

Poly-Autoregressive Prediction for Modeling Interactions

Appendix

9. Related Work: Case Studies

Action recognition/forecasting. Recent advancements in action recognition have significantly improved our ability to understand and classify human activities in videos, starting with the SlowFast network [16], which introduced a two-pathway approach that processes visual information at different frame rates to capture slow and fast motion patterns. This resembles ventral and dorsal pathways of human brain for action understanding and object recognition, respectively. Following the rise of vision transformers [14], MViT [15] showed promising results on action understanding benchmarks with multi-scale transformers. Recently, Hiera [42], presented a hierarchical vision transformer that leverages multi-scale feature learning to enhance action recognition performance, by utilizing masked image pretraining as in MAE [21]. LART [40], expanded on these prior works by incorporating 3D human pose trajectories to achieve better action prediction performance. Some works forecast and anticipate actions [24]. [48] perform action forecasting on videos using relational information. [29] train an RNN on long-form videos to contextualize the long past and improve predictions of the future.

Car trajectory prediction. In the autonomous driving literature, forecasting the future motion of cars is a popular problem [11, 22], facilitated by an influx of datasets in recent years [5, 7, 49]. Many important approaches have focused on modeling the environment in conjunction with multiple agents [6, 12, 43]; our framework only focuses on multi-agent interactions. More recent advancements have seen the rise of transformer-based methods in trajectory prediction [35, 61]. In particular, MotionLM [44] forecasts multi-agent trajectories by encoding motion in discrete acceleration tokens and passing these tokens through a transformer decoder that cross-attends to the Wayformer [31] scene encoder. When applying the PAR framework to trajectory prediction, we use acceleration tokens to discretize car motion.

6D pose estimation and hand-object interaction. 6D pose estimation from monocular camera images has been extensively studied [26, 52, 55, 60]. Additionally, a related area of research known as 6D object pose tracking leverages temporal cues to improve the accuracy of 6D pose estimation in video sequences [13, 56–58]. There is also significant interest in learning state and action information of hands and objects through hand-object interaction data, sourced from both curated and in-the-wild video data [59]. Of particular relevance to 6D pose estimation is the DexYCB dataset [8], which contains 1000 videos of human subjects interacting

with 20 objects on a table with randomized tabletop arrangements and 6D object poses. For the third case study in this paper, we propose using the PAR framework to model hand-object interactions, demonstrating that incorporating the hand as an agent provides a useful prior for enhancing object rotation and translation predictions compared to AR modeling.

10. Additional PAR Framework Details

10.1. Implementation details

Token embeddings and loss. For discrete tokens, we use a standard learned embedding layer to convert the tokens to the hidden dimension d_h of the model. To compute the loss, we use a classification loss between the predicted distribution (output logits) and the input ground truth tokens. For continuous inputs with dimension d , we learn a linear layer to project from d to d_h , and a second un-projection layer to project from d_h back to d . To compute the loss, we take the last hidden state of the model, un-project it back to d , and then compute a regression loss in the original token space.

Next-timestep prediction. In standard autoregressive models (such as our single-agent model in section 3.2) the next token prediction objective is enforced by computing the loss on an input and predicted target that are both shifted by one. Now, we will instead shift both by N , so that for a given token, the model operating on our flattened sequence of $N * T$ tokens predicts a token corresponding to the next timestep but the same agent.

Inference. For a single-agent model, starting with an initial sequence history of h tokens, we feed these into the model to get the next token, which we then append to our sequence to form a new sequence of $h + 1$ tokens. We repeat this process to generate arbitrarily long sequences.

For our multi-agent model, we start with a ground-truth history of h timesteps, which corresponds to $h * N$ tokens, including the ego agent, agent N . Inputting this to the model results in the last output token being our ego agent at timestep $h + 1$. Then, to predict the next timestep $h + 2$, we concatenate to the ground truth $h * N$ tokens the ground truth of agents $1 : N - 1$ at timestep $h + 1$ and our prediction of the ego agent at timestep $h + 1$, and we repeat this process.

