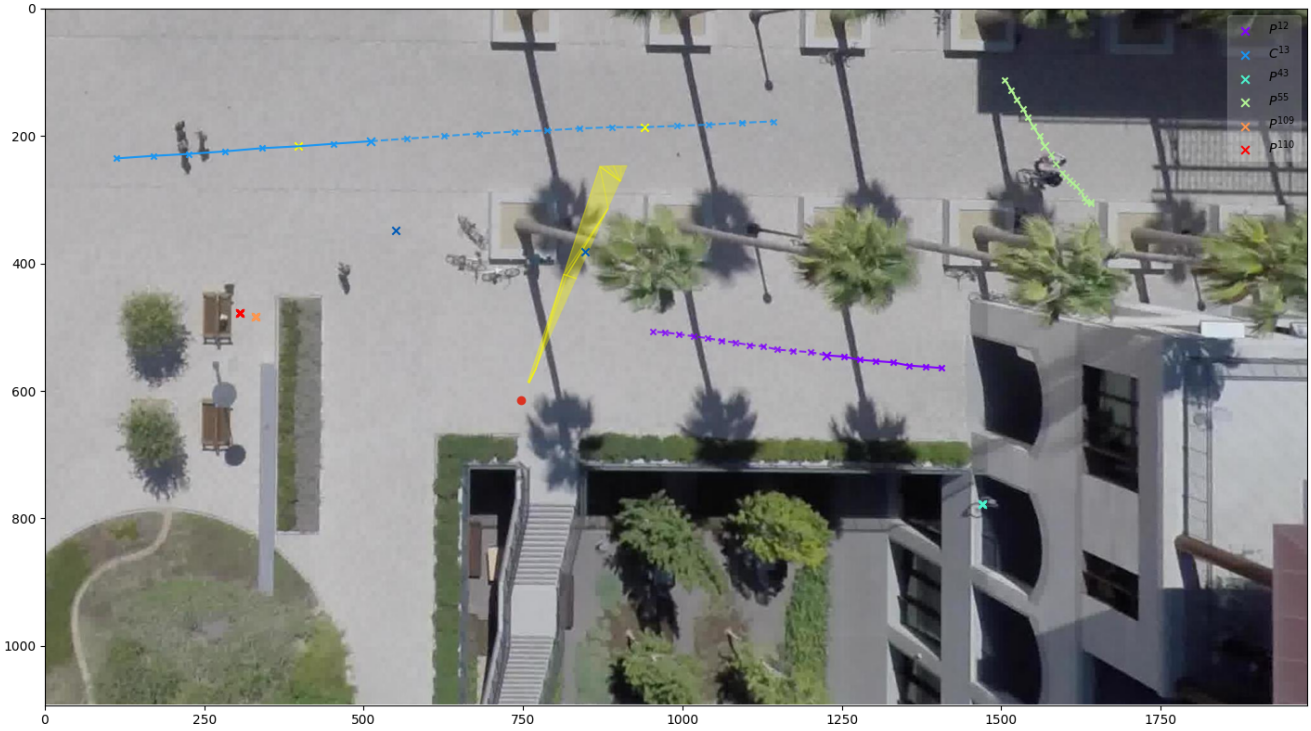
Fig. 10: An illustration of the occlusion simulation process.

(c) The candidate region for placement of $\boldsymbol{p}_{\text{ego}}$ (in green) is defined as $B_{\text{ego}} - R_{\cancel{\text{ego}}}$. A random point is sampled from this region. This point will be the virtual ego, $\boldsymbol{p}_{\text{ego}}$ (the red circle). A small region around $\boldsymbol{p}_{\text{ego}}$ is added to $O_{\cancel{\text{occl}}}$, ensuring a minimal distance $d_{\text{ego, occl}}$ between the occluder and the ego. The triangular candidate region for placement of $\boldsymbol{p}_{\text{occl, 1}}$ is obtained using $\boldsymbol{p}_{\text{ego}}$, the last observed point (in yellow), and the first unobserved point.

