

Maintaining Awareness of the Focus of Attention of a Conversation: A Robot-Centric Reinforcement Learning Approach

Marynel Vázquez¹, Aaron Steinfeld¹ and Scott E. Hudson^{2,3}

Abstract—We explore online reinforcement learning techniques to find good policies to control the orientation of a mobile robot during social group conversations. In this scenario, we assume that the correct behavior for the robot should convey attentiveness to the focus of attention of the conversation. Thus, the robot should turn towards the speaker. Our results from tests in a simulated environment show that a new state representation that we designed for this problem can be used to find good policies for the robot. These policies can generalize across interactions with different numbers of people and can handle various levels of sensing noise.

I. INTRODUCTION

Mobile social robots are being designed to operate in human environments, with and around groups of people. For example, Cobots [1] navigate university buildings and perform tasks in collaboration with nearby users. Frog [2] has operated as a museum tour guide, often guiding groups of visitors from one place to another. Even though these and other projects within the robotics community have improved robot capabilities in human environments, autonomous robot motion during group conversations has been understudied. Most efforts to control the spatial behavior of robots in these situations have relied on tele-operation [3], [4], [5]. A few exceptions are rule-based approaches [6], which tend to generalize poorly to new situations, and generative models of proxemic behavior [7]. The latter models worked for adapting a robot’s position with respect to a single user, but have not been tested with more people.

Inspired by the success of reinforcement learning (RL) in robotics [8], we explore RL techniques to find good policies to control the orientation of a robot during social group conversations. Body orientation is important because it is often considered as communicative and meaningful by users [9]. For robots with a small number of degrees of freedom, like Cobot [1] or Frog [2], body orientation controls the direction of important social features, such as their faces, as well as the directions of many of their sensors. Thus, the robots’ orientation can significantly affect users’ interpretations of their actions and their sensing capabilities. Furthermore, body orientation can be used to induce spatial reconfigurations during interactions [10]. This is a subtle and effective strategy for redirecting the focus of a conversation.

We approach the problem of controlling the orientation of a robot during group conversations using the Oz of Wizard

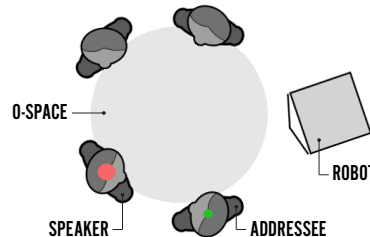


Fig. 1. Simulated group conversation between a robot and four people. The red and green circles on top of the agents identify the speaker and addressees, respectively. The big gray circle represents the o-space of the group’s F-formation.

methodology [11]. Our efforts are focused on evaluating RL approaches in simulated group conversations (Fig. 1) as a precursor to future work with real users. The simulation offers the opportunity to systematically study the effect of sensing noise on the performance of the robot as well as the generalization of learned policies to other group interactions. This is a first step towards automatically optimizing robots’ spatial behavior during situated conversations with users and reducing the amount of engineering required for this task.

One of our main contributions is a robot-centric state representation for reinforcement learning that is agnostic to the number of people in the conversation. This means that the same representation can be used to control the robot while it interacts with 2, 3, 4 or more people and that the learned policies can easily generalize across these scenarios.

Even though robots may take a turn to speak during conversations, we concentrate our evaluation on situations in which the users are the active speakers and the focus of attention. These situations are more interesting to study than their counterparts because the robot does not have control of the interaction dynamics in them and, thus, must adapt to the flow of the conversation. Under these circumstances, we assume that the correct behavior for the robot is to turn towards the speaker to convey *attentiveness* to this person and maintain awareness of the focus of attention.

The rest of this paper is organized as follows. Section II describes our general approach to control the orientation of a robot with reinforcement learning. Section III presents background models from social psychology that informed the design of our simulation environment and details the interaction dynamics and sensing mechanisms that we modeled for this work. Section V and VI then describe our empirical evaluation and results. Finally, Section VII summarizes our results and future research directions.

¹Robotics Institute, Carnegie Mellon University, Pittsburgh, PA. {marynel, steinfeld}@cmu.edu

²Human-Computer Interaction Institute, Carnegie Mellon University, Pittsburgh, PA. scott.e.hudson@cs.cmu.edu

³Disney Research, Pittsburgh, PA.

II. GENERAL APPROACH

We model our motion control problem as a sequential decision-making process. At any time-step t , the robot (or agent) receives some representation of the environment state s_t and executes an action a_t . Executing this action triggers a transition to a following state, represented by s_{t+1} , and results in an immediate reward r_{t+1} . The goal of the robot is to choose actions that maximize the discounted total reward that it receives while it interacts with the world. That is, maximize $\sum_{t=0}^{\infty} \lambda^t r_{t+1}$ with $\lambda \in [0, 1]$ a discount rate.

Out of the many RL techniques that exist, we focus on evaluating popular online approaches that estimate an *action-value function* $Q(s, a)$ to try to find solutions to the motion control problem. The function

$$Q^\pi(s, a) = E_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \mid s_0 = s, a_0 = a \right]$$

is an estimate of how good it is to choose action a in a given state s and then follow policy π . Readers interested in more details about this function are encouraged to refer to [12].

Online approaches are advantageous for our task because they improve as the robot interacts with people and they can adapt quickly to specific interaction dynamics. The latter is particularly advantageous when the social context changes, e.g., when some of the members of the conversation leave or others join the interaction.

III. GROUP SIMULATION

We developed a simulated environment to test control policies for a robot. The simulation was inspired by models from social psychology that explain human spatial behavior during free-standing group conversations [13], [14].

A. Background

When people stand freely in open, public spaces, they tend to maintain distinct spatial organizations known as face formations, or *F-formations* in short. F-formations maximize the opportunities of the interactants to monitor one another during a conversation and maintains their group as a spatially distinct unit from other nearby interactions.

