Assisted Perception: Optimizing Observations to Communicate State

Siddharth Reddy, Sergey Levine, Anca D. Dragan
Department of Electrical Engineering and Computer Science
University of California, Berkeley
{sgr,svlevine,anca}@berkeley.edu

Abstract: We aim to help users estimate the state of the world in tasks like robotic teleoperation and navigation with visual impairment, where user may have systematic biases that lead to suboptimal behavior: they might struggle to process observations from multiple sensors simultaneously, receive delayed observations, or overestimate distances to obstacles. While we cannot directly change the user's internal beliefs or their internal state estimation process, our insight is that we can still assist them by modifying the user's observations. Instead of showing the user their true observations, *we synthesize new observations that lead to more accurate internal state estimates when processed by the user*. We refer to this method as assistive state estimation (ASE): an automated assistant uses the true observations to infer the state of the world, then generates a modified observation for the user to consume (e.g., through an augmented reality interface), and optimizes the modification to induce the user's new beliefs to match the assistant's current beliefs. To predict the effect of the modified observation on the user's beliefs, ASE learns a model of the user's state estimation process: after each task completion, it searches for a model that would have led to beliefs that explain the user's actions. We evaluate ASE in a user study with 12 participants who each perform four tasks: two tasks with known user biases – bandwidth-limited image classification and a driving video game with observation delay – and two with unknown biases that our method has to learn – guided 2D navigation and a lunar lander teleoperation video game. ASE's general-purpose approach to synthesizing informative observations enables a different assistance strategy to emerge in each domain, such as quickly revealing informative pixels to speed up image classification, using a dynamics model to undo observation delay in driving, identifying nearby landmarks for navigation, and exaggerating a visual indicator of tilt in the lander game. The results show that ASE substantially improves the task performance of users with bandwidth constraints, observation delay, and other unknown biases.

1 Introduction

People cannot directly access the state of the world, and must instead estimate it from sensory observations (Knill and Richards, 1996). Unfortunately, systematic biases in the user's state estimation process can lead to inaccurate beliefs and suboptimal actions. For example, the user may not be able to keep track of many different sensors simultaneously while flying a plane (Mulder, 1999), or navigate with a visual impairment while listening to a smartphone guide exhaustively list all nearby objects (Panëels et al., 2013). Tasks performed over a network, like teleoperating space robots (Fong et al., 2013), may require the user to compensate for unintuitive, intermittent delays in observations. Lens distortions can cause drivers to overestimate distances to obstacles: the warning, "objects in mirror may be closer than they appear," is engraved into the side mirrors of cars.

Short of intervening in human cognition through brain stimulation, or training users to overcome their biases, how can we assist users with performing more accurate perception? The key idea in this paper is that an automated assistant can intervene in human perception by modifying the observations the user receives. Given the user's biases, different observations lead to different state estimates. We

4th Conference on Robot Learning (CoRL 2020), Cambridge MA, USA.

invert this process to 'trick' the person into arriving at the correct estimate: we modify observations so that, when processed by the biased user, they induce an accurate state estimate.

Figure 1 describes our method: the assistant collects observations from the environment, performs state estimation unencumbered by cognitive biases, then shows the user a synthetic, optimized observation that induces accurate beliefs when processed by their biased perception system. These synthetic observations could be constrained to augment real observations – for example, in an augmented reality interface (Zhao et al., 2019) – or could completely replace them – for instance, by replacing the user's video feed for teleoperating a robot. Crucially, this approach does not require knowing the current task the user is performing: rather than inducing the optimal action for the task, we induce accurate state estimates, so that the user can then decide on task-appropriate actions.

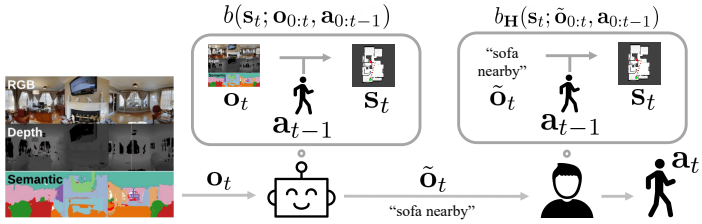


Figure 1: The assistant processes observations $\mathbf{o}_t$ generated by the environment on behalf of the user $\mathbf{H}$, updates its belief distribution over the current state $b(\mathbf{s}_t; \mathbf{o}_{0:t}, \mathbf{a}_{0:t-1})$, then synthesizes an observation $\tilde{\mathbf{o}}_t$ that will induce accurate beliefs $b_{\mathbf{H}}(\mathbf{s}_t; \tilde{\mathbf{o}}_{0:t-1}, \tilde{\mathbf{o}}_t, \mathbf{a}_{0:t-1})$ when shown to the user, enabling the user to make better decisions $\mathbf{a}_t$.

The main challenge is that in order to determine how informative a synthetic observation will be to the user, we need a model of the user's state estimation process. For instance, we might not know the user's bandwidth constraint, observation delay, or even what kinds of biases they have. We introduce an approach for training this model online: we start assisting with an initial model, and collect data of the user suboptimally performing tasks (e.g., navigating to different goals). We assume that upon task completion, the assistant gets to know what the task was – e.g., which goal the user was trying to reach – and can compute a near-optimal policy for that task – e.g., using maximum entropy reinforcement learning (Levine, 2018). We then look in hindsight at the user's actions, and optimize model parameters that lead to state estimates which make the observed actions seem near-optimal. Intuitively, we ask what the user must have believed, and what state estimation process would have led to those beliefs given the observations they received. Our experiments show that the quality of assistance improves as we collect more data and the maximum-likelihood model gets closer to the user's internal state estimation process (Figure 7 in the appendix).

Our primary contribution is the assistive state estimation (ASE) algorithm for optimizing synthetic observations to induce accurate beliefs about the current state in the user. We evaluate ASE through a user study with 12 participants who each perform four tasks: two where the user's bias is known – image classification from bandwidth-limited input, and a driving video game with observation delay – and two where the bias is unknown – navigating a 2D simulation with limited vision where the user only remembers the locations of certain objects but not others, and a lunar lander teleoperation video game. The lander experiment is particularly interesting: the assistant learns to modify the tilt indicator *away* from its real value; actually improving the user's task performance, possibly because users tend to underestimate tilt. Our user studies show that in all domains, ASE substantially improves the user's task performance, relative to a passive baseline that simply shows the user an ambient observation generated by the environment. In addition to the user study, we perform experiments with simulated users that show ASE improves the accuracy of simulated users' internal beliefs.

2 Assisting Users by Optimizing Observations

We formulate the assistance problem as follows. We assume that the environment follows a partially observable Markov decision process (POMDP; Kaelbling et al., 1998) with state space $\mathcal{S}$, observation space Ω, initial state distribution $p^{\text{init}}(\mathbf{s}_0)$, state transition dynamics $p^{\text{dyn}}(\mathbf{s}'|\mathbf{s}, \mathbf{a})$, observation model $p^{\text{obs}}(\mathbf{o}|\mathbf{s})$, and unknown reward function $R(\mathbf{s}, \mathbf{a})$. At each timestep t, the assistant samples an ambient observation $\mathbf{o}_t$ from the environment. The assistant then intervenes and provides the user with a different observation $\tilde{\mathbf{o}}_t \in \Omega$. Since the reward function is unknown, we cannot compute the optimal action and provide the user with an observation that will induce them to take the optimal action. Instead, we aim for a task-agnostic method that assists the user by providing them with an observation that efficiently communicates the current state.

Our approach to this problem is outlined in Figure 1. We assume that the user's state estimation process differs from the assistant's, and that this mismatch leads to suboptimal user behavior. We assist the user by showing them synthetic observations that induce accurate beliefs about the current state. In particular, the assistant first performs state estimation, then optimizes an observation to update the user's beliefs to match the assistant's beliefs. To improve the assistant, we learn a personalized model of the user's state estimation process from demonstrations of suboptimal user actions on known tasks.

