Supplementary Materials
PHASE: PHysically-grounded Abstract Social Events for Machine Social Perception

Aviv Netanyahu* **Tianmin Shu*** **Boris Katz** **Andrei Barbu** **Joshua B. Tenenbaum**
{avivn, tshu, boris, abarbu, jbt}@mit.edu
Massachusetts Institute of Technology, Cambridge, MA 02139

Method	Goal			Relation
	Top-1	Top-2	Top-3	
SIMPLE (Initial)	0.795	0.825	0.835	0.82
SIMPLE (Global)	0.835	0.865	0.880	0.84
SIMPLE (Full)	0.870	0.890	0.910	0.88

Table 1: Ablation study on the effect of proposal updates based on local estimation for Task 1. The evaluation is using the test set of PHASE. Initial, Global, and Full represent SIMPLE with only the initial proposals, SIMPLE using global estimation for updating the proposals, and SIMPLE as proposed in Algorithm 2.

Additional Experimental Results

Ablation Study (Evaluation on PHASE)

To evaluate the effect of the proposal update based on location estimation in SIMPLE, we compare the full approach with (i) a variant without proposal update, and (ii) a variant with update based on the complete trajectories instead of location estimation (also with 6 iterations). We report the performance for Task 1 on the test set of PHASE in Table 1. The results demonstrate that local estimation indeed can help find better proposals through a handful of iterations.

Details of Joint Physical-Social Simulation

Our joint physical-social simulation is described in Algorithm 1, which includes a physical simulation $\mathcal{T}$, and a hierarchical planner which consists of a high-level planner (HP) and a low-level planner (LP). Given the scene configuration, the simulation updates the belief particles based on new observations, uses the hierarchical planner to sample actions for all agents based on the updated particles, feeds the actions to the physics engine to simulate one step, and renders 5 frames of video based on the simulated physical states. The final video has a frame rate of 20 FPS. We discuss more implementation details as follows.

Predicates, Symbolic States, Goals, and Subgoals

In our simulation, we define a set of predicates as summarized in Table 2. These predicates and their negations are used to (i) convert a physical state into a symbolic state, and also (ii) become a subgoal space that our hierarchical planner considers for the high-level plans.

Furthermore, the final goal states for physical goals and social goals of agents are also represented by a subset of these predicates, i.e., ON(*agent/object*, *landmark*), TOUCH(*agent*, *agent)*, and their negations.

Algorithm 1: Joint Physical-Social Simulation

Input: $g_1, g_2, \alpha_{12}, \alpha_{21}, f_1, f_2$, and initial state s^1
Output: Abstract social event $s^{1:T}$
for *agent* $i = 1, \cdots, 2$ **do**
  Initialize belief particles $\{b^0_{i,k}\}_{k=1}^K$;
end
for *time steps* $t = 1, \cdots, T$ **do**
  for *agent* $i = 1, \cdots, 2$ **do**
    Update observation o_i^t;
    Update belief particles $\{b^t_{i,k}\}_{k=1}^K$ based on o_i^t;
    Set the other agent $j \leftarrow \{1, 2\} \setminus \{i\}$;
    for *each particle* $k = 1, \cdots, K$ **do**
      Get subgoal $h^t_{i,k} \leftarrow \text{HP}(g_i, g_j, \alpha_{ij}, b^t_{i,k})$;
    end
    for *subgoal* $h \in \mathcal{H}$ **do**
      Esitmate value $V(\mathcal{B}_i^t, h, g_i, g_j, \alpha_{ij}) = \frac{1}{K}\sum_{k=1}^K \mathbb{1}(h = h^t_{i,k}) - \frac{\lambda}{\sum_{k=1}^K \mathbb{1}(h=h^t_{i,k})}\sum_{k=1}^K \mathbb{1}(h = h^t_{i,k})\hat{C}(b^t_{i,k}, s_g)$;
    end
    Select subgoal $h^t_{i,*} = \arg\max_h V(\mathcal{B}_i^t, h, g_i, g_j, \alpha_{ij})$;
    Get belief particles $\tilde{\mathcal{B}}_i^t$ that correspond to $h^t_{i,*}$;
    Get action $a_i^t \leftarrow \text{LP}(\tilde{\mathcal{B}}_i^t, h^t_{i,*})$;
  end
  Update state $s^{t+1} \leftarrow \mathcal{T}(s^t, \{a_i^t\}_{i=1}^2, \{f_i\}_{i=1}^2)$;
end

Hierarchical Planner

For the **high-level** planner we use A* to search for a plan of subgoals for $N = 2$ agents based on $K = 50$ belief particles.