For a multiagent next-token prediction ablation, to predict the ego agent at timestep $h + 1$, we feed in the ground truth of agents $1 : N - 1$ at $h + 1$ to our model to predict our ego agent, agent N , at timestep $h + 1$. We continue this process of giving our model the ground truth tokens of agents $1 : N - 1$ to predict agent N at each timestep. This means

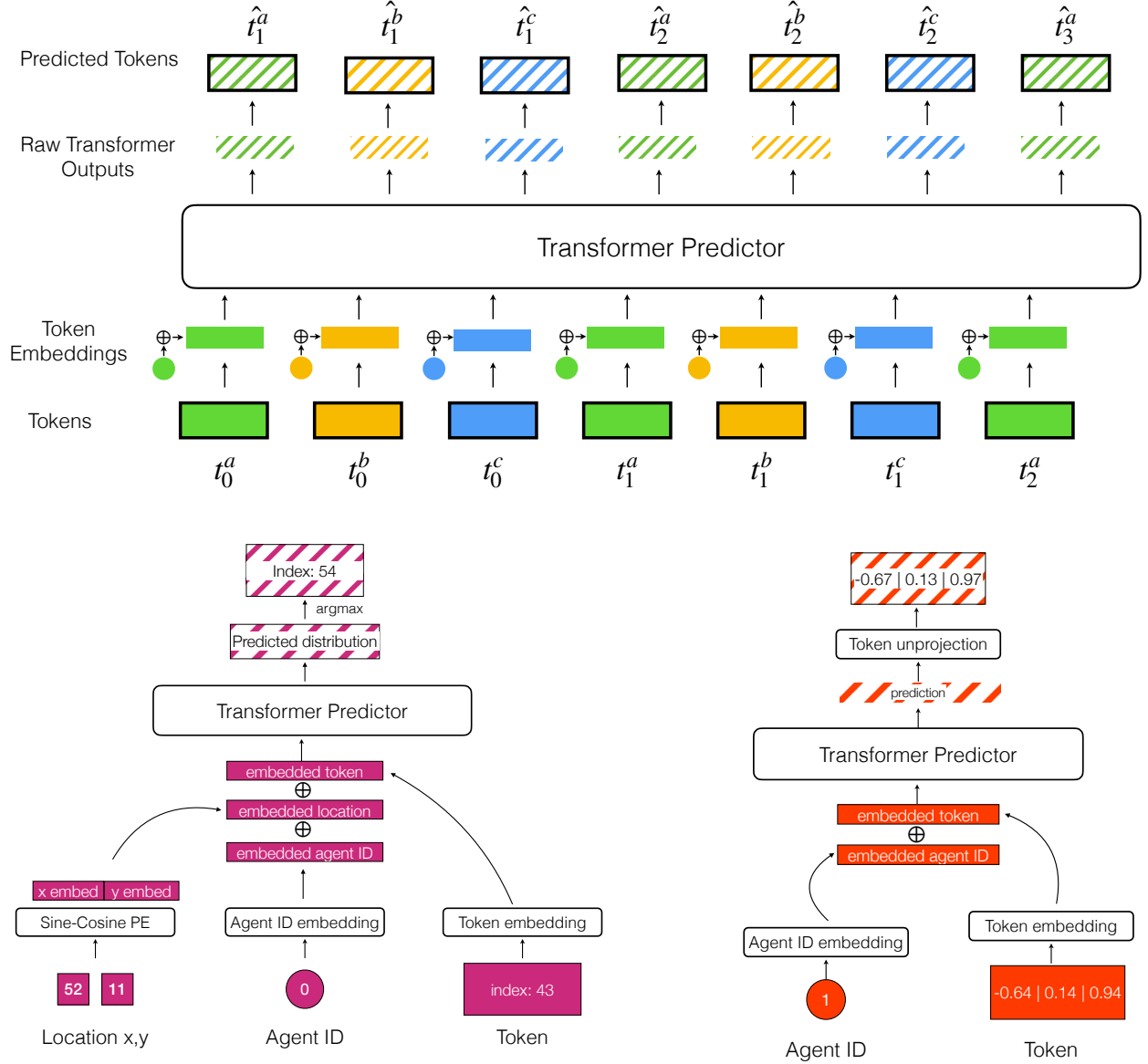


Figure 9. **Architecture** Top: *PAR training with teacher forcing*. Here, we see that the tokens are input to the model and projected to a token embedding dimension of size $d_h = 128$, where embeddings such as the agent ID embedding can be summed. Then, the transformer output is sampled (discrete tokens; left) or deprojected (continuous tokens; right) to produce a predicted token. At inference, the same embeddings are summed and the same conversion from transformer output to token space occurs. Bottom left: *PAR on discrete tokens*. In this case, the token in question is an integer index into a codebook and the standard transformer embedding layer is used to project to d_h . Here we show the agent ID embedding and our Location Positional Encoding summed to the embedded token. After being passed to the transformer, the output is a distribution of logits, from which a token can be sampled—we use argmax in our experiments. In the discrete case, the token is converted back to the actual modality via a detokenizing step. Bottom right: *PAR on continuous tokens*. Here, the token is projected to d_h using a learned projection layer. The output from the transformer is of size d_h , and we have a trained deprojection layer to project back from d_h to the token dimension. We do not add any operations after the deprojection to constrain predicted token values.

