DELIVERABLE D5.1

INITIAL SOCIAL SKILLS LEARNING COMPONENT AND SIMULATION ENVIRONMENT

Simon Keizer (HWU) and Oliver Lemon (HWU)

**Beneficiaries:** HWU (lead)  
**Workpackage:** WP5: Machine Learning for Social Skills Execution  
**Description:** This deliverable describes the initial social skills execution and learning component, along with the simulation environment that is used for testing and training. The social skills executor (SSE) generates output actions for the system, including both communicative and non-communicative actions. The SSE is modelled as a hierarchy of Markov Decision Processes (MDPs) with policies that can be trained simultaneously using Hierarchical Reinforcement Learning. The simulation environment consists of multiple simulated users that enter the scene, try to get the system’s attention, and order a drink. The SSE policies are optimised in interaction with this simulation environment, making use of the reward signals provided by the simulated users.

**Version:** Final  
**Nature:** Prototype (P)  
**Dissemination level:** Public (PU)  
**Pages:** 11  
**Date:** 2012-07-31
1 Executive Summary

This deliverable describes the initial social skills execution and learning component, along with the simulation environment that is used for testing and training. The Social Skills Executor (SSE) generates output actions for the system, including both communicative and non-communicative actions. In Figure 1 the architecture of the JAMES system is given. The SSE takes as input the social state estimated by the Social State Recogniser (SSR) developed in WP3 and sends output actions via a Switch to the Output Planner, which realises the actions by passing information to the Talking Head Controller and Robot Motion Planner. The SSE is an alternative to the Planner/Execution Monitor developed in WP4: whereas the latter generates actions based on real-time planning [5], the SSE generates actions based on learned policies. The Switch is used to choose between the two alternatives.

The decision-making process of the SSE consists of three stages: 1) social multi-user coordination: managing the system’s engagement with the users in the scene (e.g., accept a user’s bid for attention, or proceed with an engaged user); 2) single-user interaction: if deciding to proceed with an engaged user, generating a response to that user, in the form of a communicative act (e.g., asking what drink the user wants) or physical action (e.g., serving a drink); and 3) multi-modal fission: selecting modalities for realising a selected communicative action, for example for a greeting, use speech only (say “Hello”), or speech combined with a head gesture (say “Hello” and nod).

The SSE is modelled as a hierarchy of two Markov Decision Processes (MDPs), representing the social multi-user coordination and single-user interaction decision stages. The two associated policies are trained simultaneously using Hierarchical Reinforcement Learning.

The simulation environment consists of multiple simulated users that enter the scene, try to get the system’s attention, and order a drink. The SSE can be run in interaction with this environment, not only for testing, but also for training SSE policies. Based on the reward signals provided by the simulated users, the SSE policies are jointly optimised by maximising the long term cumulative reward, using a Monte Carlo Control algorithm [7].

To some extent, this approach is based on the statistical dialogue management framework set out in [10], but...
extended to the case of multi-user, multi-modal interactions. Note also that the interactions with the simulation environment happen on a frame-by-frame basis, as opposed to event-based (or turn-based) interactions in standard spoken dialogue systems. Via the generated behaviour of the simulated users, the simulation environment generates an audio-visual input stream that is processed by the SSR. In the actual JAMES system, this input stream is obtained from the vision and speech processing modules. Based on the social state that is maintained by this SSR, the SSE selects system actions according to its learned strategy. The output of the SSE also constitutes a stream of observations, processed by the simulation environment.

2 Evaluation

As a first evaluation, the SSE has been trained and evaluated in interaction with the simulated environment. Performance is measured in terms of success rate and average reward. If a user was able to place an order and was served the correct drink, this counts as a success. Each user provides a reward signal, composed of a positive score for a successful interaction, and negative scores (i.e., penalties) for the time taken to be served, and for various types of undesired system responses.

The results of our experiments on SSE strategy learning are very promising. The learned strategies outperform a variety of baselines by up to 15% in terms of success rate. In particular, we show that in noisy conditions, the learned strategies are more robust than the hand-coded strategies that were evaluated. More details about these results are given in the attached paper (see Section 5).

In order to generate realistic simulated data, the simulation environment is for a part based on the collected human-human data from WP8. In addition, we will make use of the data that has been (and will be) collected with the JAMES system itself, to improve the simulation. The simulation will be evaluated by measuring the similarity between the simulated and real data, e.g. in terms of the Kullback-Leibler (KL) divergence between distributions. We will build on previous work on evaluation of user simulations for spoken dialogue systems [6, 3], extending it for generating multi-user, multi-modal, situated interactions.