*Equal contribution.

Predicate	Definition
ON(*agent/object*, *landmark*)	An entity is on a landmark
TOUCH(*agent*, *agent/object*)	An agent touches another entity
ATTACH(*agent*, *object*)	An object is attached to an agent's body
CLOSE(*agent/object*, *agent/object/landmark*)	An entity is within a certain distance away from another entity or a landmark

Table 2: Predicates and their definitions. Note that we also consider their negations, which are not shown in the table for brevity.

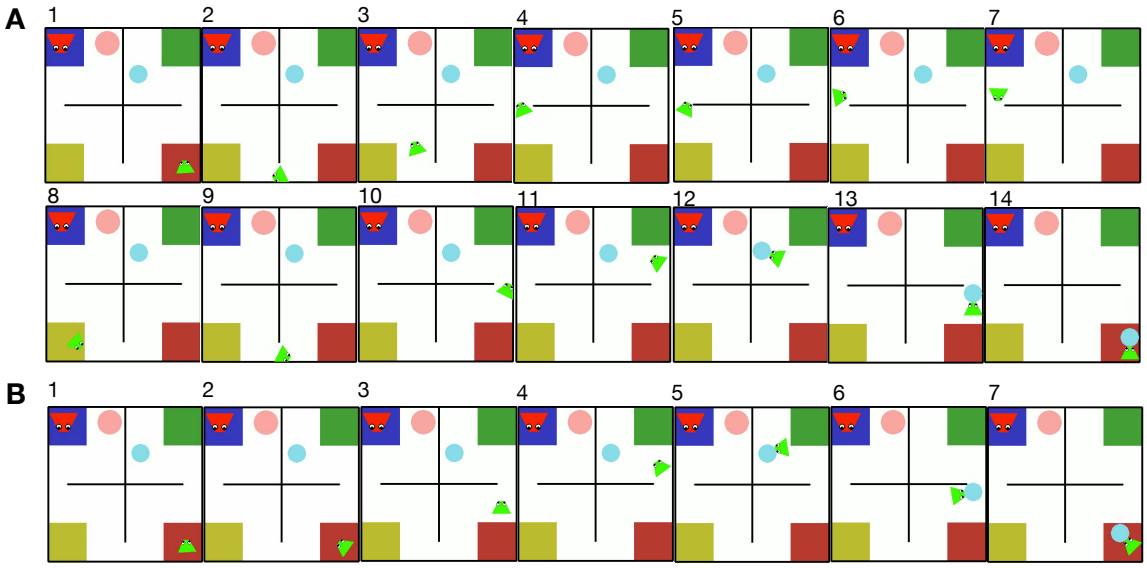

Figure 1: Illustration of the effect of estimated value function for the high-level planner. The numbers indicate temporal order of the frames. In both sequences, the green agent's goal is to move the blue object to the red landmark in the bottom-right corner. Since it does not see the blue object initially, it needs to first find the object. (**A**) The sequence when $\lambda = 0$. since there is more unseen space in the left part of the environment, it is more likely that the blue object is in the left part. So the green agent first searches the left part when not considering the cost of doing so. (**B**) The sequence when $\lambda = 0.05$. When considering the cost, it is worthwhile for the green agent to search the nearby region first. The chance of finding the blue object there is slightly lower than the left region, but the resulting cost is considerably lower. In particular, it first looks around (frame #2) and then proceeds to search the upper-right part (frame #3 and #4). This comparison demonstrates that an appropriate λ could give us more natural agent behaviors under partial observability.

To ensure a subgoal selection for simulating natural agent behavior without expensive computation, we design a heuristics-based value estimation $V(\mathcal{B}_i^t, h, g_i, g_j, \alpha_{ij})$ for each subgoal as shown in Algorithm 1. This value function favors subgoals that are more likely to be the best subgoal in the true state (i.e., high frequency subgoals generated by all belief particles) and have lower cost (i.e., $\hat{C}$ estimated by the distance from the current state to the final goal state according to a given belief particle). By changing the weight λ, we are able to alter the agent's behavior. Figure 1 demonstrates an example of how λ affects the agent's behavior. In practice, we find $\lambda = 0.05$ offers a good balance and can consistently generate natural behaviors.

For the **low-level** action planner, we use POMCP (Silver and Veness 2010) with 1000 simulations and 10 rollout steps. For exploration in POMCP, we adopt a variant of PUCT algorithm introduced in Silver et al. (2018), where we use $c_{\text{init}} = 1.25$ and $c_{\text{base}} = 1000$.