that this ablation necessarily has more information available at inference time.

teacher forcing, and in-depth details of how discrete and continuous tokens are processed.

10.2. Architecture Details

A detailed diagram of our architecture can be seen in Fig. 9, where we show the overall PAR method during training with

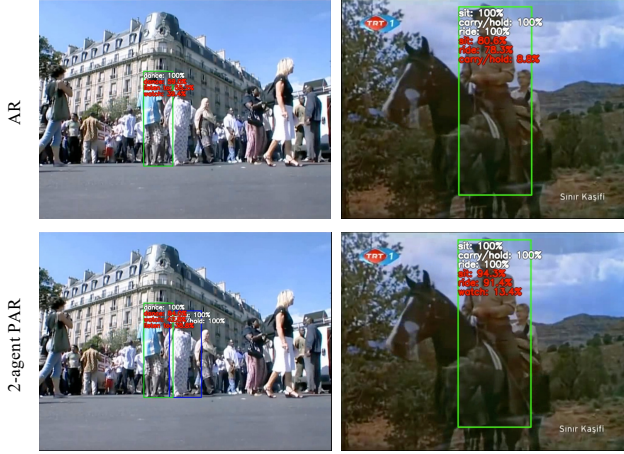


Figure 10. **Qualitative examples of action prediction on single-person actions.** While *dance*, *ride* and *sit* are not multi-person actions, our method is able to predict them more accurately in these examples (and overall by margins of +2.0, +1.0 and +2.7 mAP points respectively, see Fig. 11). This is likely because these actions are co-occurring between two agents who are both partaking in the activity in question, and our 2-agent PAR model is able to reason over this. Note that in the horse-riding example, the second agent is present in the history, but the tracking failed and they are not present at the timesteps we are predicting (see full history and predictions in supplementary video). This context in the history is sufficient to help our PAR model make better predictions.

11. Additional Experimental Results

11.1. Additional Results on AVA Action Forecasting

We see the results of our 1-agent AR and 2-agent PAR methods on the AVA 1-person classes in Fig. 11. On the vast majority of these classes, our 2-agent PAR method is still stronger than 1-agent AR. This is likely because there are many actions that people carry out together, whether it be 2 people both *dancing* (+2.0), *walking together* (+11.3), *watching TV* (+1.7), or *listening to music* (+5.4). We show a qualitative example of some of these single-agent action classes in Fig. 10, where we see that the PAR approach helps the model make better predictions.