Face formations begin when the members of a group position themselves such that their *transactional segments* intersect (Fig. 2a). These segments extend in front of each person and encompass the physical space that they are using for their current activity.

For group conversations, the intersection of the transactional segments is known as the *o-space* of the corresponding F-formation. In the case of pairs, the o-space is between the members of a face-to-face arrangement (as in Fig. 2a) or in front of them in a side-by-side or “L” arrangement (Fig. 2b and 2c). This pattern can also be observed for bigger groups, which tend to form semi-circular or circular arrangements (Fig. 2d and 2e). Experimental evidence suggests that the notion of transactional segments and o-spaces can be applied to social robots as well [14], [10]. Figure 1 provides an example in the context of our simulation.

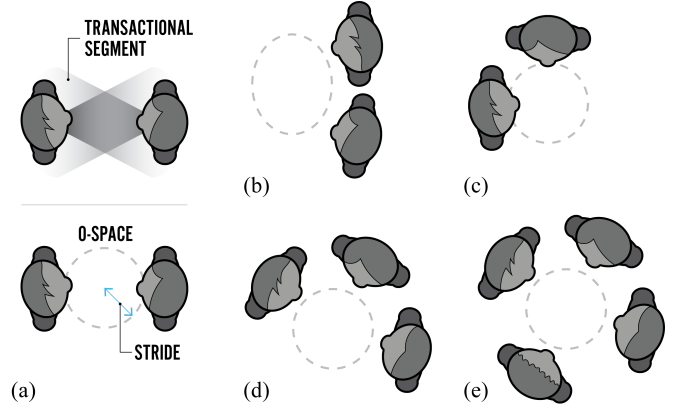


Fig. 2. Various spatial arrangements that may appear during human conversations. The top sketch in (a) shows the transactional segments of two people. The intersection of these segments is known as the o-space (dashed circles). The bottom sketch in (a) indicates the stride of the o-space, which is one of the main parameters of our simulation (see Sec. III-C).

B. Main Loop

Our simulation models a conversation in which the interactants establish an F-formation. To start, the simulation places the robot and the people in its conversation in a circular formation, and it chooses the first speaker and addressee randomly. The addressee may be another interactant or the whole group. The main loop of the simulation then proceeds as follows:

- 1) Update the simulated clock.
- 2) Update the state of the robot with the latest action that was chosen by the RL agent that controls it.
- 3) Choose new speaker and addressee if the previous speaker finished talking.
 - a) Set desired head orientation for the speaker.
 - b) Set desired head orientation for the people who are listening.
- 4) Update the states of all the people in the conversation.
- 5) Compute the reward for the robot.
- 6) Update the robot's internal representation of the state of the conversation based on its sensing capabilities.
- 7) Return the reward and new state representation to the RL agent so that it can choose the next action.

The set of actions that the robot can execute are angular velocity commands that change its orientation with respect to the group. The robot's representation of the state of the conversation is later detailed in Sec. IV.

In terms of head orientations, speakers are set to turn their heads toward their addressee or to the center of the o-space if they address the whole group. The other people are set to turn their heads towards the speaker, as described in the next section. Heads rotate at a fixed angular velocity towards their respective target, typically during multiple simulation steps.

For this work, we never allowed the robot to take a turn to speak. As a result, it had to adapt to the flow of the conversation set by the rest of the group.

C. Main Simulation Parameters

The main parameters that control the spatial arrangement of the group and the dynamics of the interaction are:

Number of interactants: The number of people in the conversation, including the robot.

O-Space center: The location of the o-space in the world-coordinate frame of the simulation.

Stride: The expected distance between the center of the o-space and the interactants (as depicted in Fig. 2a).

Time step: Time elapsed between simulation updates.

Robot actions: List of angular velocity commands that can be executed by the robot.

Speaking time distribution: Normal distribution that models how long a person typically speaks when he or she takes the turn to intervene.

Look-at noise: Whenever a person i in the simulation looks at the speaker, his or her head is set towards the angle

$$\beta_i = \arctan(d_y/d_x) + \varepsilon_i \quad (1)$$

with $\mathbf{d} = [d_x \ d_y]^T = \mathbf{p}_s - \mathbf{p}_i$ and $\varepsilon_i \sim \mathcal{N}(0, \sigma_i^2)$

The vectors \mathbf{p}_s and \mathbf{p}_i in eq. (1) denote the 2D position of the speaker and person i , respectively. Thus, \mathbf{d} is the direction towards the speaker from \mathbf{p}_i . The noise term ε_i controls how accurately the person looks at the speaker, depending on the variance σ_i . If the noise term makes the head turn more than 90 degrees from the front of the person, then β_i is clamped to prevent directing the head of person i backwards.

In general, the o-space center was at the origin of the world coordinate frame and the stride was 1.25 meters. For the actions of the robot, we allowed five angular velocity commands: $[-15.0, -7.5, 0.0, 7.5, 15.0]$ (in deg/sec). These commands were executed at 5Hz using a time step of 0.2 seconds.

The speaking time distribution was modeled as a normal distribution with mean $\mu = 5.0$ seconds and $\sigma = 2.0$. Samples from this distribution that were smaller than 0.5 seconds were clamped up to this value to avoid having extremely short speaking times.¹

D. Robot Perception

We modeled the robot as a platform with a small number of degrees of freedom, similar to Cobot [1] or Frog [2]. The robot had two sensors that were fixed to the front of its body. The first sensor was a camera that could be used to detect the position, as well as the head and body orientations of each person in the conversation. The second sensor was a microphone array that provided the angular directions toward nearby speakers from the perspective of the robot.