3 Future work

The next step in evaluating the SSE will be to integrate the component into the actual JAMES system and evaluate it with human users, in a similar experiment as for the initial JAMES prototype [2]. This evaluation will again result in new data, which can be exploited to improve the simulated environment. In particular, user satisfaction scores obtained in the evaluation can be used to refine the reward function of the simulated users. A PARADISE style regression [8] can be performed to correlate objective features with user satisfaction, and hence the SSE strategy can be trained to maximise user satisfaction.

In order to make the SSE more robust to noisy observations (in both speech and vision input), the MDP framework will be extended to a POMDP (Partially Observable Markov Decision Process) framework. Instead of finding optimal policies that map single estimated states to actions, we plan to develop POMDP policies that operate on distributions over states. This framework allows to explicitly represent uncertainty about the estimated states, caused by noisy input observations, and this uncertainty can be exploited in learning optimal policies for social interaction.

As an alternative to the supervised approach of learning to recognise social states (WP3) and using reinforcement learning for optimising policies for social interaction that operate on such states (WP5), an overarching unsupervised approach is being developed as well. In this approach, rather than using input observations to maintain a model of the social state (with its own explicit representation), a non-parametric Bayesian method is proposed to automatically infer distributional representations of POMDP states from the observation stream. A major advantage of this approach is that no annotation of social states is required. The POMDP model with the underlying non-parametric state-inference is essentially an Infinite POMDP (iPOMDP) [1], which has been
adapted to multimodal interaction [9]. Action selection policies for this iPOMDP can be found using techniques such as forward search or reinforcement learning in interaction with the simulated environment. In the final deliverable D5.2 we will report in more detail on this path of research.

As a follow-up on recent work by a MSc student at Heriot-Watt who developed a basic multi-user, multi-modal bartender application for a NAO robot [4], we will also explore the integration of a version of the social skills executor into a NAO-based bartender system.

4 Software

Both Social Skills Executor and Simulated Environment are implemented in java. In addition, there are shell scripts and Perl scripts for running policy optimisation and evaluation experiments. The source code is available on request and via the project SVN repository.

The main java classes in the implementation are:

- **SocialSceneAgent**: represents a generic agent operating in a multi-agent environment.
- **DialogueAct**: represents a communicative action, realised through a combination of speech and/or gestures.
- **SocialSceneObservation**: represents a multi-channel observation for a single time-frame, consisting of a speech channel (represented as a DialogueAct) and a set of vision channels for each user in the scene.
- **Social Skills Executor**: represents the bartender agent (subclass of SocialSceneAgent).
- **InformationState**: represents the information state of the bartender agent, containing information about which users the agent believes are in the social scene, what their engagement state is, what their goal is, whether they have been served, etcetera.
- **UserState**: represents the information the bartender agent has about a single user.
- **Policy**: represents an action selection policy of the bartender agent, that essentially defines a function that maps state-action pairs to values, and contains the main update function for learning (the Social Skills Executor has two instances of this class, corresponding to the two MDPs it uses for decision-making).
- **SocialSceneSimulator**: represents a multi-user simulated environment.
- **UserSimulator**: represents a single user (subclass of SocialSceneAgent).

5 Attached paper

The following paper is attached to this report:


Abstract: We present a novel framework for learning socially intelligent interaction strategies for multi-user, situated, multi-modal, systems. The framework is developed using the example of a robot bartender that tracks multiple customers, takes their orders, and serves drinks. It consists of an Interaction Manager (IM) that maintains a model of the social state and selects actions given that state, and a Simulated Environment consisting of multiple simulated users. The action selection part of the IM is modelled through a hierarchy of two policies: a social interaction policy for managing engagement with multiple users, and a policy for managing the interaction with a single engaged user. The two policies can be optimised jointly using Hierarchical Reinforcement Learning.
We present promising results on policy optimisation for this framework, showing that learned policies outperform a variety of baselines by up to 15% in terms of success rate, and that they are more robust to noise.

References


ABSTRACT

We present a novel framework for learning socially intelligent interaction strategies for multi-user, situated, multi-modal, systems. The framework is developed using the example of a robot bartender that tracks multiple customers, takes their orders, and serves drinks. It consists of an Interaction Manager (IM) that maintains a model of the 'social state' and selects actions given that state, and a Simulated Environment consisting of multiple simulated users. The action selection part of the IM is modelled through a hierarchy of two policies: a social interaction policy for managing engagement with multiple users, and a policy for managing the interaction with a single engaged user. The two policies can be optimised jointly using Hierarchical Reinforcement Learning. We present promising results on policy optimisation for this framework, showing that learned policies outperform a variety of baselines by up to 15% in terms of success rate, and that they are more robust to noise.