Evaluation dataset The AVA test set annotations are not released. Since we are focused on action forecasting from ground-truth past annotations instead of predicting actions from video frames, we evaluate on the validation set.

11.2. Additional Results on Object Pose Estimation

See Figures 12 and 13 for more qualitative results on rotation and translation predictions, respectively.

Additionally, we perform an ablation on the effect of the agent ID embedding on the performance of our system for the hand-object interaction. The results are presented on the test split of the dataset, using the best checkpoint selected

Type	Method	Ag-ID embd	MSE (m^2) ↓	GEO (rad) ↓
Transl	1-ag AR	N/A	3.68×10^{-3}	-
Transl	2-ag PAR	✗	2.26×10^{-3}	-
Transl	2-ag PAR	✓	2.17×10^{-3}	-
Rot	1-ag AR	N/A	-	0.919
Rot	2-ag PAR	✗	-	0.895
Rot	2-ag PAR	✓	-	0.837

Table 7. **Object pose estimation PAR ablation.** All results are on the test split of the dataset. We see that for the case of object pose forecasting, using the agent ID embedding helps improve the performance on both translation and rotation prediction.

Method	Timestep pred	Ag-ID embd	ADE	FDE ↑
1-ag AR	N/A	N/A	1.44	3.57
3-ag AR	✗	✗	1.36	3.37
3-ag PAR*	✗	✓	1.36	3.37
3-ag PAR*	✓	✗	1.36	3.35
3-ag PAR	✓	✓	1.35	3.34

Table 8. **Car trajectory prediction PAR ablation.** All results use acceleration tokens, and the 3-agent methods use the location positional encoding.

based on performance on the validation split after 500 epochs of training (to convergence) with agent ID embeddings. As seen in Table 7, removing the agent ID embedding for the 2-agent PAR has a more significant effect on the rotation estimation than the translation estimation. This may be attributed to the fact that in the translation estimation, both the hand and the object tokens are represented as 3D translations, but in the case of the rotation estimation, the hand token is represented as a 3D translation, while the object token is represented as a quaternion. Thus, having the agent ID embedding is helpful for tokens that measure different types of quantities.

Please note that we do not include an ablation on the next timestep prediction, because that would entail inputting the hand token and predicting the object token. Shifting by 1 instead of by 2 (our number of agents – see Figure 3 and Section 10.1) would entail computing a loss on tokens of different dimensions which is not possible. The next-timestep prediction component of the PAR framework is necessary for a task such as this one with multiple data modalities.

11.3. Additional Results on Car Trajectory Prediction

We conduct an ablation on our the agent ID embedding and next timestep prediction for the 3-agent PAR model in table 8. We see that in our cars case study, our 3-agent PAR model slightly outperforms the 3-agent models without the