Belief Representation and Update

Each agent's belief is represented by $K = 50$ particles in the simulation. Each particle represents a possible world state that is consistent with the observations. The state in a particle includes the environment layout, and physical properties of each entity — shape, size, center position, orientation of the body, linear and angular velocity, and attached entities.

Each particle is updated with the ground truth properties of *observed* entities: the agent itself, other entities in its field of view (approximated by 1×1 grid cells on the map) or entities in contact with the agent. Entities that are in contact with observed entities are also defined as observed.[1] Contact occurs when entities are attached or collide, and is signaled by agents' touch sensory.

Unobserved entity properties differ between particles. We start by randomly sampling possible initial positions from

[1]This is to ensure that the agent knows (i) whether there is *any* other agent grabbing the same object it is currently grabbing, and (ii) whether an observed agent is grabbing *any* object.

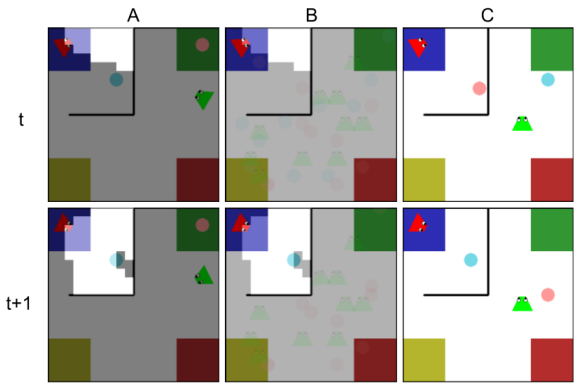

Figure 2: Illustration of how the red agent updates its belief using $K = 10$ particles. (**A**) True states s^t (top) and s^{t+1} (bottom). The bright pixels indicate the red agent's field of view. (**B**) $\{b^t_{\text{red},k}\}_{k=1}^K$ (top) and $\{b^{t+1}_{\text{red},k}\}_{k=1}^K$ (bottom). The states in all the particles are visualized together. At step $t+1$ the red agent observes the blue object via its field of view. All particles are then updated accordingly with the ground truth properties of the blue object, and the inconsistent belief states are also resampled. (**C**) The state in one of the belief particles, $b^t_{\text{red},k}$ (top) and $b^{t+1}_{\text{red},k}$ (bottom). The particle is updated with ground truth properties blue object at step $t+1$. The properties of the pink object are resampled at step $t+1$ since its believed position in step t conflicts with the observation at step $t+1$.

the 2D environment and setting other properties (orientation and velocity) to 0. To update a belief particle from t to $t+1$, we first apply the physics engine to simulate one step, where we assume constant motion for entities. Then we check the consistency between the simulated state at $t+1$ and the actual observation at $t+1$. For entities that contradict the observation, we resample their positions and orientations. We then repeat the consistency check and resampling until there is no conflict.

Figure 2 depicts an example of how an agent updates its belief from step t to step $t+1$ based on its observation at step $t+1$.

More Example Events in PHASE

We show more example events in PHASE in the supplementary video.

Details of SIMPLE

We provide a sketch of SIMPLE in Algorithm 2, where $G(\cdot)$ is our simulation, $P(t_{l,m}|\hat{s}^{1:T}_{l,m}, s^{1:T}, \eta) \propto e^{\eta \sum_{\tau=t_{l,m}}^{t_{l,m}+\Delta T} ||\hat{s}^\tau_{l,m} - s^\tau||_2}$. For all experiments, we set $L = 6$, $M = 15$, $\eta = 0.1$, $\beta = 0.05$, and $\Delta T = 10$.

Based on the final proposals and their weights, we can define posterior probability of the relationship between two agents as follows (F, A, N indicates friendly, adversarial, and neutral respectively):