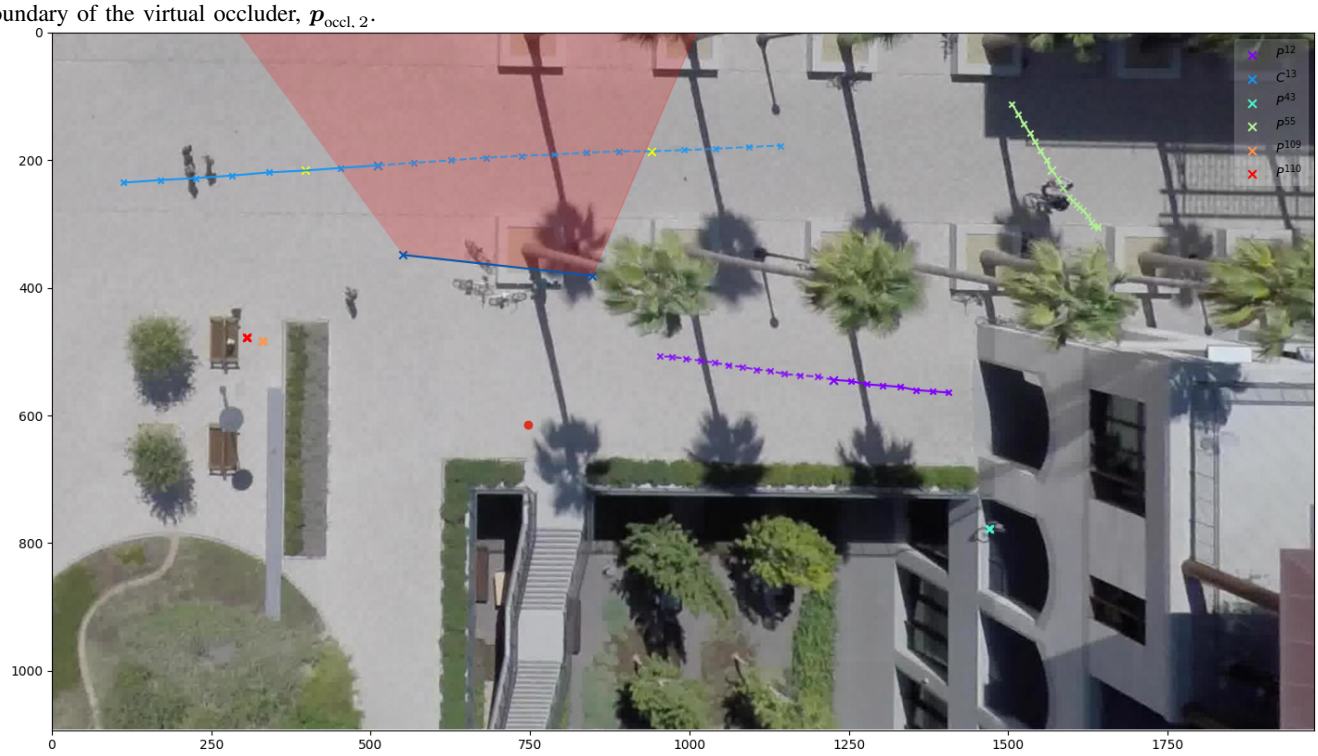


(d) $O_{\cancel{\text{occl}}}$ is subtracted from the candidate yellow region. A point is sampled from this resulting region. This point will be the first boundary of the virtual occluder, $\boldsymbol{p}_{\text{occl, 1}}$. The triangular candidate region for placement of $\boldsymbol{p}_{\text{occl, 2}}$ is obtained from $\boldsymbol{p}_{\text{ego}}$, the first reobserved point (in yellow), and the last unobserved point. The visibility polygon of $\boldsymbol{p}_{\text{occl, 1}}$ with the previously defined orange regions is computed (in blue).

Fig. 10: An illustration of the occlusion simulation process (cont.).



(e) The intersection of the second triangular yellow region with the blue region is computed, so as to guarantee that the placement of $\boldsymbol{p}_{\text{occl, 2}}$ will result in a virtual occluder that does not intersect with any agent's trajectory. A point is sampled from this region. It will be the second boundary of the virtual occluder, $\boldsymbol{p}_{\text{occl, 2}}$.