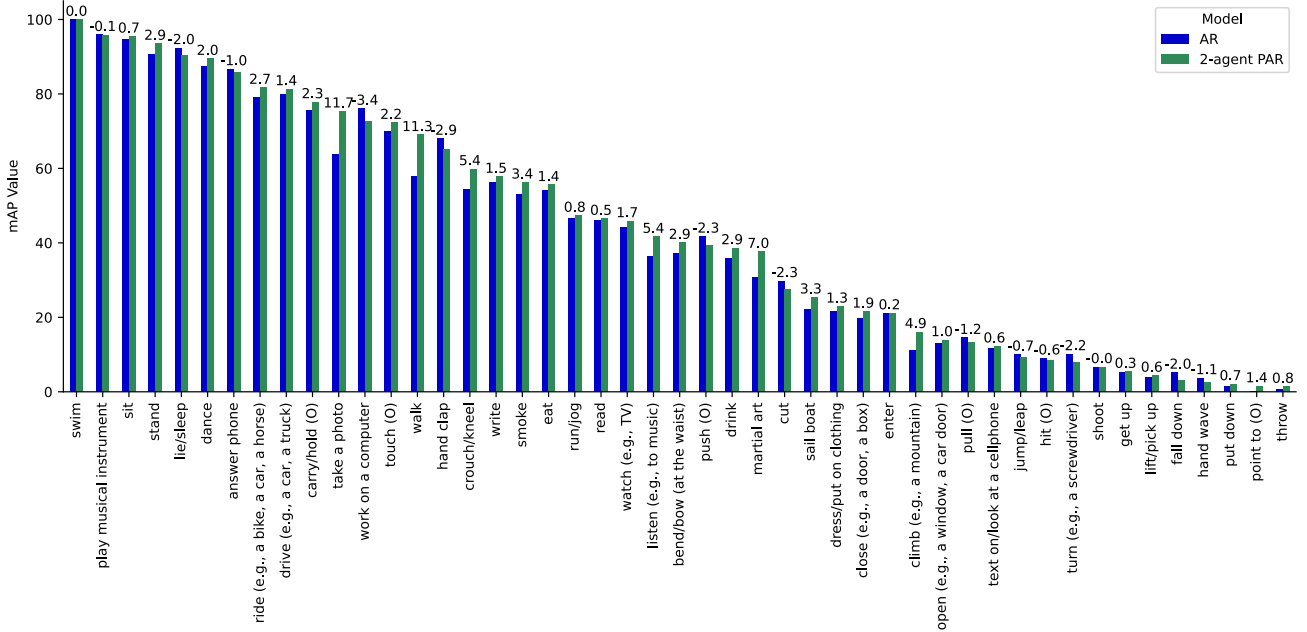


Figure 11. **Per-class mAP on AVA single-person actions.** On these actions, our PAR method is still stronger for the majority of action classes as compared to single-agent AR. For instance, we get an absolute 11.3 mAP gain on walking - people often walk in groups, so it makes sense that this action would benefit from our PAR method.

LPE	Method	ADE ↓	FDE ↓
✗	3-agent PAR	1.40	3.44
✗	10-agent PAR	1.39	3.43
✓	3-agent PAR	1.35	3.34
✓	10-agent PAR	1.35	3.35

Table 9. **Car trajectory prediction with 3 vs 10 agents.** All results use acceleration tokens.

next timestep prediction and agent ID embedding. Note that not using next timestep prediction actually results in the model having more information at inference time (see the last paragraph of Sec. 10.1), so this combined with nuScenes being a relatively simple dataset, and the small acceleration-based motion token vocabulary could explain why the results are comparable.

11.4. Results with more than Three Agents

We experiment with using 10-agent PAR instead of 3-agent PAR for car trajectory prediction on nuScenes, Table 9. We see that our model has similar performance across both numbers of agents. While PAR using our small model can still learn and reason through the increased complexity of 10 agents, the increased agents do not help. We hypothesize that for cars driving on the road, there are two most influential agents: the car directly in front and the one in the

adjacent lane, especially during lane changes. Therefore, we hypothesize that beyond two neighboring agents, other agents add limited value, especially on a simple dataset such as nuScenes, which is supported by these results.

Since AVA scenes often involve at most two people and DexYCB inherently includes only one hand and one object, we only go beyond two agents on nuScenes. However, PAR’s ability to handle more than 3 agents could be useful in tasks with complex group interactions, such as team sports like basketball, where many agents play key roles simultaneously.

12. Additional Case Study Implementation Details

We stabilize learning by using Exponential Moving Average (EMA) for training our experiments with a decay rate of 0.999 for action prediction and object translation/rotation estimation, and 0.9999 for car trajectory prediction.