Algorithm 2: Sketch of SIMPLE

Input: $s^{1:T}$, L, M, η, β, ΔT
Output: $\{Y_{L,m}\}_{m=1}^M$ and their weights $\{w_{L,m}\}_{m=1}^M$
for $m = 1, \cdots, M$ **do**
  Initial proposal $Y_{0,m} \sim Q(Y_{0,m}|S^{1"T})$;
  Synthesize trajectories $\hat{s}^{1:T}_{l,m} \leftarrow G(Y_{0,m})$;
  $w_{0,m} = \frac{P(s^{1:T}|Y_{0,m})}{\sum_{k=1}^M P(s^{1:T}|Y_{0,k})}$;
end
for $l = 0, \cdots, L-1$ **do**
  for $m = 1, \cdots, M$ **do**
    Sample a step $t_{l,m} \sim P(t_{l,m}|\hat{s}^{1:T}_{l,m}, s^{1:T}, \eta)$;
    Set $S' = s^{t_{l,m}:t_{l,m}+\Delta T}$;
    Sample a new proposal $Y' \sim Q(Y_{l+1,m}|S')$;
    Synthesize trajectories $\hat{s}^{1:T}_{l+1,m} \leftarrow G(Y')$;
    $\alpha = \min\{1, \frac{Q(Y'|S')P(s^{1:T}|Y')}{Q(Y_{l,m}|S')P(s^{1:T}|Y_{l,m})}\}$;
    Sample $u \sim \text{Uniform}(0,1)$;
    If $u < \alpha$, $Y_{l+1,m} \leftarrow Y'$, otherwise $Y_{l+1,m} \leftarrow Y_{l,m}$;
    $w_{l+1,m} = \frac{P(s^{1:T}|Y_{l+1,m})}{\sum_{k=1}^M P(s^{1:T}|Y_{l+1,k})}$;
  end
end

$$
\begin{aligned}
P(\text{F}|s^{1:T}) = \; & P(\alpha_{ij} > 0 \text{ or } \alpha_{ji} > 0|s^{1:T}) \\
& + P(g_i = g_j|s^{1:T}) \\
& \cdot P(\alpha_{ij} = 0, \alpha_{ji} = 0|s^{1:T}),
\end{aligned} \tag{1}
$$

$$
\begin{aligned}
P(\text{A}|s^{1:T}) = \; & P(\alpha_{ij} < 0 \text{ or } \alpha_{ji} < 0|s^{1:T}) \\
& + P(\text{conflicting } g_i \& g_j|s^{1:T}) \\
& \cdot P(\alpha_{ij} = 0, \alpha_{ji}|s^{1:T}),
\end{aligned} \tag{2}
$$

and

$$
P(\text{N}|s^{1:T}) = 1 - P(\text{F}|s^{1:T}) - P(\text{A}|s^{1:T}), \tag{3}
$$

where conflicting g_i and g_j include two types of scenarios: (i) two agents have the goal of putting the same object on different landmarks, and (ii) agent i has the goal of approaching agent j while agent j has the goal of avoiding agent i.

Bottom-up Proposals

We devise a bottom-up proposal based on heuristics extracted from observed trajectories within a time interval $S^{t_1:t_2}$, i.e., $Y \sim Q(Y|S^{t_1:t_2})$. In this work, the proposal distribution is decomposed into separate terms for proposing goals (g_i, g_j), social utility weights (α_{ij}, α_{ji}), and strengths (f_i, f_j) respectively, i.e.,

$$
\begin{aligned}
Q(Y|S^{t_1:t_2}) = \; & Q(g_i|S^{t_1:t_2})Q(g_j|S^{t_1:t_2}) \\
& \cdot Q(\alpha_{ij}, \alpha_{ji}|S^{t_1:t_2}) \\
& \cdot Q(f_i|S^{t_1:t_2})Q(f_j|S^{t_1:t_2}).
\end{aligned} \tag{4}
$$

We define the goal proposal distribution for an agent by

$$
\begin{aligned}
Q(g|S^{t_1:t_2}) \; & \propto e^{\gamma||s_i^{t_2} - s_g||_2} e^{\gamma(||s_i^{t_2} - s_g||_2 - |s_i^{t_1} - s_g||_2)} \\
& \propto e^{\gamma(2||s_i^{t_2} - s_g||_2 - |s_i^{t_1} - s_g||_2)},
\end{aligned} \tag{5}
$$

where $\gamma = 10$ is a constant weight. Intuitively, if the trajectories have demonstrated either achievement at the end of the period (t_2) or progress towards a goal during the period (from t_1 to t_2), then that goal is likely to be the true goal. For the social utility weights, we first randomly select $u \in \{-1, 0, 1\}$. If $u = 0$, we set both α_{ij} and α_{ji} to be zero; if $u \in \{-1, 1\}$, we randomly select either α_{ij} or α_{ji}, and set it to be u while setting the other one to be zero. This is essentially assuming that there will be at most one agent pursuing a social goal in a social event. For the strengths, we train a 2-layer MLP (64-dim for each layer) using training data in PHASE to estimate the maximum forces that agents can exert.