(f) W_{occl} is defined as the segment bounded by $\boldsymbol{p}_{\text{occl, 1}}$ and $\boldsymbol{p}_{\text{occl, 2}}$ (in blue). The visibility polygon of $\boldsymbol{p}_{\text{ego}}$ is computed, with the boundary of the scene and W_{occl} as obstacles. This visibility polygon is equivalent to W_V. The complementary of this region (in red), which is equivalent to W_O, occludes the trajectory of the target agent between their first and last observable trajectory points (in yellow).

Fig. 10: An illustration of the occlusion simulation process (cont.).

APPENDIX E
PERFORMANCE SUMMARY TABLES

We provide the following performance summary tables, aggregating average scores over different subsets of our dataset.

TABLE IX: Performance summaries for models evaluated exclusively on the subset of fully observed agents of the test set (score averages aggregated over a set of 91220 trajectories). ADE/FDE scores are expressed in m.

	#	Model	Train Data	Test Data	MIN ADE	MIN FDE	MEAN ADE	MEAN FDE
a.	I	AgentFormer	FO	FO	0.4012	0.7073	2.2193	4.8175
b.	I	AgentFormer	FO		0.3936	0.6934	2.1470	4.6852
c.	I	AgentFormer	FO		0.3958	0.7049	2.1149	4.6158
d.	I	AgentFormer	FO		0.3983	0.7016	2.1599	4.6517
e.	I	AgentFormer	FO		0.3996	0.7068	2.1864	4.7002
f.	*I*	*Avg. AgentFormer*	*FO*		*0.3977*	*0.7028*	*2.1655*	*4.6941*
g.	II	OcclusionFormer	FO		0.4085	0.7285	2.2030	4.7196
h.	II	OcclusionFormer	FO		0.4009	0.7127	2.2050	4.7574
i.	II	OcclusionFormer	FO		0.4033	0.7130	2.1786	4.6817
j.	II	OcclusionFormer	FO		0.3950	0.7011	2.2366	4.7932
k.	II	OcclusionFormer	FO		0.4065	0.7284	2.2482	4.8775
l.	*II*	*Avg. OcclusionFormer*	*FO*		*0.4028*	*0.7168*	*2.2143*	*4.7659*
m.	II	OcclusionFormer	FO	OS	0.4227	0.7325	2.2271	4.7287
n.	II	OcclusionFormer	FO		0.4106	0.7140	2.2125	4.7429
o.	II	OcclusionFormer	FO		0.4161	0.7142	2.1920	4.6763
p.	II	OcclusionFormer	FO		0.4071	0.7024	2.2533	4.7902
q.	II	OcclusionFormer	FO		0.4182	0.7314	2.2751	4.8914
r.	*II*	*Avg. OcclusionFormer*	*FO*		*0.4150*	*0.7189*	*2.2320*	*4.7659*
s.	III	OcclusionFormer	OS		0.4345	0.7761	2.4211	5.1676
t.	IV	OcclusionFormer	DS		0.4256	0.7416	2.1797	4.6190
u.	V	O.Former w/ map A	DS		0.5175	0.8830	2.7205	5.7172
v.	VI	O.Former w/ map B	DS		0.4495	0.7975	2.2082	4.7138
w.	II	OcclusionFormer	FO	OS, I	0.4079	0.7273	2.1930	4.7013
x.	II	OcclusionFormer	FO		0.4007	0.7115	2.1941	4.7360
y.	II	OcclusionFormer	FO		0.4031	0.7119	2.1689	4.6644
z.	II	OcclusionFormer	FO		0.3947	0.6998	2.2280	4.7773
aa.	II	OcclusionFormer	FO		0.4058	0.7266	2.2389	4.8600
ab.	*II*	*Avg. OcclusionFormer*	*FO*		*0.4024*	*0.7154*	*2.2046*	*4.7478*