12.1. Car Trajectory Prediction

Tokenization Instead of discretizing the xy position space, we discretize the motion, resulting in discrete velocity or acceleration tokens computed as follows. We take each agents ground truth trajectory (past and future), shift it so that the trajectory starts at $x, y = 0, 0$, and then rotate the trajectory such that its initial heading at $t = 0$ is 0 radians. We divide velocity space into 128 even bins in $[-18, 18]$ meters. We

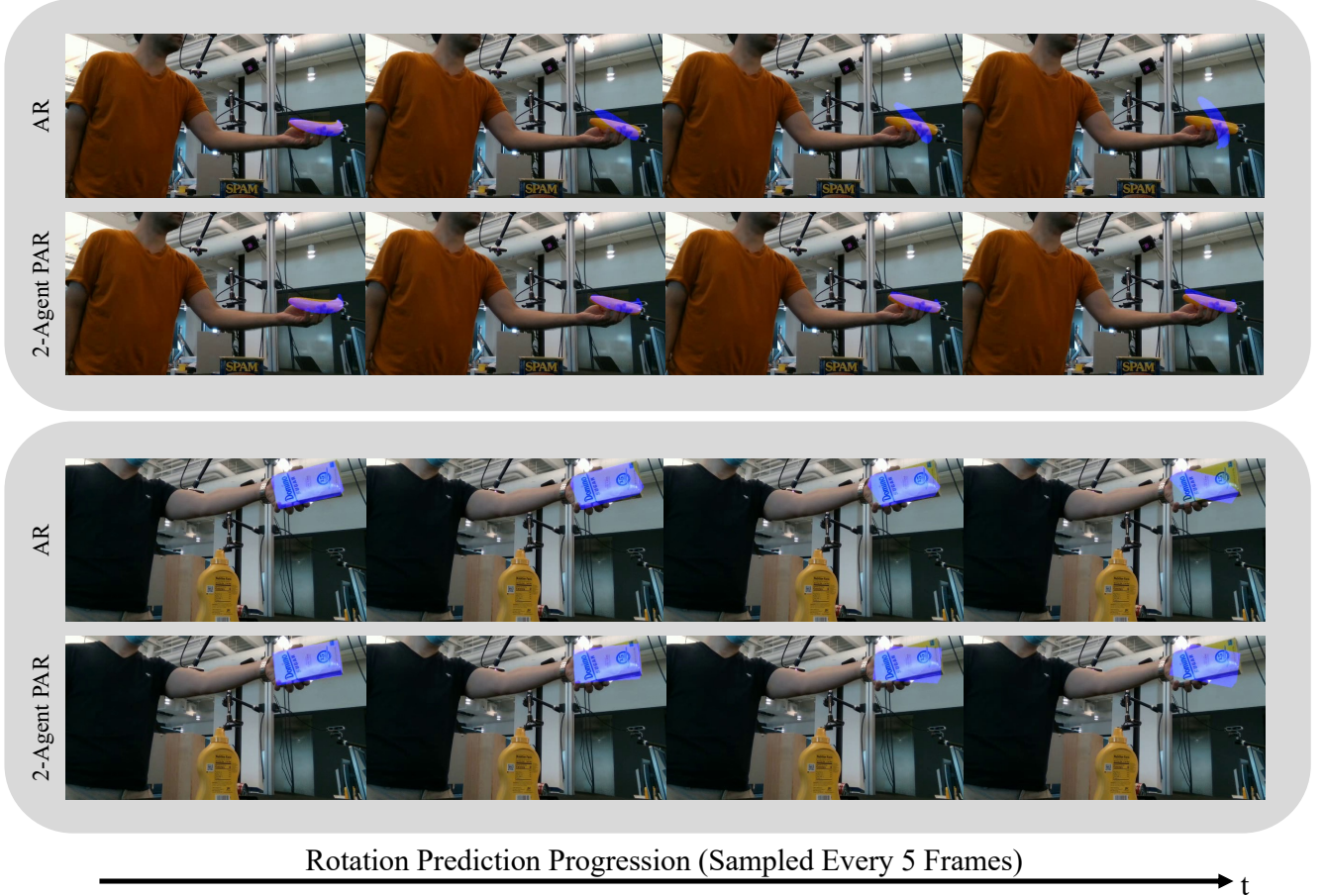


Figure 12. **Rotation prediction qualitative results.** We show results from two videos. The projected 3D model in blue has the ground-truth translation for visualization purposes and our predicted rotation. In the top row (AR), the results depict the object of interest as the sole agent, while the bottom row (2-agent PAR) demonstrates improved performance by incorporating the human hand as a second agent in the grasping interaction.