¹We acknowledge that this model is a crude approximation of the real dynamics of group conversations because people often speak for significantly longer than 5 seconds. We opted for short speaking times, though, because longer speeches result in fewer speakers in any given interaction. The fewer the speakers, the fewer times the robot needs to adjust its orientation, which simplifies the control problem.

The camera and the microphone array had configurable fields of view. People outside of these fields of view could not be sensed by the robot, and those within could be sensed with some probability. The specific values that we used for these parameters are provided in Section VI.

E. Reward

In this work, we assume that the correct behavior for the robot is to turn towards the speaker. Consequently, the reward r_{t+1} that the simulated environment provided to the robot as a result of taking an action a_t was:

$$r_{t+1} = \exp(-\varphi^2) + b \quad (2)$$

with $b = \begin{cases} 1 & \text{if } \text{abs}(\varphi) \leq \tau \text{ and } a_t == 0.0 \\ 0 & \text{otherwise} \end{cases}$

The angle $\varphi \in [-\pi, \pi]$ was the difference between the orientation of the robot and the angle representing the direction towards the speaker from the robot's position. The bonus b was given to reward zero angular velocity commands when the difference φ was small. In general, we used 10 degrees for τ to prevent oscillatory motions.

F. Limitations

Even though the simulation was useful to explore reinforcement learning techniques for motion control, it is by no means a perfect model of the real world. Our simulation did not capture all the complexity and variability of human behavior during group conversations nor sensing noise. Nonetheless, our efforts are an important first step towards testing RL techniques for the problem under consideration. The policies learned from our simulation can be considered to be prior knowledge to learn better behaviors with real users. Furthermore, the simulation allowed us to explore the sensitivity of several methods to particular types of noise, something that is hard to accomplish during human-robot interaction experiments [11].

IV. STATE REPRESENTATION

Our key contribution in this work is a state representation that is well suited to solve our motion control problem with RL techniques. This representation is composed of six features:

f1. Continuous feature in $[-\pi, \pi]$ representing the rotation that the robot needs to execute to direct its body towards the speaker. This value is the same as φ in eq. (2) if the speaker is within the field of view of the robot's microphone array and the sensor detected the audio signal coming from this person. Otherwise, $f1$ is set to zero by convention.

f2. Binary feature indicating if $f1$ is valid or not. This feature is zero if the speaker is out of the field of view of this sensor or the audio signal was not detected. Otherwise, $f2$ is one.

f3. Continuous feature in $[-\pi, \pi]$ representing the rotation that the robot needs to execute to direct its body towards the point of maximum *social saliency* induced by the people in its group (Fig. 3). Social saliency models gaze concurrences and is estimated using the primary gaze rays of the people

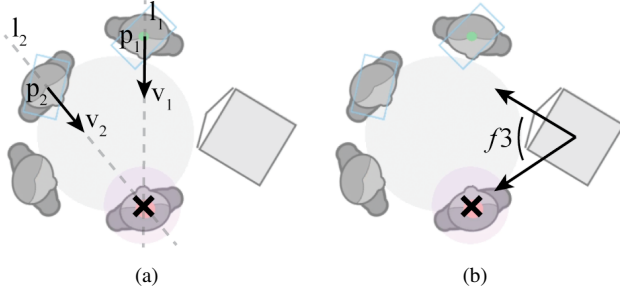


Fig. 3. Example of the primary gaze rays l used to compute the point of maximum social saliency (a) and the resulting feature $f3$ used in our state representation (b). The point of maximum social saliency is marked with an \times and surrounded by a light-colored circle. The primary gaze rays are estimated only for persons 1 and 2, who are visible through the robot’s cameras. These rays are estimated based on the position p and head direction v of each person.

visible through the robot’s camera, as described in [15]. The point of maximum saliency is the one that has the greatest contribution from the people who are visible (with ties broken randomly). If only one person is visible, we uniformly sample possible social saliency locations along the primary gaze ray of this person and use their average for $f3$. If nobody is visible, then $f3$ is set to zero by convention.

$f4$. Binary feature indicating if $f3$ is valid. This feature is zero if nobody was detected through the robot’s camera and, thus, social saliency could not be computed. Otherwise, $f4$ is one.

$f5$. Continuous feature in $[-\pi, \pi]$ representing the rotation that the robot needs to execute to orient its body towards the center of the o-space of its conversational group. We compute an estimate c of the true o-space location using an exponential moving average of center proposals:

$$c = (1.0 - n * a)c + a \sum_{i=1}^n \underbrace{p_i + s * u_i}_{\text{center proposal}} \quad (3)$$

where $a \in [0, 1]$ controls the contribution of every proposal, s is an expected value for the stride of the o-space, n is the total number of people visible through the robot’s camera, p_i is the position of the i -th person that is visible, and u_i is a unitary vector pointed in the same direction as the front of the body of person i . The model used to generate o-space center proposals $(p + s * u)$ is inspired by prior work [16], [17] and is used to initialize c with the proposal corresponding to the first person detected by the robot if no prior value has been computed. Note that c is not updated when no one is visible through the camera and that $f5$ takes the zero value when there is no estimate of c .

$f6$. Binary feature indicating if $f5$ is valid. If no o-space center has been estimated, then $f6$ is zero. Otherwise, $f6$ is one.

A. Properties

Our state representation is agnostic to the size of the group. This means that the same features can be used to describe conversations with a few people or with more interactants.

Because all of the features of the proposed representation are computed from the perspective of the robot, their descriptive power is dependent on the performance of the robot’s sensors. Inevitably, low detection probabilities and small fields of view negatively affect this representation.

In general, the point of maximum social saliency is closer to the true location of the speaker when more people are detected through the robot’s camera and more people look at the speaker. Likewise, the estimate of the o-space center converges to the true location faster when more people are detected.