2.1 Preliminaries: Assumptions about State Estimation

The standard, recursive Bayesian filter (Thrun et al., 2005) performs state estimation using the belief update,

$$b(\mathbf{s}_t|\mathbf{o}_{0:t}, a_{0:t-1}) \propto p^{\text{obs}}(\mathbf{o}_t|\mathbf{s}_t) \int_{\mathcal{S}} p^{\text{dyn}}(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{a}_{t-1}) b(\mathbf{s}_{t-1}|\mathbf{o}_{0:t-1}, \mathbf{a}_{0:t-2}) d\mathbf{s}_{t-1}. \tag{1}$$

In domains with a small, discrete state space $\mathcal{S}$, we compute exact belief updates using Equation 1. In domains with high-dimensional, continuous states, the belief update in Equation 1 may be intractable to compute. To address this issue, we represent the state estimation process in continuous domains as

$$b(\mathbf{s}_t|\mathbf{o}_{0:t}, \mathbf{a}_{0:t-1}) = \mathcal{N}(\mathbf{s}_t; \mu = f(\mathbf{o}_{0:t}, \mathbf{a}_{0:t-1}), \Sigma = I\sigma^2), \tag{2}$$

where f is a known state encoder that maps a sequence of observations and actions to a continuous, vector-valued state. Although this procedure does not necessarily perform Bayesian belief updates, it enables us to apply our method to domains where the true initial state distribution p^{init}, true dynamics model p^{dyn}, and true observation model p^{obs} are unknown, but a state encoder f is available.

2.2 Synthesizing Observations that Induce Accurate Beliefs

To assist the user, we synthesize an observation $\tilde{\mathbf{o}}_t$ such that, after the user observes $\tilde{\mathbf{o}}_t$ and updates their beliefs about the current state $\mathbf{s}_t$, the user's beliefs will match the assistant's beliefs. Formally, given a history $(\mathbf{o}_{0:t}, \tilde{\mathbf{o}}_{0:t-1}, \mathbf{a}_{0:t-1})$, the assistant decides which observation to provide to the user $\mathbf{H}$ by greedily minimizing the KL-divergence between the assistant's beliefs and the user's beliefs at the end of the current timestep:

$$\tilde{\mathbf{o}}_t \leftarrow \arg\min_{\tilde{\mathbf{o}}_t \in \Omega} D_{\text{KL}}(\underbrace{b(\mathbf{s}_t|\mathbf{o}_{0:t}, \mathbf{a}_{0:t-1})}_{\text{assistant's beliefs}} \,\|\, \underbrace{\hat{b}_{\mathbf{H}}(\mathbf{s}_t|\tilde{\mathbf{o}}_{0:t-1}, \tilde{\mathbf{o}}_t, \mathbf{a}_{0:t-1})}_{\text{assistant's prediction of user's beliefs}}), \tag{3}$$

where $\hat{b}_{\mathbf{H}}$ is the assistant's model of the user's state estimation process. The assistant's beliefs are fixed during this optimization, having already been conditioned on the most recent ambient observation $\mathbf{o}_t$ generated by the environment, while the user's beliefs are conditioned on the synthetic observation $\tilde{\mathbf{o}}_t$ and can thus be optimized. The experiments in Section 3 illustrate how different assistance strategies emerge from Equation 3, such as revealing informative pixels for image classification, undoing observation delay in driving by forward-predicting the current observation, identifying landmarks for navigation, and exaggerating indicators of dangerous states in a landing task.

2.3 Learning Personalized Models of State Estimation

To optimize the synthetic observation in Equation 3, we need to model how the user will update their beliefs in response to observations. We assume the user's unknown state estimation process $b_{\mathbf{H}}$ differs from the assistant's known process b described in Equations 1 and 2. In particular, we assume $b_{\mathbf{H}}$ lies in hypothesis space $\mathcal{B}$. The hypothesis space, which we parameterize as $\mathcal{B} = \{b_\theta : \theta \in \Theta\}$, captures our prior assumptions about possible user biases. If we want to make minimal assumptions about the user's biases, we could define θ to be the weights in a neural network state encoder f_θ that defines the belief update b_θ via Equation 2. If instead we assume that the user performs a Bayesian belief update on each new observation and action, but potentially ignores or misinterprets certain observations, we could define θ to be the observation probabilities $p^{\text{obs}}_\theta(\mathbf{o}|\mathbf{s}) = \theta_{\mathbf{o},\mathbf{s}}$ in Equation 1. In each of our experiments, we make different assumptions about the user, leading to different choices of hypothesis space $\mathcal{B}$.

We search the hypothesis space for a model that best explains user behavior. We assume access to a dataset $\mathcal{D}$ of demonstrations of suboptimal user actions on known tasks. This dataset could be

generated offline by the user without the assistant's help, or generated online while the assistant helps the user. After each demonstration episode, we ask the user what task they were trying to perform during that episode. The task could be specified, for example, through a goal state or a reward function. Let $\tau = (\mathbf{o}_{0:T-1}, \mathbf{a}_{0:T-1})$ denote a demonstration, where T is the episode length. We model the user's actions as rational with respect to their beliefs about the current state:

$$p(\mathbf{a}_t|\mathbf{o}_{0:t}, \mathbf{a}_{0:t-1}; \theta) = \int_{\mathcal{S}} \pi(a_t|s_t) b_\theta(\mathbf{s}_t|\mathbf{o}_{0:t}, \mathbf{a}_{0:t-1}) d\mathbf{s}_t, \tag{4}$$

where the $\pi(a|s)$ is the user's policy, which we assume to be near-optimal for their desired task. We compute π in hindsight after asking the user what task they were trying to demonstrate; e.g., by asking the user to write down the reward function, then doing maximum entropy reinforcement learning. Note that we only need to know a near-optimal policy for the tasks in the demonstrations used to train the user model. We do not need to know the policy at test time when we synthesize observations to assist the user. We assume that the user's policy π for a given task and their belief update $b_{\mathbf{H}}$ do not change once the assistant begins modifying observations using Equation 3. In practice, even if the user adapts their policy or state estimation process to the assistant, this tends to improve performance rather than hurt it.

We then use gradient descent to compute the maximum-likelihood estimate,

$$\hat{\theta} \leftarrow \arg\max_{\theta} \sum_{\tau \in \mathcal{D}} \sum_{t} \log p(\mathbf{a}_t|\mathbf{o}_{0:t}, \mathbf{a}_{0:t-1}; \theta). \tag{5}$$

We select the maximum-likelihood model to be our model of the user's state estimation process: $\hat{b}_{\mathbf{H}} \leftarrow b_{\hat{\theta}}$. This model enables us to predict the effect of an observation on the user's beliefs.

2.4 Assistive State Estimation

Our assistive state estimation (ASE) method is summarized in Algorithm 1. We initialize the user model $\hat{b}_{\mathbf{H}}$ with an initial model b_{init}. In domains with a small, discrete state space $\mathcal{S}$, we assume knowledge of the initial state distribution, state transition dynamics, and observation model, in order to compute Bayesian belief updates using Equation 1. In domains with high-dimensional, continuous states, we instead assume knowledge of a state encoder, so we can estimate the current state using Equation 2. At the start of each timestep t, the assistant collects an observation $\mathbf{o}_t$ from the environment. The assistant then optimizes a synthetic observation $\tilde{\mathbf{o}}_t$ that, when shown to the user, will induce beliefs that match the assistant's (Equation 3). The user sees the synthetic observation $\tilde{\mathbf{o}}_t$, takes an action $\mathbf{a}_t$, and the environment generates the next state $\mathbf{s}_{t+1}$. At the end of each episode, we ask the user what task they were trying to perform, add the episode to the dataset $\mathcal{D}$, and re-train the user model $\hat{b}_{\mathbf{H}}$ using Equation 5.

Algorithm 1 Assistive State Estimation (ASE)

Require $b_{\text{init}} \in \mathcal{B}$ {initial model of user}
if $\mathcal{S}$ is discrete **then**
  Require $p^{\text{init}}(\mathbf{s}_0)$, $p^{\text{dyn}}(\mathbf{s}'|\mathbf{s}, \mathbf{a})$, $p^{\text{obs}}(\mathbf{o}|\mathbf{s})$
  {for assistant's belief update in Equation 1}
else if $\mathcal{S}$ is continuous **then**
  Require state encoder $f(\mathbf{o}_{0:t}, \mathbf{a}_{0:t-1})$
  {for assistant's belief update in Equation 2}
Initialize $\mathcal{D} \leftarrow \emptyset$ {user demonstrations}
Initialize $\hat{b}_{\mathbf{H}} \leftarrow b_{\text{init}}$ {assistant's model of user}
while true do
  $s_0 \sim p^{\text{init}}(\mathbf{s}_0)$
  for $t \in \{0, 1, 2, \ldots, T-1\}$ **do**
    $\mathbf{o}_t \sim p^{\text{obs}}(\mathbf{o}_t|\mathbf{s}_t)$ {assistant sees true observation, then updates its beliefs}
    $\tilde{\mathbf{o}}_t \leftarrow \arg\min_{\tilde{\mathbf{o}}_t \in \Omega} D_{\text{KL}}(b(\mathbf{s}_t|\mathbf{o}_t) \parallel \hat{b}_{\mathbf{H}}(\mathbf{s}_t|\tilde{o}_t))$ (Equation 3) {assistant synthesizes observation}
    $\mathbf{a}_t \sim p(\mathbf{a}_t|\tilde{\mathbf{o}}_{0:t}, \mathbf{a}_{0:t-1})$ {user sees synthetic observation, updates their beliefs, then takes action}
    $\mathbf{s}_{t+1} \sim p^{\text{dyn}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\tilde{\mathbf{o}}_{0:T-1}, \mathbf{a}_{0:T-1})\}$
  $\hat{\theta} \leftarrow \arg\max_\theta \sum_{\tau \in \mathcal{D}} \sum_t \log p(\mathbf{a}_t|\tilde{\mathbf{o}}_{0:t}, \mathbf{a}_{0:t-1}; \theta)$ {assistant learns model of user}
  $\hat{b}_{\mathbf{H}} \leftarrow b_{\hat{\theta}}$