then, separately for x and y , take the difference between each pair of coordinates in the trajectory, to get a length $T - 1$ sequence of deltas. Each of these deltas is mapped to a bin index.

We first experimented with velocity tokens, taking the Cartesian product of bin space to give each xy -delta one single integer index between 1 and $128 * 128 = 16384$. To get acceleration tokens, we take the difference between each x delta and y delta, and bin these differences into 13 bins. We then take the Cartesian product of bin space to get a vocabulary between 1 and $13 * 13 = 169$.

Location Positional Encoding (LPE) We implement our location positional encoding as follows.

We first compute relative location to the agent we are predicting (the “ego” agent) at the first timestep of the history. The ego agent trajectory is shifted to be at location $(0, 0)$ at time $t = 0$, and all other agents are shifted to be relative to the ego agents position. We also rotate the ego agent trajectory to have a heading of 0, and rotate all other agents

trajectories relative to this ego agent trajectory.

We normalize these relative locations (in meters) to be between 0 and 1. We then quantize these normalized locations to be an integer between 0 and 100. We next pass these locations (x and y separately) into a sin-cos positional encoding. Instead of operating on sequence position indices, the positional encoding operates on the quantized locations. We compute separate positional encodings for x and y . The encoding dimensions is half of the hidden dimension, and we concatenate the x and y encodings to get one encoding. We then sum the result of this encoding to the model inputs at training for the full trajectory (history and future).

At inference, we compute this encoding on the full trajectory (history and future) for agents 1 to $N-1$, but for our ego agent, we only use the history location ground truth. To get the future locations, at each sampling step, we integrate over our velocity or acceleration token to update the predicted location one step at a time, and then pass that location into our encoding.