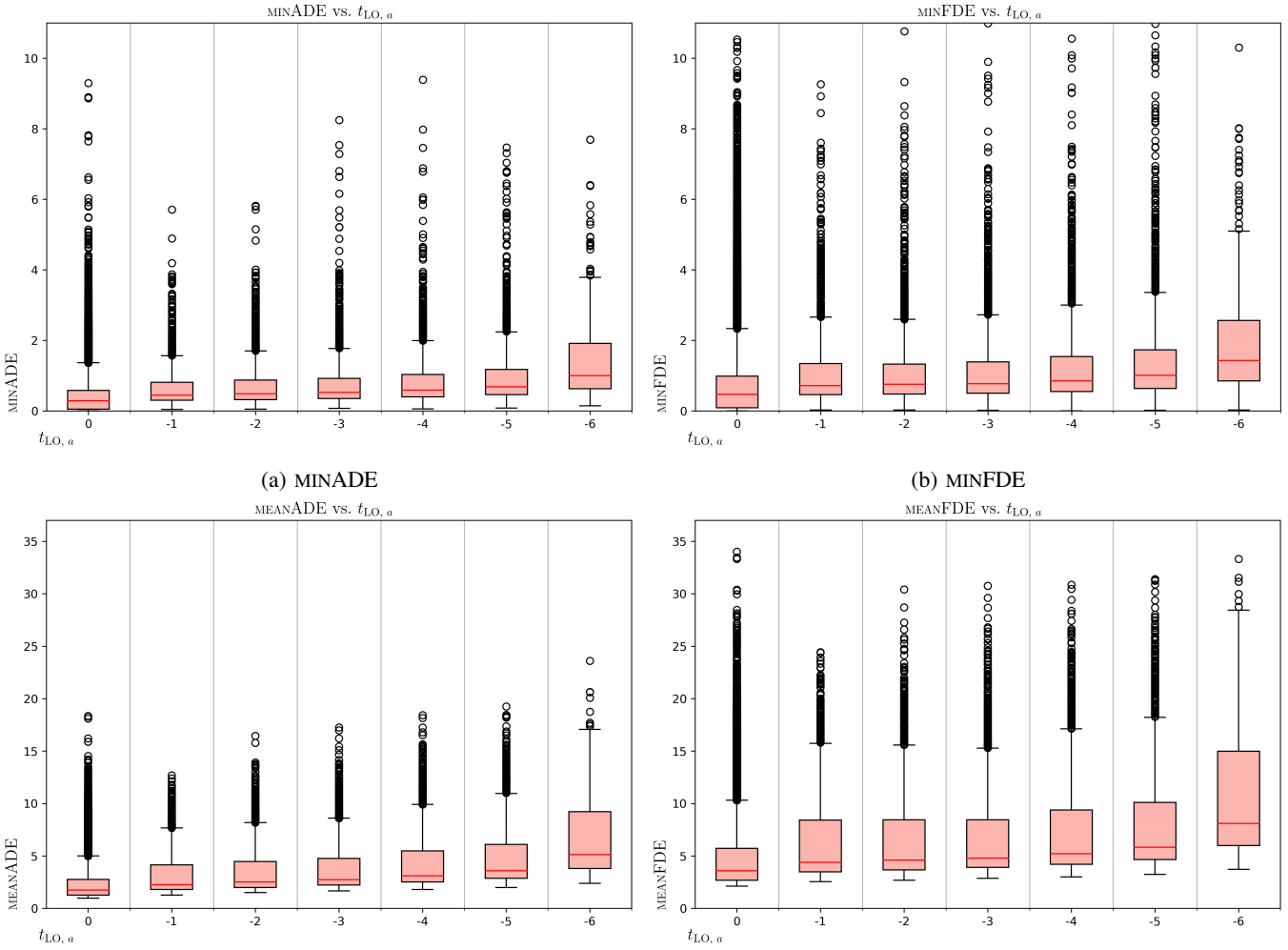
TABLE X: Performance summaries for models evaluated exclusively on the subset of occluded agents of the test set (score averages aggregated over a set of 11971 trajectories). ADE/FDE scores are expressed in m.