Related work. Schmitt et al. (2017), Daptardar et al. (2019), and Jarrett and van der Schaar (2020) learn a model of the user's internal state estimation process in order to improve inverse reinforcement learning (Ng and Russell, 2000, Baker et al., 2009), whereas we focus on assisting users by controlling observations. Bühler and Weisswange (2020) assist users by communicating informative observations, but require knowledge of the user's reward function at test time, assume a discrete state space, and do not learn a personalized model of the user's internal state estimation process. Hilgard et al. (2019) learn to visualize high-dimensional examples to assist users with one-step classification tasks, whereas we focus on sequential decision-making and make minimal assumptions about the desired task. The closest prior work in assistive navigation for visually-impaired users tackles the problem of planning instructional guidance actions under uncertainty about how the user will respond to instructions (Ohn-Bar et al., 2018). In our work, we take a fundamentally different approach to assistance: help the user estimate their current state, then decide for themselves which action to take.

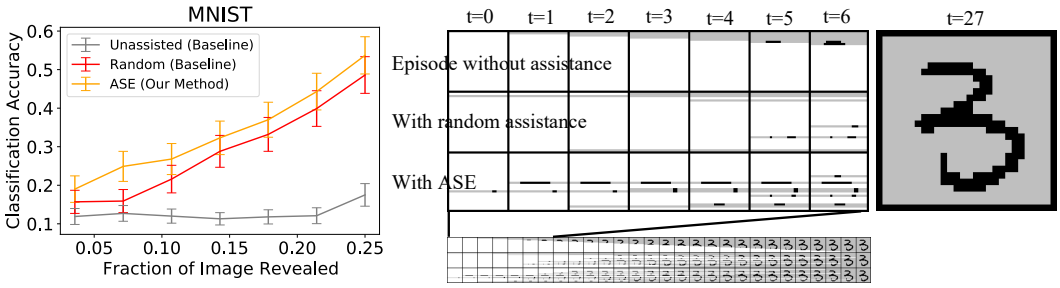


Figure 2: Standard error across 100 episodes. ASE tends to quickly reveal rows near the middle and rows with many non-zero pixels, enabling the user to more accurately guess the label earlier. In the unassisted condition, revealing rows in order from top to bottom is not as quick to reveal informative pixels. The random baseline tends to spread them out uniformly throughout the image, which is a good strategy in the long run but does not necessarily reveal informative pixels early in the episode.

3 User Studies

In our experiments, we evaluate the degree to which our method can provide helpful assistance to users, both in the case where we have prior knowledge of their state estimation process, and where we do not have such knowledge and must instead learn the state estimation model in the loop. We conduct a user study with 12 participants who each perform four tasks: classifying MNIST images under bandwidth constraints (LeCun, 1998), playing the Car Racing video game from the OpenAI Gym with observation delay, navigating a simulated 2D environment with limited vision, and playing the Lunar Lander video game from the OpenAI Gym with limited vision (Brockman et al., 2016). We also conduct experiments with simulated users to study our method under ideal assumptions, and measure the accuracy of the simulated users' internal beliefs (Appendix A.1 contains details).

3.1 Assisting Users with Known Biases

Our first experiment seeks to answer **Q1**: can we assist users when we assume we know their state estimation process? We test this hypothesis on MNIST image classification, and the Car Racing video game from the OpenAI Gym.

MNIST image classification with a bandwidth constraint. In this experiment, we test our method's ability to assist the user when the user cannot leverage their memory of past actions and their knowledge of the state transition dynamics to infer the current state, and must rely entirely on observations. To that end, we formulate a sequential image classification task in which the user's actions have no effect on the state. We take the standard MNIST digit classification problem and intentionally introduce a bandwidth constraint: at each timestep, the user is shown one row of 28 pixels in the 28x28 image, and must try to classify the image given only the pixels observed so far. The assistant observes the full image at the start of the episode, and aims to help the user classify the image as quickly as possible by showing the user informative pixels. We assume that the user's belief update $b_{\mathbf{H}}$ is equivalent to the assistant's belief update b, except that it can only process one row of pixels per timestep. The assistant uses a recurrent neural network state encoder f to compute the belief update b via Equation 2, where f is trained offline to reconstruct the full image given a sequence of pixel observations. We compute the optimal synthetic observation $\tilde{\mathbf{o}}_t$ by simply enumerating all rows of pixels that have not been shown to the user yet, and computing the KL-divergence (Equation 3) for each possible value of $\tilde{\mathbf{o}}_t$. We evaluate (1) an unassisted baseline that reveals the pixel rows in order from top to bottom; (2) a random baseline that reveals a new pixel row sampled uniformly at random; and (3) ASE. Appendix A.2 describes the experimental setup in more detail.

Figure 2 shows that ASE substantially outperforms the unassisted baseline (orange vs. gray curve), and enables the user to classify the digit using fewer timesteps (i.e., fewer pixels) than the random baseline (orange vs. red curve). We ran a one-way repeated measures ANOVA on the classification accuracy dependent measure from the random and ASE conditions, with the presence of ASE as a factor and the digit ID and fraction of image revealed as covariates, and found that $f(1, 5452) = 7.97, p < .01$. While the effect was not substantial – the assisted user's least-squares-mean accuracy was 74.2%, while the unassisted user's was 71.7% – the assisted user achieved significantly higher

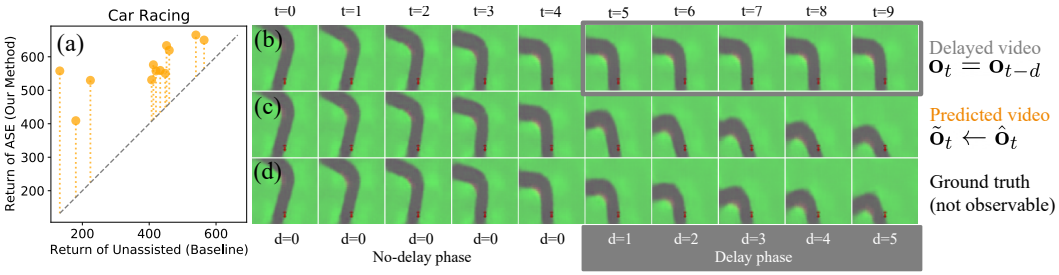


Figure 3: **(a)** Each orange circle represents one of the 12 participants. The dashed gray line shows baseline-equivalent performance, and the dotted orange lines show the difference between assisted and unassisted performance. Per-user return is averaged across 3 episodes (50 seconds each). **(b-d)** Top-down views of approaching a left turn with observation delay d at time t: **(b)** outdated ambient observation $\mathbf{o}_t$, **(c)** forward-predicted observation representative of the current state $\hat{\mathbf{o}}_t$, and **(d)** the ground truth, which cannot be observed by either the user or the assistant. ASE shows the user the forward-predicted observation $\hat{\mathbf{o}}_t$, which is closer to the ground truth than the outdated ambient observation $\mathbf{o}_t$ that the user would see by default, especially when the delay is d is large. Videos are included in the supplementary materials.

accuracy than the unassisted user. Although the uniform-random baseline happens to perform well on MNIST, it performs extremely poorly in simulation experiments with 2D navigation and Car Racing (Table 1 and Figure 8 in the appendix).