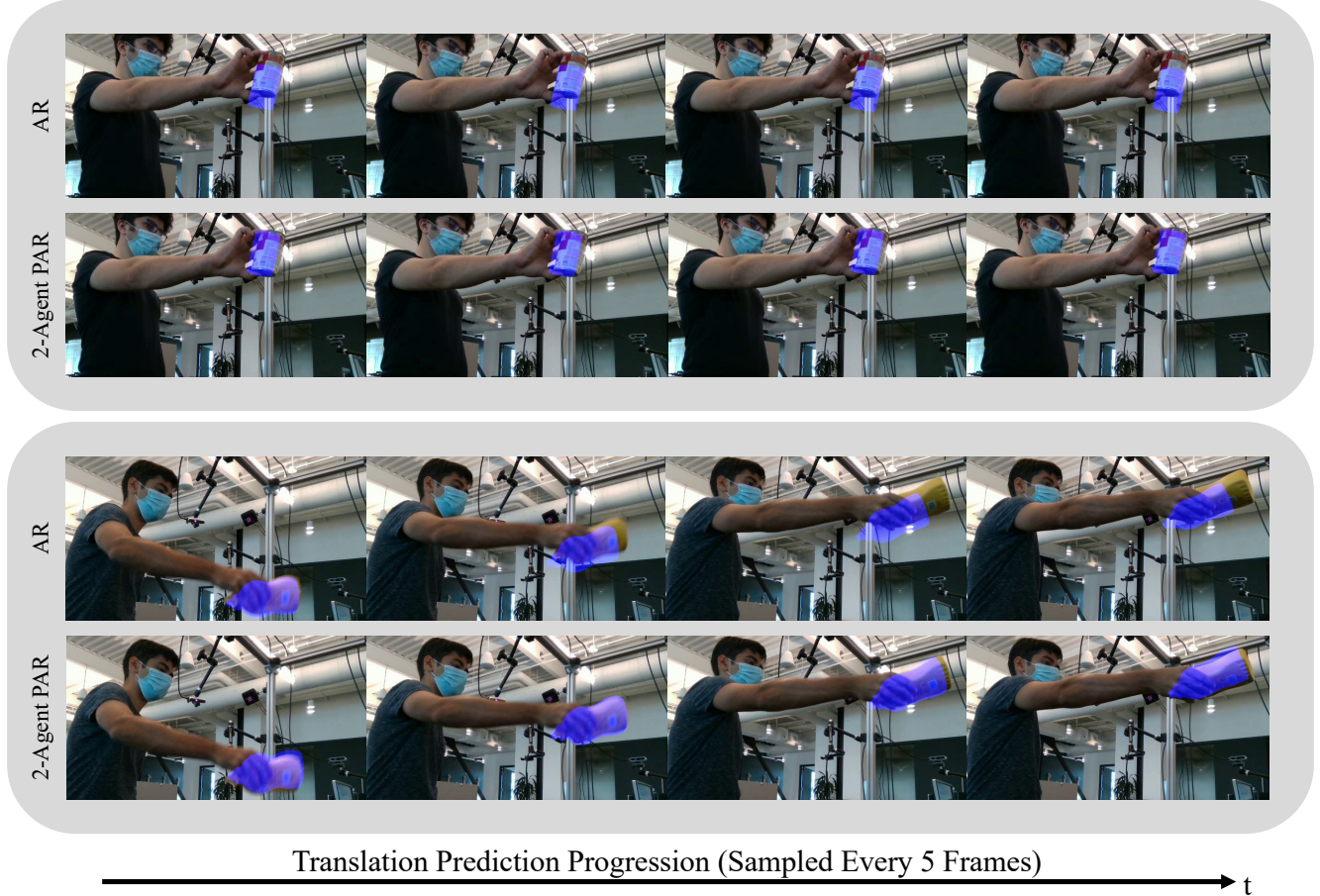


Figure 13. **Translation prediction qualitative result.** We show results from two videos. The projected 3D model in blue has the ground-truth rotation for visualization purposes and our predicted translation. In the top row (AR), the results depict the object of interest as the sole agent, while the bottom row (2-agent PAR) demonstrates improved performance by incorporating the human hand as a second agent in the grasping interaction.

Evaluation dataset Since the nuScenes test set can only be evaluated by submitting to the leaderboard, but we are interested in demonstrating the effectiveness of PAR over AR, we evaluate on the nuScenes validation set.

12.2. Object Pose Estimation

As mentioned in the main text, we use relative translations with respect to the hand location for computing the PAR results. In translation prediction, the relative translation is defined with respect to the hand’s position at the current timestep. This means the hand token is always treated as the origin (a zero vector) at each timestep, while the object translation corresponds to the difference between the current positions of the hand and the object.

For rotation prediction, however, we cannot compute the object rotation relative to the hand because our dataset does not provide hand rotation information. Instead, for the hand token, we compute the relative translation with respect to the hand’s position in the first frame of the sequence. This

ensures the hand location is only treated as the origin in the first timestep, enabling the hand’s motion to influence the rotation prediction throughout the sequence.

For training both prediction tasks, we stabilize learning by employing Exponential Moving Average (EMA); rotation prediction uses a decay rate of 0.999, while translation prediction uses a decay rate of 0.99. Both prediction tasks use the AdamW optimizer with learning rate of 10^{-4} .

13. Supplementary Video

Our supplementary video contains the full video for all qualitative results shown in this paper, and additional qualitative results. No temporal smoothing is applied, nor are any other modifications made to the results shown in the videos.