V. EVALUATION

We performed a series of experiments to evaluate several RL agents in our simulated environment. Our goal was not to prove the superiority of one method, but rather to evaluate empirically what kind of approach may be better suited for our particular task and whether the proposed state representation generalized as expected. More precisely, our experiments focused on addressing:

- 1) whether a robot could quickly learn reasonable policies for the orientation task in our simulated environment;
- 2) how the performance of the RL agents under consideration degraded with noisy measurements;
- 3) how their performance could be affected by atypical human behavior; and
- 4) whether learned policies could generalize to conversations with more or fewer interactants.

To address the first goal above, we studied the amount of reward that several agents received as a function of time, as well as how quickly their performance saturated. This test included agents that estimated action values for discrete versions of the state space or that approximated them using linear regression. For the second and third goals, we studied how measurement noise affected the performance of the agents that, on average, learned good policies faster. For the last goal, we tested policies that were learned from interacting with 4 people in other group conversations.

A. Agents

We considered several agents for our evaluation. First, we decided to test model-based RL methods because they tend to be more sample efficient than model-free approaches when good transition and reward models can be learned quickly. These methods included TEXPLORE [18], which was designed for the robotics domain, and DYNA-2 [19], which can leverage prior experience while learning. Because the latter architecture uses Sarsa [12] to estimate the action-value function Q , then we also decided to test Sarsa by itself as a model-free method. Brief descriptions of the specific versions of these agents that we considered for our evaluation are provided below for completeness.

Sarsa(λ): Baseline on-policy learning agent [12]. This implementation discretizes the continuous features of the state space (Sec. IV) and estimates the action-value function

Q using a tabular representation. With any new tuple $(\mathbf{s}_t, a_t, r_t, \mathbf{s}_{t+1}, a_{t+1})$,

$$Q_{t+1}(\mathbf{s}, a) = Q_t(\mathbf{s}, a) + \alpha \delta_t e_t \text{ for all } (\mathbf{s}, a) \quad (4)$$

with $\delta_t = r_{t+1} + \gamma Q_t(\mathbf{s}_{t+1}, a_{t+1}) - Q_t(\mathbf{s}_t, a_t)$

where α is the learning rate, γ is the discount factor, and e_t is the (accumulating) eligibility trace [20]:

$$e_t = \begin{cases} \gamma \lambda e_{t-1} + 1 & \text{for } (\mathbf{s}_t, a_t) \\ \gamma \lambda e_{t-1} & \text{for all other state-action pairs} \end{cases} \quad (5)$$

which represents the credit assigned to state-action pairs for subsequent errors in evaluation.

For action selection, this agent uses the common ϵ -greedy policy, which chooses the best actions $\arg \max_a Q(\mathbf{s}, a)$ with a probability of $1 - \epsilon$. Otherwise, it selects a random action.

Sarsa(λ) with tile coding and adaptive learning rate: Sarsa agent with linear function approximation [21]:

$$Q(\mathbf{s}, a) = \phi(\mathbf{s}, a)^T \theta \quad (6)$$

where ϕ is a function that transforms the state-action pair to a large binary vector with tile coding [22] and θ is a collection of weights. The weights get updated by the rule $\theta_{t+1} = \theta_t + \alpha \delta_t e_t$, with the eligibility traces being $e_t = \gamma \lambda e_{t-1} + \phi(\mathbf{s}_t, a_t)$ and α being updated automatically according to [23]. This agent also uses an ϵ -greedy policy.

DYNA-2: RL architecture that combines sample-base learning with sample-based planning [19], as described in Algorithm 1. The agent has two “memories” that encapsulate all of the features and parameters used to estimate the value function. The *permanent* memory (ϕ, θ) is updated from real-world experiences and is used to compute the best overall estimate of the action-value function $Q(\mathbf{s}, a) = \phi(\mathbf{s}, a)^T \theta$ (“learn” procedure of Alg. 1). The *transient* memory $(\phi, \bar{\theta})$ is updated during simulations to track a local correction to the permanent memory (“search” procedure). This update is achieved with the combined action-value function $\bar{Q}(\mathbf{s}, a) = \phi(\mathbf{s}, a)^T \theta + \bar{\phi}(\mathbf{s}, a)^T \bar{\theta}$.

Algorithm 1: Main steps of DYNA-2

```

Procedure learn ( $\mathbf{s}_t, a_t, r_t, \mathbf{s}'_{t+1}$ )
  Store  $(\mathbf{s}_t, a_t, r_t, \mathbf{s}_{t+1})$  to update dynamics model
  search( $\mathbf{s}_{t+1}$ ) and pick next action  $a_{t+1} = \pi(\mathbf{s}, \bar{Q})$ 
  // update permanent memory (Sarsa)
   $\theta_{t+1} = \theta_t + \alpha[r_t + \gamma Q(\mathbf{s}_{t+1}, a_{t+1}) - Q(\mathbf{s}_t, a_t)]e_t$ 
  return ( $a_{t+1}$ )

Procedure search ( $\mathbf{s}$ )
  for  $k = 1$  to  $\text{num\_rollouts}$  do
    Initialize  $\bar{e}_0 = \mathbf{0}$  and  $\mathbf{s}_0 = \mathbf{s}$ 
    Pick action  $a_0 = \pi(\mathbf{s}_0, \bar{Q})$ 
    for  $t = 0$  to  $\text{max\_steps} - 1$  do
       $(\mathbf{s}_{t+1}, r_t) = \text{queryDynamicsModel}(\mathbf{s}_t, a_t)$ 
      Pick next action  $a_{t+1} = \pi(\mathbf{s}_{t+1}, \bar{Q})$ 
      // update transient memory
       $\bar{\theta}_{t+1} = \bar{\theta}_t + \bar{\alpha}[r_t + \gamma \bar{Q}(\mathbf{s}_{t+1}, a_{t+1}) - \bar{Q}(\mathbf{s}_t, a_t)]\bar{e}_t$ 

```

The main difference between Alg. 1 and its description by Silver and colleagues [19] is that we apply DYNA-2 to a non-episodic scenario with discounted returns ($\gamma < 1$). Moreover, we use Sarsa(λ) with tile coding and an adaptive learning rate to estimate θ and $\bar{\theta}$. For the transition and reward models, we use regression forests, as described in the next paragraphs for TEXPLORE. Whenever enough samples are collected to learn a new model, we clear DYNA’s transient memory so that it quickly adapts to the new dynamics.