Car Racing video game with observation delay. In this experiment, we test our method's ability to assist the user in a real-time driving game with delayed observations, where the user tends to react to outdated observations as if they are current. Our assistant sees the same delayed observations as the user, but instead of passing them to the user, replaces the user's video feed with synthetic images produced by a generative model. To optimize these images to induce the correct state beliefs in the user, the assistant forward-predicts the current state from the delayed observation and the user's most recent actions, then constructs an image observation representative of the predicted current state. By default, this environment emits a 64x64 RGB image observation with a top-down view of the car, and the user can steer left or right using their keyboard (Figure 3). To simulate intermittent observation delays, we set up the environment to alternate between a no-delay phase of emitting new observations immediately (for 5 timesteps) and a delay phase of repeatedly emitting the final observation from the previous no-delay phase (for 5 timesteps). Both the assistant and the user experience the same delay. We assume that the user's belief update $b_{\mathbf{H}}$ is identical to the assistant's belief update b, except that $b_{\mathbf{H}}$ treats observations as if they are never delayed. In practice, although users can clearly tell there is a delay, they are incapable of adjusting to it and steer as if there is no delay. The assistant uses a recurrent neural network (RNN) state encoder f to compute the belief update b via Equation 2, and a variational auto-encoder (VAE; Kingma and Welling, 2013) to synthesize image observations from the hidden states of f. If the last d observations are delayed, a straightforward solution emerges from the assistant's belief-matching objective in Equation 3: replace the delayed observations $\mathbf{o}_{t-d+1:t}$ with recursively predicted, non-delayed observations $\hat{\mathbf{o}}_{t-d+1:t}$ from the RNN encoder f and VAE image decoder, and show the user the prediction of the current observation: $\tilde{\mathbf{o}}_t \leftarrow \hat{\mathbf{o}}_t$. If the last observation $\mathbf{o}_t$ was not delayed, then the assistant simply shows the ambient observation: $\tilde{\mathbf{o}}_t \leftarrow \mathbf{o}_t$. We evaluate (1) an unassisted baseline that passively shows the ambient observation generated by the environment and (2) ASE. Appendix A.2 describes the experimental setup in more detail.

Plot (a) in Figure 3 shows that users are able to achieve substantially larger returns (i.e., drive on the road and stay off the grass more often) with the ASE assistant compared to the unassisted condition. ASE makes the user's video feed smoother by predicting the current observation when the true current observation is delayed, which makes real-time, closed-loop control of the car substantially easier. Users in the unassisted condition tended not to change their steering action when the true images were delayed, while assisted users were able to rapidly switch steering actions even during delay phases, by responding to the assistant's synthetic images. We ran a one-way repeated measures ANOVA on the returns from the unassisted and ASE conditions with the presence of ASE as a factor, and found that $f(1, 11) = 41.01, p < .001$. The assisted user achieved significantly higher returns than the unassisted user. The subjective evaluations in Table 3 in the appendix corroborate these results: users reported perceiving smaller delays and feeling more in control of the car when they were assisted. One reason that users may have perceived a small delay even in the assisted condition

is that the assistant uses an imperfect, learned state encoder f in its belief update. This suggests that even when the assistant has an imperfect state estimation process, ASE can still improve the user's task performance; the assistant's process just has to be more accurate than the user's. Videos in the supplementary material illustrate the difference between unassisted and assisted observations in more detail.

3.2 Learning to Assist Users with Unknown Biases

Our second experiment seeks to answer **Q2**: can we assist users when we do not know their state estimation process, and must learn a model of it? We test this hypothesis first in a 2D navigation task, then in a variant of the Lunar Lander video game from the OpenAI Gym.

2D navigation with incomplete mental map of object locations. In this experiment, we intentionally introduce a bias into the user's perception (unknown to ASE), and test whether ASE can learn a user model that recovers this bias. Inspired by assistive navigation systems like In Situ Audio Services (Panëels et al., 2013), which inform visually-impaired users about nearby points of interest through audio feedback as they walk down a street, we set up a simulated 2D navigation task in which the user cannot directly access their current position and orientation, but can infer them using text observations that describe nearby objects. To incept a controlled user bias, we intentionally do not include the locations of certain objects in the user's 'mental map,' which prevents the user from using observations of those objects to infer their current state as they navigate to a goal. To effectively assist the user, ASE must learn that the user ignores observations that mention these unknown objects. Figure 9 in the appendix illustrates the 'mental map' of the 5x5 grid world shown to the user. At each timestep, the user is told about one of the objects directly in front of them. Some objects are unique, while other objects have multiple instances that exist in different locations (e.g., one computer vs. multiple plants). The objects are divided into 3 categories: (a) unique but unknown, (b) not unique but known, and (c) both unique and known. The assistant knows the locations of all objects, and can observe all objects in front of the user simultaneously. Following Section 2.3, we parameterize the user model b_θ as a Bayesian belief update (Equation 1) that uses observation model $p_\theta^{\text{obs}}(\mathbf{o}|\mathbf{s})$. The parameter $\theta \in [0, 1]$ weights the observation probabilities of objects in category (a): $p_\theta^{\text{obs}}(\mathbf{o}|\mathbf{s}) \propto \theta \cdot p^{\text{obs}}(\mathbf{o}|\mathbf{s})$ for all objects $\mathbf{o}$ in category (a). Because we intentionally make category (a) objects unknown to the user, we know the true value: $\theta = 0$. We would like ASE to learn this value from the user's observed behavior. We evaluate (1) an unassisted baseline that passively shows the ambient observation generated by the environment; (2) a naïve version of ASE that does not train the user model, and instead continues using the initial model b_{init} where $\theta = 1$; and (3) ASE. Appendix A.2 describes the experimental setup in more detail.

Figure 4 shows that users are able to move toward the goal substantially faster with the ASE assistant compared to the unassisted condition. Furthermore, learning a model of the user's observation model substantially improved the assistant's performance compared to the naïve assistant. In the unassisted condition, the user receives many observations of objects in category (b), which are known but relatively uninformative since they have multiple known locations. In the naïve condition, the user receives many observations of objects in categories (a), which are unique but unknown. ASE learns that objects in category (a) are unknown (i.e., $\hat{\theta} = 0$), so it only shows the unique and known objects in category (c). We ran a one-way repeated measures ANOVA on the time-to-goal dependent measure from the unassisted and ASE conditions with the presence of ASE as a factor, and found that $f(1, 11) = 18.02, p < .01$. The assisted user reached the goal significantly faster than the unassisted user. The subjective evaluations in Table 4 in the appendix corroborate these results: users reported finding the observations more helpful in the ASE condition compared to the unassisted condition.

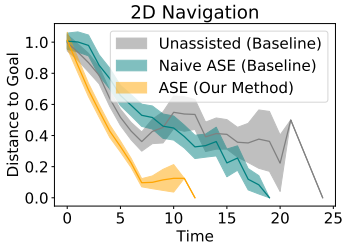


Figure 4: Standard error across 55 episodes (5 episodes per user).

Lunar Lander video game with limited vision. In this experiment, we evaluate whether ASE can learn a personalized model of naturally-occurring user biases in the Lunar Lander game, in which users tend to land at an unsafe angle. We conjecture that this suboptimal user behavior is caused by underestimating the lander's tilt, and that the assistant might learn to help the user by showing them an image in which the lander's tilt is exaggerated beyond the ground truth. At each timestep, the

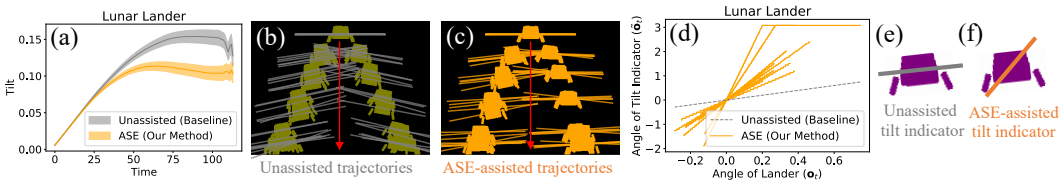


Figure 5: **(a)** Standard error across 120 episodes (10 episodes per user). **(b)** Sample of unassisted trajectories from the user studies. **(c)** With assistance, the user keeps the lander more level. **(d-f)** ASE tends to exaggerate the tilt indicator (orange vs. gray line), and personalizes the exaggeration to the user (each orange line in **(d)** corresponds to a different user). Videos are included in the supplementary materials.