	#	Model	Train Data	Test Data	MIN ADE	MIN FDE	MEAN ADE	MEAN FDE	MIN_{past} ADE	MIN_{past} FDE	$\text{MEAN}_{\text{past}}$ ADE	$\text{MEAN}_{\text{past}}$ FDE	OAO	$\text{OAC}_{t=0}$
a.	I	AgentFormer	FO	FO	0.5005	0.8773	2.4392	5.2881	N/A	N/A	N/A	N/A	N/A	N/A
b.	I	AgentFormer	FO		0.4905	0.8518	2.3556	5.1132	N/A	N/A	N/A	N/A	N/A	N/A
c.	I	AgentFormer	FO		0.4965	0.8780	2.3497	5.1211	N/A	N/A	N/A	N/A	N/A	N/A
d.	I	AgentFormer	FO		0.4931	0.8593	2.3690	5.1028	N/A	N/A	N/A	N/A	N/A	N/A
e.	I	AgentFormer	FO		0.4942	0.8717	2.4048	5.1764	N/A	N/A	N/A	N/A	N/A	N/A
f.	*I*	*Avg. AgentFormer*	*FO*		*0.4950*	*0.8676*	*2.3837*	*5.1603*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*
g.	II	OcclusionFormer	FO		0.5075	0.8938	2.4728	5.3130	N/A	N/A	N/A	N/A	N/A	N/A
h.	II	OcclusionFormer	FO		0.4993	0.8769	2.4318	5.2525	N/A	N/A	N/A	N/A	N/A	N/A
i.	II	OcclusionFormer	FO		0.5032	0.8862	2.4276	5.2220	N/A	N/A	N/A	N/A	N/A	N/A
j.	II	OcclusionFormer	FO		0.4945	0.8661	2.4563	5.2746	N/A	N/A	N/A	N/A	N/A	N/A
k.	II	OcclusionFormer	FO		0.5082	0.9030	2.5347	5.4964	N/A	N/A	N/A	N/A	N/A	N/A
l.	*II*	*Avg. OcclusionFormer*	*FO*		*0.5025*	*0.8852*	*2.4647*	*5.3117*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*
m.	II	OcclusionFormer	FO	OS	1.4921	1.5451	4.6715	8.0710	0.7832	0.9392	1.4100	1.5388	0.7974	0.8216
n.	II	OcclusionFormer	FO		1.4616	1.4054	4.1984	7.1918	0.7990	1.0551	1.2384	1.5032	0.8701	0.8858
o.	II	OcclusionFormer	FO		1.4944	1.5164	4.4310	7.4925	0.7987	1.0700	1.3362	1.5299	0.8422	0.8591
p.	II	OcclusionFormer	FO		1.3902	1.4136	4.3769	7.5507	0.8206	0.9411	1.4628	1.4807	0.8379	0.8732
q.	II	OcclusionFormer	FO		1.5735	1.6359	4.7560	8.0479	0.7813	0.9579	1.5339	1.6080	0.7835	0.8069
r.	*II*	*Avg. OcclusionFormer*	*FO*		*1.4824*	*1.5033*	*4.4868*	*7.6708*	*0.7966*	*0.9926*	*1.3963*	*1.5321*	*0.8262*	*0.8493*
s.	III	OcclusionFormer	OS		0.8299	1.2851	4.0959	7.3148	0.1473	0.1828	0.4971	0.8267	0.9002	0.9243
t.	IV	OcclusionFormer	DS		0.9432	1.4342	4.5396	7.7556	0.2097	0.2405	0.7207	1.1428	0.8738	0.8791
u.	V	O.Former w/ map A	DS		1.0300	1.5287	4.8526	8.3020	0.3558	0.3680	0.8332	1.2476	0.9095	0.9042
v.	VI	O.Former w/ map B	DS		0.9850	1.4990	4.0173	7.0485	0.4146	0.4712	0.6422	0.9111	0.9787	0.9813
w.	II	OcclusionFormer	FO	OS, I	0.8728	1.2000	2.8609	5.6261	0.2974	0.4759	0.2974	0.4759	0.9504	0.9662
x.	II	OcclusionFormer	FO		0.8651	1.1839	2.8421	5.5949	0.2974	0.4759	0.2974	0.4759	0.9504	0.9662
y.	II	OcclusionFormer	FO		0.8642	1.1731	2.8345	5.5562	0.2974	0.4759	0.2974	0.4759	0.9504	0.9662
z.	II	OcclusionFormer	FO		0.8527	1.1537	2.8818	5.6575	0.2974	0.4759	0.2974	0.4759	0.9504	0.9662
aa.	II	OcclusionFormer	FO		0.8744	1.2011	2.9329	5.8146	0.2974	0.4759	0.2974	0.4759	0.9504	0.9662
ab.	*II*	*Avg. OcclusionFormer*	*FO*		*0.8658*	*1.1824*	*2.8704*	*5.6498*	*0.2974*	*0.4759*	*0.2974*	*0.4759*	*0.9504*	*0.9662*

TABLE XI: Performance summaries for models evaluated exclusively on the subset of agents undergoing a *difficult occlusion event* within the test set (score averages aggregated over a set of 353 trajectories). ADE/FDE scores are expressed in m.