TEXPLORE: Model-based architecture that uses sample-based planning [18]. In particular, this architecture uses a regression forest to estimate the transition and reward functions from real experience. Every time a query is made for planning, a random tree from the forest is chosen to make the prediction. This tree can be considered as one possible hypothesis of the true model of the domain. For planning, this architecture uses the UCT(λ) algorithm with discretized states-action spaces. Sample actions are selectively chosen using Upper Confidence Bounds [24].

In contrast with the original description of TEXPLORE [18], our implementation does not generalize action-values across depths in the search tree nor runs the act, plan, and model threads in parallel. The first modification was made because generalizing values resulted in poor performance in our particular domain. The second change simplified our implementation. Even though speed is important for robotics applications, such as ours, it was not a crucial factor for the present work.

B. Other Implementation Details

We used RL-Glue² as the interface between our simulated environment with the agents under consideration. For the Sarsa(λ) agent with tile coding, we used PyRL’s implementation.³ For the rest, we used our own implementation in Python, as described in the previous section. Moreover, we used the same model approximator for Dyna-2 and TEXPLORE. The approximator was a collection of independent regression forests for each of the dimensions of the state space and for the reward function, as proposed by Hester [18]. Our code extended the functionalities of the regression forest model of the Scikit Learn library⁴ by adding an option to query the prediction of a random tree in the forest.

VI. RESULTS

Unless otherwise noted, the results presented in this section are averages over 10 runs of 18000 steps (equivalent to 1 hour of interaction time). We set the field of view and detection probability of the microphone array on the robot to 100 degrees and 0.9, respectively. For the camera, we used 80 degrees and 0.75. These values were set based on our prior experience with off-the-shelf sensors of this kind.

The simulations had a total of 5 interactants (including the robot) which were arranged in a circular formation with

²<http://glue.rl-community.org>

³<https://github.com/amarack/python-rl>

⁴<http://scikit-learn.org>

a stride of 1.25 meters, as illustrated in Fig. 1. In general, future rewards were discounted with $\gamma = 0.7$.

Whenever the state space was discretized by an agent, we used 24 bins for each of the continuous angular features. For Sarsa(λ) and the ϵ -greedy policies, we set $\lambda = 0.4$ and $\epsilon = 0.1$, respectively. For the discrete Sarsa agent, $\alpha = 0.7$; for the continuous version, we used 64 tiles to approximate the Q function. In the case of DYNA-2 and TEXPLORE, we used regression forests with 15 trees to estimate a model of the dynamics. This model was updated every 300 samples (i.e., once a minute). For sample-based planning, we used 16 rollouts with 3 look-ahead actions. These parameters provided good results in our simulated environment.

A. Learning to Orient

First, we studied how quickly the RL agents under consideration learned to orient towards the speaker. We considered the same look-at noise distribution $\mathcal{N}(0, \sigma^2)$ in eq. (1) for all the people in the simulation, where σ was set to 0.0, 0.3, 0.6, 0.9, or 1.2 radians (which is equivalent to 0.0, 17.2, 34.3, 51.5, and 68.8 degrees). In the particular case where σ was zero, no noise was added to head orientations.

In general, all the agents converged to good policies within 1000 to 2000 steps (i.e., within 3.3 to 6.6 minutes) except for the baseline version of Sarsa with a tabular Q representation. The poor performance of this version of Sarsa with respect to the other agents can also be observed in Figure 4. This plot shows the number of speakers towards whom the robot failed to orient with an angular velocity of 0.0 rad/sec and with $abs(\varphi)$ in eq. (2) less than 10 deg. These can be considered speakers the the robot failed to acknowledge properly. The fact that all the agents but the baseline version of Sarsa found good policies quickly suggests that generalizing Q estimates across states is beneficial for our task.

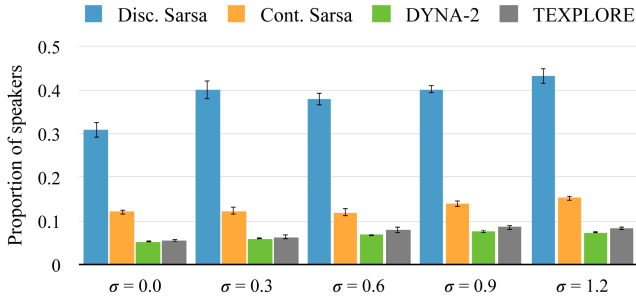


Fig. 4. Proportion of speakers towards whom the agents failed to orient as a function of the look-at noise (lower is better). “Disc. Sarsa” is the discrete version of Sarsa(λ) with tabular Q ; “Cont. Sarsa” is the continuous version with tile coding. Error bars represent standard errors.

Figure 5 shows the proportion of steps in which the agents achieved good behavior and received a bonus $b = 1.0$ as part of their reward (see eq. (2)). In general, model-based agents performed the best in terms of orienting properly towards the speakers. This result reinforces the idea that random forests are sample efficient when it comes to estimating dynamics models [18]. Furthermore, Fig.5 shows that the more people look away from the speaker, the worse the agents tend to

perform. This reduction in performance happens because higher σ values lead to fewer gaze concurrences, which is precisely what social saliency tries to estimate. One option to counter-act this effect is to estimate σ for every person during the conversation and incorporate this information into the estimate of social saliency (see Section 3.2 of [15]).