environment emits an image of the lander, and the user can fire the left or right thruster using their keyboard (plot (b) in Figure 5). The objective is to make sure the lander stays level as it descends, using the thrusters to prevent it from tilting left or right. The image includes a visual indicator of the lander's tilt, which is separate from the body of the lander (plot (e) in Figure 5). The assistant is capable of freely changing the angle of this tilt indicator in the image observation shown to the user, but cannot change any other aspect of the image (e.g., the lander body itself). To simplify our model of the user, we focus on one feature: the angle of the lander. In the user model, an observation is characterized by the angle of the tilt indicator: the observation space is $\Omega = [-\pi, \pi]$. A state is characterized by the lander's angle: the state space is $\mathcal{S} = [-\pi, \pi]$. By default, the angle of the tilt indicator is equal to the lander's angle: $p^{\text{obs}}(\mathbf{o}|\mathbf{s}) = \mathbb{1}[\mathbf{o} = \mathbf{s}]$. We assume the user's suboptimality stems from incorrectly inferring the lander's tilt from the angle of the tilt indicator: it is easy to tell when the lander is severely tilted, but harder to tell when the lander is only slightly tilted. This is a problem for the user, since keeping the lander level requires detecting tilt early when it is still small, so that the thrusters have enough time to force the lander upright. We represent the user model b_θ using a simple logistic model: $b_\theta(\mathbf{o}_{0:t}, \mathbf{a}_{0:t-1}) = -\pi + 2\pi \cdot \sigma(\theta_0 + \theta_1 \cdot \mathbf{o}_t)$, where σ is the sigmoid function. The optimal synthetic observation anticipates the user's internal distortion: $\tilde{\mathbf{o}}_t \leftarrow b_{\hat{\theta}}^{-1}(\mathbf{o}_t)$. We evaluate (1) an unassisted baseline that rotates the tilt indicator to exactly match the lander's angle and (2) ASE. Appendix A.2 describes the task in more detail.

Plot (a) in Figure 5 shows that users are able to substantially decrease the tilt of the lander throughout the episode with the ASE assistant compared to the unassisted condition. The assistant learns that the user infers a smaller lander angle than the observed tilt indicator's angle. This leads to an assistance policy that exaggerates observations by rotating the tilt indicator to exceed the lander's true angle. Furthermore, plot (d) in Figure 5 shows that the learned distortion model varies across users. We ran a one-way repeated measures ANOVA on the average tilt dependent measure from the unassisted and ASE conditions with the presence of ASE as a factor, and found that $f(1, 11) = 6.30, p < .05$. The assisted user's average tilt was significantly smaller than the unassisted user's. The subjective evaluations in Table 2 in the appendix corroborate these results: users reported finding it easier to tell when the lander was tilted in the ASE condition compared to the unassisted condition.

4 Limitations and Future Work

Our method assumes that we can solve for near-optimal state estimators b and policies π, which may not be feasible in real-world domains. Fortunately, recent work has demonstrated substantial improvements in learning approximate state estimators and policies in complex environments with image observations, unknown dynamics, and other challenges (Zhang et al., 2019, Hafner et al., 2019). While our proof of concept does not make use of these advances, incorporating them into a more practical assistive state estimation system is a promising direction for future work. Furthermore, our user studies are limited in that we do not know if the improvement in users' task performance is caused by more accurate internal beliefs, or by some other feature of the assistance condition, since we cannot directly measure those beliefs. Our ultimate goal is for the user to make better decisions when assisted, and the results show that the belief-matching objective in Equation 3 accomplishes this in four different domains. Even so, one direction for future work is to design experiments that more directly measure the user's internal beliefs during decision-making.

References

D. C. Knill and W. Richards. *Perception as Bayesian inference*. 1996.

M. Mulder. *Cybernetics of tunnel-in-the-sky displays*. PhD thesis, Delft University of Technology, 1999.

S. A. Panëels, A. Olmos, J. R. Blum, and J. R. Cooperstock. Listen to it yourself! evaluating usability of what's around me? for the blind. In *SIGCHI Conference on Human Factors in Computing Systems*, 2013.

T. Fong, J. Rochlis Zumbado, N. Currie, A. Mishkin, and D. L. Akin. Space telerobotics: unique challenges to human-robot collaboration in space. *Reviews of Human Factors and Ergonomics*, 2013.

Y. Zhao, S. Szpiro, L. Shi, and S. Azenkot. Designing and evaluating a customizable head-mounted vision enhancement system for people with low vision. *ACM Transactions on Accessible Computing (TACCESS)*, 2019.

S. Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 1998.

S. Thrun, W. Burgard, and D. Fox. Probalistic robotics. 2005.

F. Schmitt, H.-J. Bieg, M. Herman, and C. A. Rothkopf. I see what you see: Inferring sensor and policy models of human real-world motor behavior. In *AAAI Conference on Artificial Intelligence*, 2017.

S. Daptardar, P. Schrater, and X. Pitkow. Inverse rational control with partially observable continuous nonlinear dynamics. *arXiv preprint arXiv:1908.04696*, 2019.

D. Jarrett and M. van der Schaar. Inverse active sensing: Modeling and understanding timely decision-making. *arXiv preprint arXiv:2006.14141*, 2020.

A. Y. Ng and S. J. Russell. Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning*, 2000.

C. L. Baker, R. Saxe, and J. B. Tenenbaum. Action understanding as inverse planning. *Cognition*, 2009.

M. Bühler and T. Weisswange. Theory of mind based communication for human agent cooperation. 2020.

S. Hilgard, N. Rosenfeld, M. R. Banaji, J. Cao, and D. C. Parkes. Learning representations by humans, for humans. *arXiv preprint arXiv:1905.12686*, 2019.

E. Ohn-Bar, K. Kitani, and C. Asakawa. Personalized dynamics models for adaptive assistive navigation systems. *arXiv preprint arXiv:1804.04118*, 2018.

Y. LeCun. The MNIST database of handwritten digits, 1998. URL http://yann.lecun.com/exdb/mnist/.

G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

M. Zhang, S. Vikram, L. Smith, P. Abbeel, M. Johnson, and S. Levine. SOLAR: Deep structured representations for model-based reinforcement learning. In *International Conference on Machine Learning*, 2019.

D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *International Conference on 3D Vision*, 2017.

Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra. Habitat: A Platform for Embodied AI Research. In *IEEE/CVF International Conference on Computer Vision*, 2019.

C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 1992.

M. Bloem and N. Bambos. Infinite time horizon maximum causal entropy inverse reinforcement learning. In *IEEE Conference on Decision and Control*, 2014.

D. Ha and J. Schmidhuber. Recurrent world models facilitate policy evolution. In *Neural Information Processing Systems*, 2018.

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

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

J. Schmidhuber. Making the world differentiable: On using self-supervised fully recurrent neural networks for dynamic reinforcement learning and planning in non-stationary environments. 1990.

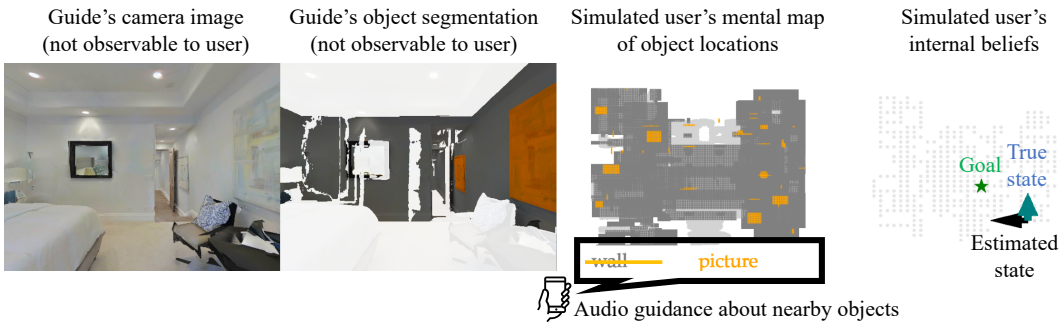


Figure 6: A simulated blind user navigating an indoor environment from the Matterport3D dataset (Chang et al., 2017), using audio guidance about nearby objects to estimate position and orientation (the same problem setting as Figure 1). The assistant sees the RGB camera image, uses the semantic mesh from the dataset to determine the list of visible objects, then replaces the ambient observation (gray), which was sampled uniformly at random from the list of visible objects, with an optimized observation (orange) that minimizes KL-divergence (Equation 3). The simulated user knows the locations of all objects, and can use this mental map to infer their current position and orientation given observations of nearby objects and memory of past movements.

A Appendix

A.1 Simulation Experiments

One of the drawbacks of running a user study with human participants is that, while we can measure task performance, we cannot directly measure the accuracy of users' internal beliefs about the current state. Studying how ASE scales with the amount of observation delay, training data, and other factors would also require a prohibitive number of human-in-the-loop experiments. To that end, we run experiments with simulated users on indoor navigation and MNIST digit classification.