	#	Model	Train Data	Test Data	MIN ADE	MIN FDE	MEAN ADE	MEAN FDE	MIN_{past} ADE	MIN_{past} FDE	$\text{MEAN}_{\text{past}}$ ADE	$\text{MEAN}_{\text{past}}$ FDE	OAO	$\text{OAC}_{t=0}$
a.	I	AgentFormer	FO	FO	0.6010	1.0635	2.8917	6.2164	N/A	N/A	N/A	N/A	N/A	N/A
b.	I	AgentFormer	FO		0.6037	1.0361	2.8138	6.0499	N/A	N/A	N/A	N/A	N/A	N/A
c.	I	AgentFormer	FO		0.6010	1.0590	2.7848	5.9878	N/A	N/A	N/A	N/A	N/A	N/A
d.	I	AgentFormer	FO		0.5791	0.9982	2.8260	6.0209	N/A	N/A	N/A	N/A	N/A	N/A
e.	I	AgentFormer	FO		0.5785	1.0093	2.8619	6.1001	N/A	N/A	N/A	N/A	N/A	N/A
f.	*I*	*Avg. AgentFormer*	*FO*		*0.5926*	*1.0332*	*2.8356*	*6.0750*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*
g.	II	OcclusionFormer	FO		0.6030	1.0589	2.9315	6.2132	N/A	N/A	N/A	N/A	N/A	N/A
h.	II	OcclusionFormer	FO		0.5802	0.9925	2.8659	6.1130	N/A	N/A	N/A	N/A	N/A	N/A
i.	II	OcclusionFormer	FO		0.6090	1.0811	2.8842	6.1442	N/A	N/A	N/A	N/A	N/A	N/A
j.	II	OcclusionFormer	FO		0.5867	1.0297	2.8831	6.1128	N/A	N/A	N/A	N/A	N/A	N/A
k.	II	OcclusionFormer	FO		0.6049	1.0386	2.9299	6.2930	N/A	N/A	N/A	N/A	N/A	N/A
l.	*II*	*Avg. OcclusionFormer*	*FO*		*0.5968*	*1.0401*	*2.8989*	*6.1752*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*
m.	II	OcclusionFormer	FO	OS	1.6273	1.9714	5.1820	9.1094	0.7084	0.8298	1.5173	1.4706	0.4178	0.4790
n.	II	OcclusionFormer	FO		1.6200	1.8756	4.8168	8.4635	0.7105	0.9052	1.3364	1.3937	0.4746	0.5368
o.	II	OcclusionFormer	FO		1.6754	1.9238	4.9395	8.5633	0.7138	0.9426	1.4332	1.4462	0.4406	0.4933
p.	II	OcclusionFormer	FO		1.5789	1.8312	5.0514	8.8182	0.7317	0.8273	1.6173	1.4143	0.4216	0.5072
q.	II	OcclusionFormer	FO		1.7693	2.0952	5.2547	9.0442	0.7195	0.8468	1.6796	1.5584	0.3859	0.4353
r.	*II*	*Avg. OcclusionFormer*	*FO*		*1.6542*	*1.9394*	*5.0489*	*8.7998*	*0.7167*	*0.8704*	*1.5168*	*1.4567*	*0.4281*	*0.4903*
s.	III	OcclusionFormer	OS		1.1390	1.6833	5.0621	8.9096	0.2634	0.3382	0.6722	1.0749	0.4241	0.3313
t.	IV	OcclusionFormer	DS		1.1715	1.6788	5.0205	8.6122	0.2332	0.2981	0.7837	1.2188	0.5274	0.4691
u.	V	O.Former w/ map A	DS		1.2900	1.9070	5.2874	9.1627	0.3618	0.4355	0.8959	1.2920	0.7231	0.7030
v.	VI	O.Former w/ map B	DS		1.1937	1.7937	4.4753	7.8381	0.4112	0.5525	0.6591	0.9723	0.8221	0.8115
w.	II	OcclusionFormer	FO	OS, I	1.6689	1.9397	4.2979	7.6262	0.6337	1.0504	0.6337	1.0504	0.2527	0.0000
x.	II	OcclusionFormer	FO		1.6889	2.0385	4.2924	7.6352	0.6337	1.0504	0.6337	1.0504	0.2527	0.0000
y.	II	OcclusionFormer	FO		1.6497	1.9489	4.2875	7.6236	0.6337	1.0504	0.6337	1.0504	0.2527	0.0000
z.	II	OcclusionFormer	FO		1.6352	1.8839	4.3020	7.6507	0.6337	1.0504	0.6337	1.0504	0.2527	0.0000
aa.	II	OcclusionFormer	FO		1.6607	1.9449	4.3411	7.7593	0.6337	1.0504	0.6337	1.0504	0.2527	0.0000
ab.	*II*	*Avg. OcclusionFormer*	*FO*		*1.6607*	*1.9512*	*4.3042*	*7.6590*	*0.6337*	*1.0504*	*0.6337*	*1.0504*	*0.2527*	*0.0000*