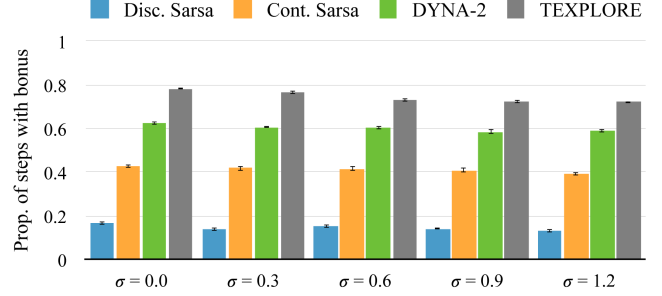


Fig. 5. Proportion of steps (out of 18000) in which the agents received a reward with a positive bonus. Results are groups by look-at distribution. Error bars represent std. errors.

Figure 6 shows the cumulative reward that the agents obtained during the 18000 steps of interaction time. TEXPLORE clearly outperformed the other agents in this respect, likely because of its better policy in comparison to ϵ -greedy.

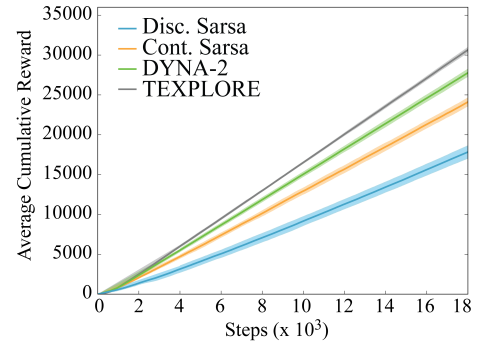


Fig. 6. Average cumulative reward for $\sigma = 0.0$. The shaded areas around the curves represent the standard error. Similar trends were obtained for the other look-at noise distributions.

B. Sensitivity to the Detection Probabilities

Because multiple factors can affect robot perception, such as background audio or illumination, we decided to further investigate how different detection probabilities for the robot’s sensors affected its performance. In particular, we focused on evaluating DYNA-2 and TEXPLORE in this experiment, given that they performed the best previously.

We evaluated two detection probabilities for the microphone array (0.75 and 0.90) and three for the camera (0.60, 0.75 and 0.90). As before, this experiment was repeated 10 times per agent until it completed 18000 steps. For the look-at noise distribution, we used $\mathcal{N}(0, 0.6^2)$ for everybody.

Figure 7 shows the proportion of steps in which the robot received a positive reward in each case. In general, the results were affected by the detection probability of the microphone array. The lower the probability, the fewer times the robot received a positive bonus. Interestingly, the results did not vary much with lower detection probabilities for the camera.

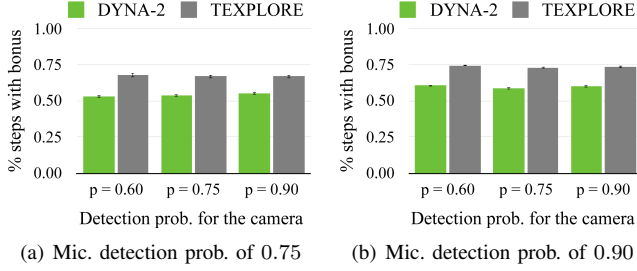


Fig. 7. Proportion of steps (out of 18000) in which the agents received a positive bonus as part of their reward. The left/right plots shows the results when the detection probability of the microphone array was 0.75/0.90.

These results are encouraging for future tests in real human-robot interactions because audio detection using microphone arrays tends to be more reliable than people detection with computer vision approaches. It is worth noting, though, that the wide field of view of the microphone array that we modeled for the robot and the lack of false positive detections likely influenced these outcomes.

C. Individual Behaviors

So far, we have considered situations in which the people in the conversation are all affected by the same look-at noise distribution and do not move. Of course, this is not realistic. People often exhibit individual behaviors that differentiate them from others. To study these types of situations, we investigated the performance of DYN-2 and TEXPLORE when one person was affected by more look-at noise than the other people as well as when people slightly adjusted their position with respect to the rest of the group.

1) *Non-Uniform Look-At Noise*: For this test, we set the look-at noise distribution of one person to $\mathcal{N}(0, 1.2^2)$ and the rest to $\mathcal{N}(0, 0.6^2)$. Because there were four people in the simulation, we tested all four combinations with one outlier.

We found that the outlier look-at noise distribution did not affect the performance of the agents; the results were similar to those obtained for the experiment of Sec. VI-A. In particular, the proportions of steps in which the robot received a bonus reward were 0.58 (STE < 0.01), 0.58 (STE < 0.01), 0.59 (STE = 0.01) and 0.59 (STE = 0.01) with DYN-2. With TEXPLORE, the proportions were 0.72 (STE = 0.01), 0.73 (STE < 0.01), 0.73 (STE < 0.01), and 0.73 (STE = 0.01). This result is not surprising given that the agents relied more on audio detections than visual information, as discussed in Sec. VI-B.