A.1.1 Improving Accuracy of Users' Internal Beliefs

Our third experiment seeks to answer **Q3**: can we improve the accuracy of simulated users' internal beliefs? We test this hypothesis in an indoor navigation task with a more realistic environment than the 5x5 layout from the user study: we take one floor of a 3D house from the Matterport3D dataset (Chang et al., 2017), and discretize it into a navigable 2D grid using the Habitat framework (Manolis Savva* et al., 2019). We simulate the user using a goal-conditioned policy $\pi(a|s;g) \propto \exp\left(Q(s,a;g)\right)$ that takes the shortest path to the goal, where the value function Q is computed using tabular soft Q-iteration (Watkins and Dayan, 1992, Bloem and Bambos, 2014) with a reward function that gives a constant negative penalty for each state transition that does not reach the goal. The simulated user's belief update $b_{\mathbf{H}}$ is identical to the assistant's belief update b, except that it ignores any observation that consists of more than one object. The assistant knows the simulated user's belief update $b_{\mathbf{H}}$. Note that this differs from the 5x5 experiment in Section 3.2, in which the user's belief update was not only bandwidth-constrained, but also tainted by a misspecified observation model due to the presence of unknown objects whose locations were not plotted in the user's mental map. The purpose of this experiment is not to test if ASE can learn a user model, but rather to test the accuracy of the user's induced beliefs. Appendix A.2 describes the experimental setup in more detail.

Manipulated factors. We evaluate (1) an unassisted baseline that passively shows the ambient observation generated by the environment and (2) ASE.

Dependent measures. We measure success rate in reaching the goal, distance to the goal at the end of the episode, number of steps taken to reach the goal, and the simulated user's internal log-likelihood of the true state (averaged across timesteps).

Analysis. Table 1 show that ASE substantially outperforms the unassisted and random baselines in assisting simulated users: not only in terms of task performance, but also the accuracy of the users' internal beliefs. ASE tends to inform the user of landmark objects that are more likely to be seen from the current state than in other states – like gym equipment, paintings, and showers – enabling the user to infer the current state more accurately. In the unassisted condition, the user tends to receive observations of objects that are common not only in the current state but also common across states –

	Success Rate	Distance to Goal	Time to Goal	Belief in True State
Unassisted (Baseline)	0.73 ± 0.04	0.10 ± 0.02	70.87 ± 2.72	-1.70 ± 0.02
Random (Baseline)	0.02 ± 0.01	1.00 ± 0.04	99.99 ± 0.71	-20.44 ± 0.08
ASE (Our Method)	1.00 ± 0.00	0.00 ± 0.00	38.55 ± 1.60	-0.72 ± 0.02

Table 1: Habitat navigation experiments that address **Q3** – can we improve the accuracy of simulated users' internal beliefs? – by comparing our method (ASE), which synthesizes an informative observation that fits within the simulated user's sensor bandwidth, to baselines that either use ambient observations generated by the environment (Unassisted) or randomly generate observations (Random). The results show that our method (ASE) substantially outperforms the baselines (Unassisted and Random). The simulated user's internal beliefs are represented as log-likelihoods. We measure performance and standard error using 100 evaluation episodes.

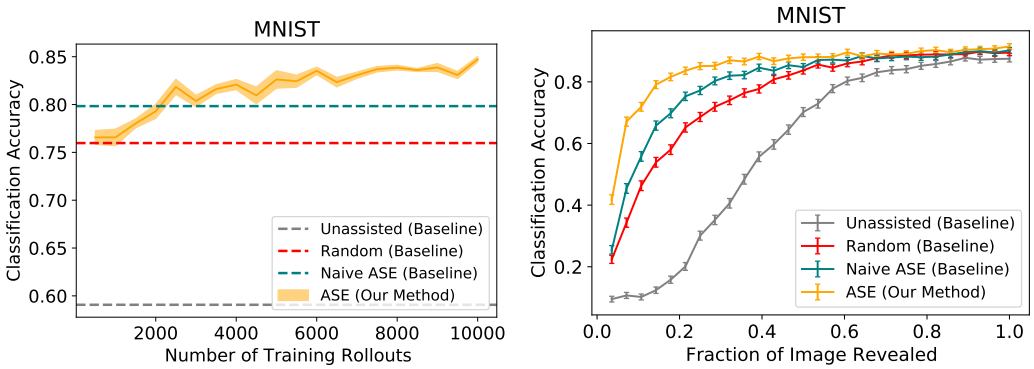


Figure 7: MNIST experiments that address **Q4** – given enough demonstration data, can ASE learn complex models of the user's state estimation process? – by comparing our method (ASE), which learns a model of the simulated user, to a baseline variant of our method that does not learn a model (Naïve ASE). The results show that with enough training data, the personalized assistant outperforms the naïve assistant by more accurately predicting the effect of a given observation on the simulated user, and thus providing more informative observations to the simulated user. We measure performance and standard error across 5 random seeds and 1000 evaluation episodes.

like walls, floors, and ceilings – which makes it difficult to uniquely identify the current state. This result illustrates that ASE can be used to improve situational awareness, independent of the user's desired task.

A.1.2 Scaling to Multivariate User Models

Our fifth experiment seeks to answer **Q4**: given enough demonstration data, can ASE learn complex models of the user's state estimation process? We test this hypothesis in the MNIST domain from Section 3.1. We simulate a user by training an LSTM sequence model to predict the image label given a sequence of pixel observations. We define the simulated user's belief update $b_{\mathbf{H}}$ using Equation 2, where the state encoder $f_{\mathbf{H}}$ maps a sequence of observations to a 32-dimensional hidden state. This hidden state is distinct from the hidden state produced by the assistant's state encoder f, due to the difference between the assistant's reconstruction objective (described in Section 3.1) and the simulated user's classification objective. ASE represents the user's state encoder as a recurrent neural network f_θ with 32 hidden units, and defines the user model b_θ via Equation 2. Appendix A.2 describes the experimental setup in more detail.

Manipulated factors. We evaluate (1) an unassisted baseline that passively shows the ambient observation generated by the environment; (2) a naïve version of ASE that does not train the user model; and (3) ASE, where we learn θ from episodes generated iteratively using Algorithm 1. The naïve assistant incorrectly assumes the user's state encoder $f_{\mathbf{H}}$ is equivalent to the assistant's state encoder f, except that it can only process one row of pixels per timestep. We vary the number of training episodes $|\mathcal{D}|$ in the ASE condition.

Dependent measures. We measure per-timestep classification accuracy, as in Section 3.1.

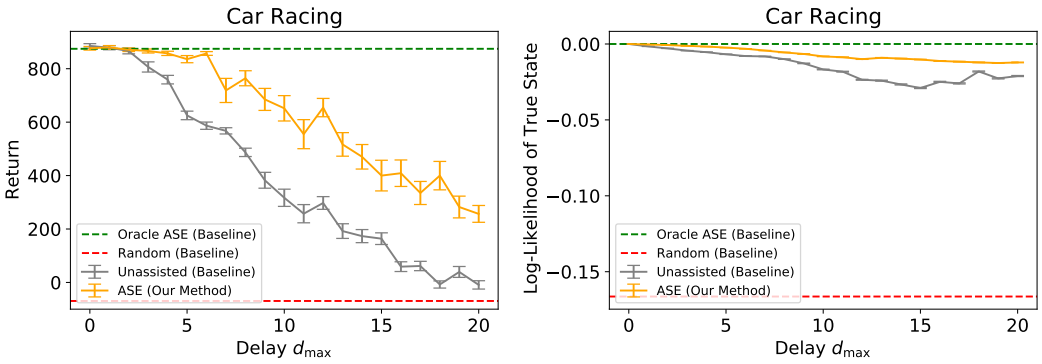


Figure 8: Car Racing experiments that address **Q5** – does ASE still improve the user's performance when observations are severely delayed? – by comparing our method (ASE), which tries to 'undo' the observation delay $d_{\max}$ by predicting the current state and showing the user an observation representative of the predicted current state, to baselines that either show the human the outdated ambient observation generated by the environment (Unassisted) or randomly generate observations (Random). The results show that ASE substantially improves the simulated user's task performance (left plot) and the simulated user's internal state estimation accuracy (right plot), especially when the delay $d_{\max}$ is high. We measure performance and standard error using 20 evaluation episodes. The gap between ASE and the oracle can be attributed to imperfections in the assistant's learned dynamics model, which is used to define its state encoder f.

Analysis. Figures 7 shows that with enough training episodes in $\mathcal{D}$, ASE can learn a model of the simulated user that enables it to outperform a naïve version of ASE that assumes the simulated user's belief update uses the same state encoder as the assistant's. This result demonstrates that ASE can scale to training an expressive, recurrent neural network model of the user's belief update b_θ.