APPENDIX F
BOXPLOTS VERSUS LAST OBSERVED TIMESTEP



(a) MINADE

(b) MINFDE

(c) MEANADE

(d) MEANFDE

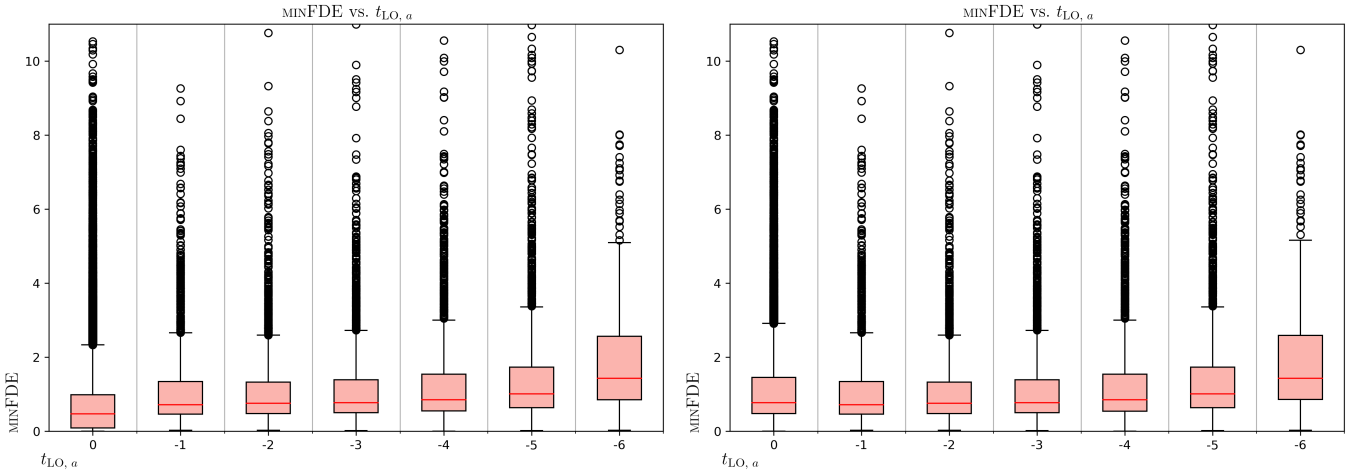
Fig. 11: Boxplot figures of OcclusionFormer's (model #III) performance scores versus $t_{\text{LO},a}$.

Observing the boxplots in Figure 11, we remark the "performance shifts" between $t_{\text{LO},a} = \{0, -1\}$ and $t_{\text{LO},a} = \{-5, -6\}$ across every score. The appearance of these shifts is noticeably different than the more subtle, progressive degradation of performance scores between $t_{\text{LO},a} = \{-1, -5\}$.

We conducted investigations in order to identify the potential causes for those particular shifts in the performance of our model. The first potential explanation for the $\{0, -1\}$ shift stems from the fact that our occlusion simulator generates *static* occlusion cases (i.e., occlusion cases where the configuration/arrangement of the occluded space does not change throughout the entire course of an instance $[-T_{\text{obs}}, T_{\text{pred}}]$). Consequently, every agent that remains immobile throughout the entirety of their past history will always be *fully observed* (stillstanding agents cannot move between W_O/W_V). We theorize that, by the design of our occlusion simulator, we implicitly assign different trajectory types to different occlusion configurations, and different categories of last observed timesteps $t_{\text{LO},a}$.