2) *Changes in Location*: We modified our simulation to induce small re-configurations of the people in the conversation. Every time a new speaker was selected, as described in Sec. III-B, we flipped a coin with a success probability p_t for every other person in the conversation. If the outcome of a flip was a success, we set a desired new position \mathbf{p}'_i for the corresponding person i and updated his or her position towards this location at a constant velocity. In particular,

$$\mathbf{p}'_i = \begin{cases} \mathbf{p}_i + \mathbf{t}_i & \text{if } \|\mathbf{p}_i - \mathbf{p}_i^{ini}\| < 0.5 \text{ meters} \\ \mathbf{p}_i^{ini} & \text{otherwise} \end{cases} \quad (7)$$

with \mathbf{p}_i^{ini} the initial location of person i at the beginning of the simulation, \mathbf{p}_i their previous location, and \mathbf{t}_i a translation drawn from a 2D normal distribution with mean $\mathbf{0}$ and covariance $[0.01 \ 0.0; 0.0 \ 0.01]$. In this manner, equation (7) induced controlled re-configurations of the spatial arrangement while preventing dissolving the group's F-formation.

For the test with translational motion, we considered four success probabilities p_t (0.1, 0.2, 0.3, and 0.4). In general, we used a constant look-at noise distribution of $\mathcal{N}(0, 0.6^2)$ and set the detection probabilities of the microphone array and the camera to 0.9 and 0.75, respectively.

Even though DYN-2 and TEXPLORE learned good policies with translational motion, we found that this motion slightly reduced the learning speed of DYN-2 in comparison to using $p_t = 0.0$ (as in Sec. VI-A). This outcome is illustrated in Figure 8. Each of the plots of this figure show the absolute angular difference between the robot and the direction towards the speaker from its location ($abs(\varphi)$ in eq. (2)). The baseline DYN-2 (without translational motion) converged to an absolute offset of about 0.2 radians (11.5 degrees) after 12000 steps, whereas DYN-2 took longer to converge with $p_t > 0.0$. TEXPLORE seemed to perform slightly better with translational motions, likely because its policy was better suited for our problem.

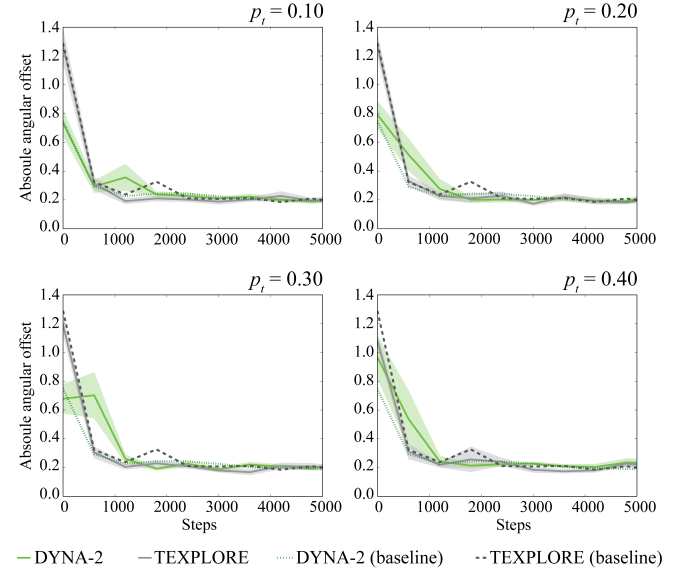


Fig. 8. Absolute angular offset (in radians) between the direction of the robot and the direction towards the speaker. The results were averaged over windows of 600 contiguous steps and over 10 runs for each agent. The shaded areas behind the DYN-2 and TEXPLORE lines represent standard errors. Baseline results correspond to $p_t = 0.0$ (no translational motion).

D. Generalization To Other Groups

Finally, we decided to test how well pre-trained agents performed in other conversations with different numbers of people. For this test, we exposed the DYN-2 and TEXPLORE agents that were trained with 4 people for the experiment of Sec. VI-A to interactions with 2, 3, 5 and 6 people. For interactions with less than 4 people, we used a stride of 1.25 meters, as in the other experiments. When

more people conversed with the robot, we increased the stride to 1.5 meters to accommodate the extra participants. For this experiment, we also let the agents adjust their Q value estimates in an online fashion. Each run lasted a total of 3000 steps (10 minutes of interaction time).

Figure 9 shows the average number of steps for which the agents received a bonus reward in the new environment. As expected, the agents that were pre-trained outperformed agents that started to learn from scratch. This result shows the generalization power of our state representation.

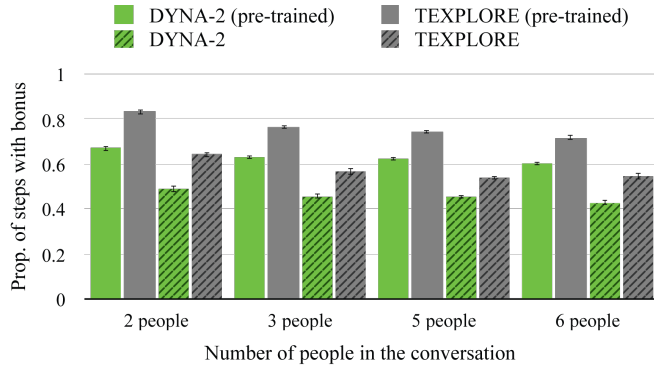


Fig. 9. Proportion of steps (out of 3000) in which the robot got a reward with a positive bonus. Pre-trained agents interacted with 4 people prior to this test; other agents learned from scratch. Error bars represent std. errors.

VII. DISCUSSION

We proposed to use reinforcement learning to control the orientation of a robot during group conversations. To start evaluating this type of approaches, we built a simulated environment based on models from social psychology that explain spatial behavior during these interactions. Our main assumption in this environment was that the robot should turn towards the speaker to convey attentiveness to the conversation and maintain awareness of the focus of attention.

Even though our simulation did not fully model the real world, it was useful for conducting a series of experiments to evaluate the performance of RL agents with a new state representation that we designed for this task. This state representation is composed of features that can be computed using standard sensors from a mobile robot and is agnostic to the number of people in the conversation. This last property helps generalize learned behaviors to other interactions.