A.1.3 Scaling to Longer Observation Delays

Our fifth experiment seeks to answer **Q5**: does ASE still improve the user's performance when observations are severely delayed? We test this hypothesis with simulated users in the Car Racing domain from Section 3.1. We simulate the user using an expert policy trained via the model-based reinforcement learning method described in Ha and Schmidhuber (2018). In addition to manipulating the assistance condition as in Section 3.1, we also manipulate the observation delay $d_{\max} \in \{0, 1, 2, ..., 20\}$. The delay $d_{\max}$ controls the length of the no-delay and delay phases. For example, in the user study in Section 3.1, the no-delay and delay phases each lasted 5 timesteps, which corresponds to $d_{\max} = 5$. In addition to measuring the task return, we also measure the simulated user's internal log-likelihood of the true state at each timestep.

Figure 8 shows that ASE substantially outperforms the unassisted and random baselines (orange vs. gray and red curves) in assisting simulated users. ASE helps the simulated user by predicting the current state given outdated observations, then showing the user an observation representative of the predicted current state. These predictions are not perfect, as shown by the gap between ASE and the oracle (orange vs. green curve), but still align the simulated user's beliefs more closely with the true state. As the delay $d_{\max}$ increases, the assistant's dynamics model – which is used to define its state encoder f (see Section 3.1) – is not able to accurately predict the current state. Hence, both assisted and unassisted performance decrease as the delay increases.

A.2 Implementation Details

Maximum-likelihood estimation. We use Adam (Kingma and Ba, 2014) to perform gradient descent on the objective in Equation 5.

MNIST. There are $T = 28$ timesteps per episode, and the user can change their label at each timestep. Each observation consists of zero or more row indices and corresponding pixel values: $\Omega = ([28] \times \mathbb{R}^{28})^*$, where $[28] = \{1, 2, ..., 28\}$ denotes the set of row indices, and $*$ denotes the Kleene star. Let $\mathbf{I}_{1:28,1:28} \in \mathbb{R}^{28\times28}$ denote the full image. The environment initially emits the full image observation $\mathbf{o}_0 = ((1, \mathbf{I}_{1,1}, \mathbf{I}_{1,2}, ..., \mathbf{I}_{1,28}), (2, \mathbf{I}_{2,1}, ...), ..., (28, \mathbf{I}_{28,1}, ...))$, and the assistant observes it. We define the state encoder f used in the assistant's belief update b (Equation 2) by

training an LSTM sequence model (Hochreiter and Schmidhuber, 1997) to reconstruct the full image given a sequence of pixel observations. To model the user's limited sensor bandwidth, we assume the user's state estimation process lies in a singleton hypothesis space $\mathcal{B} = \{b_{\mathbf{H}}\}$, where $b_{\mathbf{H}}(s_t; \mathbf{o}_{0:t})$ is identical to the assistant's belief update b except that it ignores any observation that consists of more than one row of pixels. Under these assumptions, the optimal synthetic observation $\tilde{\mathbf{o}}_t$ (Equation 3) consists of exactly one row of pixels. Furthermore, $\tilde{\mathbf{o}}_t$ minimizes the Euclidean distance between the user's latent state $f(\tilde{\mathbf{o}}_{0:t-1}, \tilde{\mathbf{o}}_t)$ after observing the partial image and the assistant's latent state $f(\mathbf{o}_0)$ after observing the full image (we take $\sigma^2 \to 0$ in the assistant's belief update in Equation 2 in order to simplify the KL-divergence to the Euclidean distance between mean states). We compute $\tilde{\mathbf{o}}_t$ by simply enumerating all rows of pixels that have not been shown to the user yet, and computing the KL-divergence (Equation 3) for each possible value of $\tilde{\mathbf{o}}_t$.

We measure the user's classification accuracy at each timestep, in order to capture how quickly the user recognizes the image over the course of an episode. We recruited 11 male and 1 female participants, with an average age of 25. Each participant was provided with the rules of the task and example images, then labeled 25 different digits. Each digit was broken down into an episode of 28 partial images, yielding a total of 700 labels per user. To avoid the confounding effect of users learning to classify images more accurately and quickly over time, we randomly interleave episodes from each of the three conditions. For example, episode 1 is unassisted, episode 2 is assisted by ASE, episode 3 is assisted by the random baseline, etc.

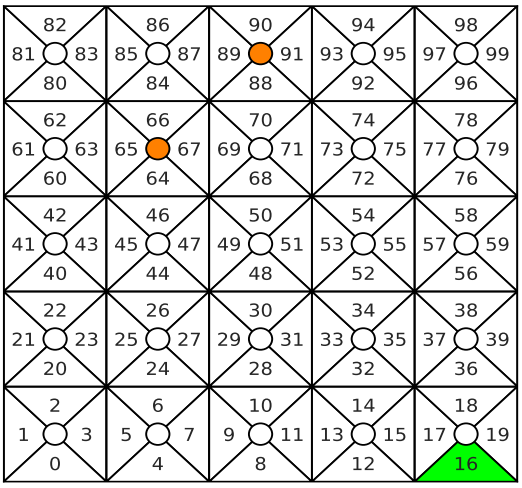
Car Racing. Each observation consists of an image and a binary feature that indicates whether the image is delayed: $\Omega = \mathbb{R}^{64x64} \times \{0, 1\}$, where 0 indicates no delay and 1 indicates delay. We define the state encoder f used in the assistant's belief update b (Equation 2) and a state-to-image decoder g by training a recurrent dynamics model (Schmidhuber, 1990) and variational auto-encoder (VAE; Kingma and Welling, 2013) on random trajectories (Ha and Schmidhuber, 2018). Hence, the state space $\mathcal{S}$ is the 256-dimensional latent space of the recurrent neural network f. If the last $d > 0$ observations are delayed, our recurrent state encoder $f(\mathbf{o}_{0:t}, \mathbf{a}_{0:t-1})$ replaces the delayed observations $\mathbf{o}_{t-d+1:t}$ with recursively predicted, non-delayed observations $\hat{\mathbf{o}}_{i>t-d} = (g(f(\mathbf{o}_{0:t-d}, \hat{\mathbf{o}}_{t-d+1:i-1}, a_{0:i-1})), 0)$, where the 0 indicates that the predicted observation is not delayed. We assume the user's state estimation process lies in a singleton hypothesis space $\mathcal{B} = \{b_{\mathbf{H}}\}$, where $b_{\mathbf{H}}(s_t; \mathbf{o}_{0:t}, \mathbf{a}_{0:t-1})$ is identical to the assistant's belief update b except that it ignores the binary delay indicator in the observations and simply assumes all observations are not delayed. If the last $d > 0$ observations are delayed, the optimal synthetic observation $\tilde{\mathbf{o}}_t$ (Equation 3) minimizes the Euclidean distance between the user's latent state $f(\tilde{\mathbf{o}}_{0:t-1}, \tilde{\mathbf{o}}_t, \mathbf{a}_{0:t-1})$ after observing the synthetic images and the assistant's latent state $f(\mathbf{o}_{0:t-d}, \hat{\mathbf{o}}_{t-d+1:t}, \mathbf{a}_{0:t-1})$ after undoing the delay d (as in MNIST, we take $\sigma^2 \to 0$ in the assistant's belief update in Equation 2 in order to simplify the KL-divergence in Equation 3 to the Euclidean distance between mean states). We approximate this optimal synthetic observation using the forward-predicted observation: $\tilde{\mathbf{o}}_t \leftarrow \hat{\mathbf{o}}_t$. If the last observation $\mathbf{o}_t$ was not delayed, then we simply show the ambient observation: $\tilde{\mathbf{o}}_t \leftarrow \mathbf{o}_t$.

We measure performance using a reward function that penalizes going off road and gives bonuses for visiting new patches of the road. We recruited 11 male and 1 female participants, with an average age of 25. Each participant was provided with the rules of the task and a short practice period of 2 episodes to familiarize themselves with the controls and dynamics. Each user played in both conditions: unassisted, and assisted by ASE. To avoid the confounding effect of users learning to play the game better over time, we counterbalanced the order of the two conditions. Each condition lasted 3 episodes, with 1000 timesteps (50 seconds) per episode.