We investigate the effect of removing stillstanding agents from our test set on the distribution of performance scores within our boxplots. We define stillstanding agents as those who have travelled less than 0.5 m over the entirety of their past history. That is, the cumulative distance between subsequent points in the time interval $t = [-T_{\text{obs}}, 0]$ is less than 0.5 m. Filtering those identities away from the test set, we remark that the $\{0, -1\}$ performance shift is greatly reduced (but not *entirely*), as can be seen in Figure 12. Stillstanding agents are a type of trajectory that is essentially exclusive to the $t_{\text{LO},a} = 0$ category. In fact, the exclusion of stillstanding agents as we previously described removes 32640 trajectories from the 103191 present in our test set, of which only 8 do not belong to the $t_{\text{LO},a} = 0$ category. From this investigation, we observe that stillstanding

agents contribute significantly to a difference in performance scores between $t_{\text{LO},\,a} = \{0, -1\}$ by *reducing* performance scores at $t = 0$, and thereby confirm our suspicion that the occlusion simulator does perform an implicit discrimination of certain trajectory profiles to be assigned to certain $t_{\text{LO},\,a}$ categories.



(a) entire test set.

(b) test set with stillstanding agents removed.

Fig. 12: Comparison of the boxplot figures of OcclusionFormer's minFDE score (model #III), when keeping/removing stillstanding agents from the test set.

When it comes to the $t_{\text{LO},\,a} = \{-5, -6\}$ shift, a clear difference in the distribution of trajectory types is more difficult to identify than for $\{0, -1\}$. It remains however clear that some underlying cause, extraneous to the occlusion duration, is contributing to the sudden degradation of scores in the $t_{\text{LO},\,a} = -6$ category. The main hypothesis we made for this underlying cause was the fact that trajectories within this category provide the prediction model with an input history of only 2 elements, depriving the model from the possibility of making inferences on the acceleration and turn rate. This hypothesis has been rejected: when generating the same boxplot figures again, using the fully observed trajectory homologues of our occlusion simulation trajectory set (i.e., using the performance scores of model #II, categorizing trajectories by their corresponding $t_{\text{LO},\,a}$ assignment in the OS set), the performance scores maintain very similar distributions within the $t_{\text{LO},\,a} = [-1, -5]$ range, but the sudden degradation at $t_{\text{LO},\,a} = -6$ remains (see Figure 13).

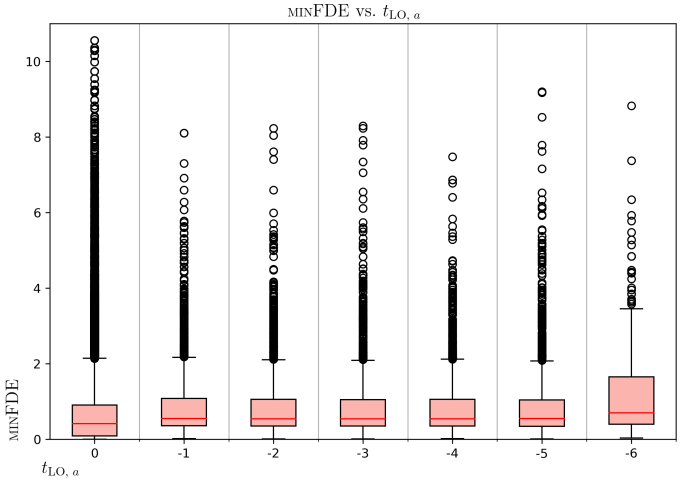


Fig. 13: Boxplot figure of OcclusionFormer's minFDE score (first iteration of model #II, evaluated on "FO"). All performance scores shown here were obtained from the model operating on *fully observed* trajectories; their assignment to a specific $t_{\text{LO},\,a}$ category is done by using the corresponding occlusion configuration of the trajectory in the "OS" dataset.

This indicates that the underlying cause in this performance degradation is not due to a difficulty of the model with handling $t_{\text{LO},\,a} = -6$ cases, but rather, an implicit selection of certain trajectories by the occlusion simulator to be assigned to this category of occlusion configuration.