Additional Details of Human Experiments

Experiment 1

The 23 labels used in this experiment are: not interacting, interacting unintentionally, chasing, running away, stalking, approaching, avoiding, meeting, gathering together, guiding, following the lead (of another agent), playing a game of tag, blocking, fighting, competing, stealing, protecting an object, attacking, hindering, bullying, playing tug of war, helping, collaborating.

Experiment 2

The online game procedure is similar to the setup in PHASE, except that actions are obtained from user input. In each game, there are two players, one for controlling each agent. The players view the environment from separate screens (via different URLs), updated with each agent's observations. Players can use the following actions by pressing keys on their keyboards: 4 directions (forward, backward, right, left), turning right or left , and grabbing or letting go of an object. We reset the velocity of each agent to 0 after each step to make it easier for players to control the agents. Before each session, the players were shown a tutorial on how the agents work (partial observability and the controls). They were given an opportunity to play freely in the game environment to get familiar with the controls. At the beginning of a session, they were told the goals assigned to both players (so they knew each others' goals) and asked to start playing the game to achieve the assigned goals. Each session ended either until the goals of both players were achieved or until the time limit was reached.

Baseline Implementation

For all neural nets, we construct the inputs as a sequence of states of multiple nodes. In particular, a node could be an entity, a landmark, or a wall. For a node, the input at a step includes a 4-dim one-hot vector for type (agent, object, landmark, or wall), color (which also indicates the identity of each entity), size, position, orientation, and velocity. We provide implementation details of each baseline as follows.

2-Level LSTM: We replace the CNN-based visual features in Ibrahim et al. (2016) by node embeddings. Specifically, we encode each node using a 64-dim fully-connected layer followed by an LSTM (64-dim) to get its embedding. For agent nodes, their node embeddings are fed to a 3-layer MLP (64-dim for each layer) and then a softmax layout for goal recognition. After a max pooling over all nodes' embeddings, we get a context feature, which is fed to another LSTM (64-dim) followed by a fully-connect layer and a softmax layer for relation recognition.

ARG: We use the same node embedding approach introduced above. Following the best performing architecture in Wu et al. (2019), we construct a fully-connected graph using dot-product for the appearance relation. We use a fully-connected graph here since the number of nodes is small and the entities are generally not far from each other. The individual action classifier for each agent node (3-layer MLP with 64-dim for each layer) and the group activity classier (3-layer MLP with 64-dim for each layer) are redefined to recognize agents' goals and relation respectively. We use 10 frames from a video to build the temporal graphs. During training, we randomly select 10 frames; for testing, we use a sliding window, and mean-pool the responses to compute the global judgment of the whole video.

Both **2-Level LSTM** and **ARG** are trained using a cross-entropy loss on goal and relation labels. We use Adam (Kingma and Ba 2014) with a learning rate of 0.001 and a batch size of 8.

Social-LSTM: We adopt the same architecture as in Alahi et al. (2016) except that for every step, it outputs a 10-step prediction for each entity node. This is to solve our online prediction task.

STSAT: We use the same architecture as in Huang et al. (2019) for the encoder components. Similarly to the adaption of **Social-LSTM**, the decoder LSTM outputs a 10-step prediction for each entity at each step as well.

We adopt $\mathcal{L}_2$ distance between the prediction and the ground-truth as loss to train the two trajectory prediction baselines. For network optimization, we use Adam with a learning rate of 0.01 and a batch size of 8.

References

Alahi, A.; Goel, K.; Ramanathan, V.; Robicquet, A.; Fei-Fei, L.; and Savarese, S. 2016. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 961–971.

Huang, Y.; Bi, H.; Li, Z.; Mao, T.; and Wang, Z. 2019. Stgat: Modeling spatial-temporal interactions for human trajectory prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, 6272–6281.

Ibrahim, M. S.; Muralidharan, S.; Deng, Z.; Vahdat, A.; and Mori, G. 2016. A hierarchical deep temporal model for group activity recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1971–1980.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. 2018. A general reinforcement learning algorithm

that masters chess, shogi, and Go through self-play. *Science* 362(6419): 1140–1144.

Silver, D.; and Veness, J. 2010. Monte-Carlo planning in large POMDPs. In *Advances in neural information processing systems*, 2164–2172.

Wu, J.; Wang, L.; Wang, L.; Guo, J.; and Wu, G. 2019. Learning actor relation graphs for group activity recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 9964–9974.