2D navigation. The states are arranged in a 5x5 grid, and the number of states is $|\mathcal{S}| = 100$. Each state (x, y, ϕ) contains a discrete position $x, y \in \mathbb{N}$ and discrete orientation $\phi \in \{\mathrm{N}, \mathrm{S}, \mathrm{E}, \mathrm{W}\}$. The actions, $\mathcal{A} = \{\text{turn left}, \text{turn right}, \text{move forward}\}$, change the user's orientation or position deterministically. The environment contains a discrete set of 78 objects (26 in each of the 3 categories): $\mathcal{X} = \{\text{chair, window, bathtub, painting, ...}\}$. The observation space is the power set of the set of objects: $\Omega = \mathcal{P}(\mathcal{X})$. The observation model $p_{\text{obs}}(\mathbf{o}|\mathbf{s})$ is a delta function on the subset of all objects $\mathbf{o}$ visible from state $\mathbf{s}$. We compute the optimal synthetic observation $\tilde{\mathbf{o}}_t$ by simply enumerating the singleton sets of objects in Ω and computing the KL-divergence (Equation 3) for each possible value of $\tilde{\mathbf{o}}_t$.

In ASE, we learn θ from the episodes collected in conditions 1 and 2. In practice, we pooled the data from the first k participants to train the model for the k-th participant,

since we do not expect the model to vary between users, and because the small amount of data collected for each individual user was too noisy to learn an accurate model from. We measure the distance from the user's current position to their goal position (normalized by distance from initial position to goal position) at each timestep, in order to capture how quickly the user moves toward the goal throughout the episode. We recruited 11 male and 1 female participants, with an average age of 25. Each participant was provided with the rules of the task and a short practice period of 3 episodes to familiarize themselves with the controls and dynamics. Each user played in all three conditions: unassisted, assisted by naïve ASE, and assisted by ASE. We counterbalanced the order of the unassisted and naïve ASE conditions. We could not counterbalance the order of the ASE condition to control for the learning effect, since ASE learns $\hat{\theta}$ from the data collected in the unassisted and naïve ASE conditions. Figure 10 shows that the introduction of the ASE assistant sharply improves the user's performance across episodes, suggesting the learning effect was not a substantial confounder. Each condition lasted 5 episodes, with 25 timesteps per episode.



Time: 0 of 25 steps
Goal: 16
Object guidance: there is a computer directly in front of you.
There are computers located at the highlighted orange circles.
What are your possible current positions and orientations?:
> 63 84 69 46 87 93 70
What action would you like to take (a/w/s/d)?:
> d

Figure 9: The user's console interface for 2D navigation.

Lunar Lander. In the experiments, we intentionally blend the color of the lander's body with the background to make it difficult for the user to see, making the tilt indicator more prominent. We define the state encoder f used in the assistant's belief update b (Equation 2) to simply pass through the most recent observation: $f(\mathbf{o}_{0:t}, \mathbf{a}_{0:t-1}) = \mathbf{o}_t$.

In ASE, we learn θ from the episodes collected in the unassisted condition, as well as 5 assisted episodes generated iteratively using Algorithm 1. We measure the absolute value of the lander's angle $|\mathbf{s}_t|$ (i.e., 'tilt') at each timestep, in order to capture how well the user is able to stabilize the lander throughout the episode. We recruited 11 male and 1 female participants, with an average age of 25. Each participant was provided with the rules of the task and a short practice period of 5 episodes to familiarize themselves with the controls and dynamics. Each user played in both conditions: unassisted, then assisted by ASE. We could not counterbalance the order of the two conditions to control for the learning effect, since ASE learns $\hat{\theta}$ from the data collected in the unassisted condition. Figure 10 shows that the introduction of the ASE assistant sharply improves the user's performance across episodes, suggesting the learning effect was not a substantial confounder. Each condition lasted 10 episodes, with 150 timesteps (10 seconds) per episode.

Habitat navigation. The number of states is $|\mathcal{S}| = 1640$. The number of objects is $|\mathcal{X}| = 34$. The initial state distribution is uniform: $s_0 \sim \text{Unif}(\mathcal{S})$. At the beginning of the episode, the user has a uniform belief distribution over possible initial states. For each episode, we sample a goal state uniformly at random from $\mathcal{S}$. There are a maximum of $T = 100$ timesteps per episode.

Table 2: Lunar Lander User Study

	p-value	Unassisted	ASE
I could tell when the lander was tilted	$< \mathbf{.05}$	5.67	**6.33**
I was able to straighten the lander before it tilted out of control	$> .05$	4.42	4.58

Subjective evaluations of the Lunar Lander user study from 12 participants. Means reported below for responses on a 7-point Likert scale, where 1 = Strongly Disagree, 4 = Neither Disagree nor Agree, and 7 = Strongly Agree. p-values from a one-way repeated measures ANOVA with the presence of assistance as a factor influencing responses.

Table 3: Car Racing User Study

	p-value	Unassisted	ASE
I was able to keep the car on the road	$< \mathbf{.0001}$	1.67	**3.75**
I could anticipate the consequences of my steering actions	$< \mathbf{.001}$	2.25	**4.17**
I could tell when the car was about to go off road	$< \mathbf{.01}$	3.08	**4.33**
I could tell when I needed to steer to keep the car on the road	$< \mathbf{.05}$	3.17	**4.83**
I was often able to determine the car's current position using the picture on the screen	$< \mathbf{.05}$	3.50	**4.75**
I could tell that the picture on the screen was sometimes delayed	$< \mathbf{.001}$	6.83	**4.25**
The delay made it harder to perform the task	$< \mathbf{.01}$	6.58	**4.83**

Subjective evaluations of the Car Racing user study from 12 participants. Means reported below for responses on a 7-point Likert scale, where 1 = Strongly Disagree, 4 = Neither Disagree nor Agree, and 7 = Strongly Agree. p-values from a one-way repeated measures ANOVA with the presence of assistance as a factor influencing responses.

Table 4: 2D Navigation User Study

		p-value	Unassisted	Assisted
Naive ASE	I was often able to infer my current position and orientation	$> .05$	5.50	5.67
	I was often able to move toward the goal	$> .05$	5.50	5.58
	I often found the guidance helpful	$> .05$	6.00	5.50
	I often forgot which position and orientation I believed was in	$> .05$	3.17	3.25
ASE	I was often able to infer my current position and orientation	$< \mathbf{.01}$	5.50	**6.83**
	I was often able to move toward the goal	$< \mathbf{.01}$	5.50	**6.83**
	I often found the guidance helpful	$< \mathbf{.01}$	6.00	**6.92**
	I often forgot which position and orientation I believed was in	$< \mathbf{.01}$	3.17	**1.42**

Subjective evaluations of the 2D navigation user study from 12 participants. Means reported below for responses on a 7-point Likert scale, where 1 = Strongly Disagree, 4 = Neither Disagree nor Agree, and 7 = Strongly Agree. p-values from a one-way repeated measures ANOVA with the presence of assistance as a factor influencing responses.

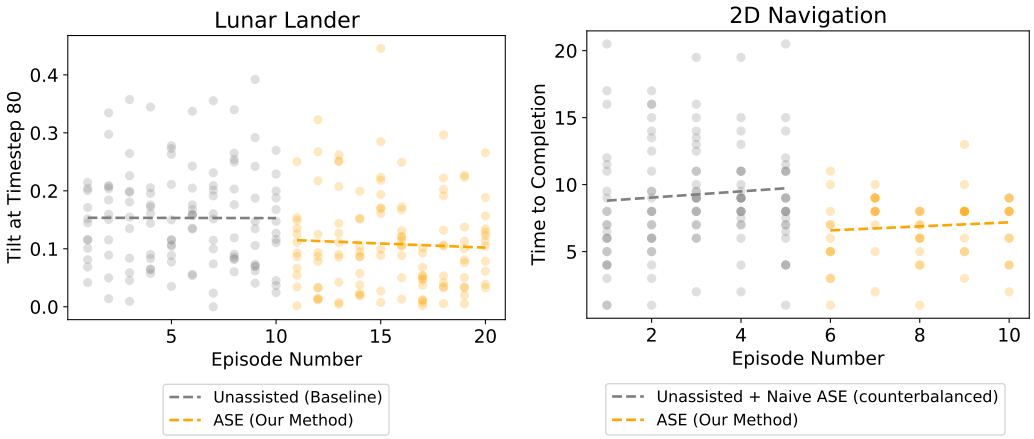


Figure 10: For a given episode number, each circle represents a different user. Each dashed line shows an ordinary least squares regression model trained on the data from a particular phase. Though we did not counterbalance the unassisted and ASE phases (only the unassisted and naïve ASE phases), the learning effect does not appear to be a substantial confounder. Performance is relatively constant during the unassisted phase, and sharply improves once the ASE phase begins. This suggests that the improvement in performance between the unassisted and ASE phases is primarily due to the introduction of the ASE assistant, rather than a learning effect. We plot the tilt at timestep 80 for Lunar Lander, since that is when the performance improvements from assistance tend to appear (see plot (a) in Figure 5).