Our experimental results showed that model-based RL agents were able to find good policies for our robot within a few minutes of interaction time. In addition, we found that random-forest approximators, as proposed in [18], were a good choice to estimate the dynamics model of our simulation and execute search-based planning.

This work is a first step towards automatically generating spatial behavior for robots during group conversations. Moving forward, we plan on evaluating online RL techniques for our motion control task on real user interactions. In particular, we are interested in using the policies that we learned from our experiments as prior knowledge for controlling the orientation of a real robot. We expect this prior to reduce the time that the robot needs to learn good policies in practice.

ACKNOWLEDGMENTS

We thank the Walt Disney Corporation for continued support of this research effort, as well as Christoph Dann and Emma Brunskill for their feedback and suggestions.

REFERENCES

- [1] M. M. Veloso, J. Biswas, B. Coltin, and S. Rosenthal, “CoBots: Robust Symbiotic Autonomous Mobile Service Robots,” in *Proc. of IJCAI’15*, 2015.
- [2] V. Evers, N. Menezes, L. Merino, D. Gavrilu, F. Nabais, M. Pantic, P. Alvitto, and D. Karreman, “The Development and Real-world Deployment of FROG, the Fun Robotic Outdoor Guide,” in *Proc. of HRI’14*, 2014.
- [3] M. Vázquez, A. Steinfeld, S. E. Hudson, and J. Forlizzi, “Spatial and Other Social Engagement Cues in a Child-robot Interaction: Effects of a Sidekick,” in *Proc. of the 2014 ACM/IEEE Int’l Conference on Human-robot Interaction*, ser. HRI ’14, 2014, pp. 391–398.
- [4] J. Vroon, M. Joosse, M. Lohse, J. Kolkmeier, J. Kim, K. Truong, G. Englebienne, D. Heylen, and V. Evers, “Dynamics of social positioning patterns in group-robot interactions,” in *Proc. of RO-MAN ’15*, 2015, pp. 394–399.
- [5] D. E. Karreman, G. D. Ludden, E. M. van Dijk, and V. Evers, “How can a tour guide robot’s orientation influence visitors’ orientation and formations?” in *Proc. of the Int’l Symp. on New Frontiers in HRI*, 2015.
- [6] M. A. Yousuf, Y. Kobayashi, Y. Kuno, A. Yamazaki, and K. Yamazaki, “Development of a Mobile Museum Guide Robot That Can Configure Spatial Formation with Visitors,” in *Intelligent Computing Technology*, ser. Lecture Notes in Computer Science, 2012, vol. 7389, pp. 423–432.
- [7] R. Mead and M. J. Matarić, “Perceptual models of human-robot proxemics,” in *Proc. ISER’14*, 2014.
- [8] J. Kober, J. A. Bagnell, and J. Peters, “Reinforcement Learning in Robotics: A Survey,” *The Int’l J. of Robotics Research*, 2013.
- [9] M. Saerbeck and C. Bartneck, “Perception of Affect Elicited by Robot Motion,” in *Proc. of HRI’10*, 2010.
- [10] H. Kuzuoka, Y. Suzuki, J. Yamashita, and K. Yamazaki, “Reconfiguring Spatial Formation Arrangement by Robot Body Orientation,” in *Proc. of HRI’10*, 2010, pp. 285–292.
- [11] A. Steinfeld, O. C. Jenkins, and B. Scassellati, “The Oz of Wizard: Simulating the human for interaction research,” in *Proc. of HRI’09*, 2009.
- [12] R. S. Sutton and A. G. Barto, “*Reinforcement Learning: An Introduction*,” The MIT Press, 2015, (second edition, in progress).
- [13] E. Goffman, *Behavior in public places: Notes on the social organization of gatherings*. Free Press of Glencoe, 1963.
- [14] A. Kendon, *Conducting Interaction: Patterns of Behavior in Focused Encounters*. Cambridge Univ. Press, 1990.
- [15] H. S. Park, E. Jain, and Y. Sheikh, “3D Social Saliency from Head-Mounted Cameras,” in *Proc. of NIPS’12*, 2012.
- [16] F. Setti, C. Russell, C. Bassetti, and M. Cristani, “F-formation Detection: Individuating Free-standing Conversational Groups in Images,” *CoRR*, vol. abs/1409.2702, 2014.
- [17] M. Vázquez, A. Steinfeld, and S. E. Hudson, “Parallel Detection of Conversational Groups of Free-Standing People and Tracking of their Lower-Body Orientation,” in *Proc. of IROS’15*, 2015.
- [18] T. Hester, “TEXPLORE: Temporal Difference Reinforcement Learning for Robots and Time-Constrained Domains,” Ph.D. dissertation, The University of Texas at Austin, Texas, USA, December 2012.
- [19] D. Silver, R. S. Sutton, and M. Müller, “Temporal-difference search in computer Go,” *Machine Learning*, vol. 87, no. 2, pp. 183–219, 2012.
- [20] R. S. Sutton, “Temporal Credit Assignment in Reinforcement Learning,” Ph.D. dissertation, University of Massachusetts Amherst, 1984.
- [21] Richard S. Sutton, “Generalization in reinforcement learning: Successful examples using sparse coarse coding,” *Advances in Neural Information Processing Systems*, pp. 1038–1044, 1996.
- [22] R. S. Sutton, “Tile Coding Software,” <http://rlai.cs.ualberta.ca/RLAI/RLtoolkit/tiles.html>, Online; accessed Feb. 2016.
- [23] W. Dabney and A. Barto, “Adaptive Step-Size for Online Temporal Difference Learning,” in *Proc. of AAAI’12*, 2012.
- [24] L. Kocsis and C. Szepesvári, “Bandit Based Monte-Carlo Planning,” in *Proc. of ECML’06*, 2006.