Index Terms— Human-robot interaction, multi-user simulation, reinforcement learning

1. INTRODUCTION

This paper discusses machine learning techniques for optimising action selection policies of embodied conversational agents in situated interaction with multiple users. We use the example domain of a robot bartender that interacts with multiple customers, taking their orders and serving the drinks that were ordered. The robot system is equipped with vision and speech input processing modules, as well as modules controlling two robot arms and a talking head. Based on observations about which users are in the scene and their behaviour, the agent must maintain a model of the social context, and decide on effective and socially appropriate responses in that context. Such a system must be able to engage in, maintain, and close interactions with users, take a user’s order by means of a spoken conversation, and serve their drinks. The overall aim is to generate interactive behaviour that is both task effective and socially appropriate: in addition to efficiently taking orders and serving drinks, the system should also deal with customers in a first-come, first-served basis, it should manage the customers’ patience by asking them politely to wait until the robot is done serving another customer, and so on. A typical session with two users would be:

A customer approaches the bar and looks at the bartender
ROBOT: [Looks at Customer 1] How can I help you?
CUSTOMER 1: A pint of cider, please.

Another customer approaches the bar and looks at the bartender
ROBOT: [Looks at Customer 2] One moment, please.
ROBOT: [Serves Customer 1]
ROBOT: [Looks at Customer 2]
Thanks for waiting. How can I help you?
CUSTOMER 2: I’d like a pint of beer.
ROBOT: [Serves Customer 2]

Our general aim is to use machine learning techniques for both recognising social states based on (generally incomplete and/or incorrect) audio-visual input and selecting actions based on the current estimated social state [1]. In this paper, we focus on the action selection part of this problem, in other words, learning social skills in multi-user human-robot interaction. In spoken dialogue systems research, the approach of modelling dialogue as a Markov Decision Process (MDP) and using Reinforcement Learning to automatically optimise action selection policies [2] has become very popular. More recently, this approach has been explored in situated interaction as well [3]. Here we follow this paradigm and extend it to multi-party, social interaction. By designing a simulated environment with multiple simulated users that can respond to a variety of system behaviours and reward and/or penalise interactions according to general design principles, the system’s action selection component can be automatically optimised for both effective and socially appropriate behaviour. Additionally, the action selection component we propose is triggered in each time-frame, rather than

The work described in this paper was partly funded by the EU FP7 Programme under Grant agreement No. 270435 (JAMES project: www.james-project.eu).
in each turn, as happens in more traditional interaction management approaches.

The paper is organised as follows. After discussing previous related work in Section 2, we present our learning framework, consisting of an Interaction Manager (IM) and a multi-user Simulated Environment (SE) (Section 3). Then, in Section 4, the policy optimisation model is described, followed by the evaluation in Section 5.

2. RELATED WORK

In recent years, there has been increasing interest in situated, multi-party interaction. Bohus and Horvitz developed a virtual receptionist that can detect people approaching a reception desk and can interact with multiple users [4]. In particular, a model was developed that uses visual cues for tracking the engagement state of users, and a machine learning approach for predicting engagement intentions. Earlier work already focused on this central aspect of recognising engagement in situated interaction (and in human-robot interaction in particular), but most of these approaches have been rule-based, see for example [5]. More recently, Klotz et al. [6] developed a rule-based multi-user engagement and interaction model, demonstrated with a NAO robot.

There has been some research on using machine learning techniques in human-robot interaction, though not for multi-user interactions. One such approach involved modelling the interaction as a Semi-Markov Decision process and using Hierarchical Reinforcement Learning for optimising decision making [3, 7]. An alternative approach to using multiple policies within a dialogue manager also incorporated POMDP models, but still focused on single-user interactions [8].

3. SOCIAL SKILLS LEARNING FRAMEWORK

Our learning framework consists of two main parts: 1) an Interaction Manager (IM), which processes audio-visual input and generates multi-modal output actions for the system to execute, and 2) a Multi-User Simulated Environment (MUSE). Fig. 1 shows the architecture of the framework. The IM consists of a) a Social State Recogniser (SSR) which maintains an estimated model of the social state based on input observations received from the vision and speech input modules as well as feedback about executed system actions, and b) a Social Skills Executor (SSE) which selects output actions based on the current recognised social state.

3.1. Social state recogniser

Our Social State Recogniser (SSR) maintains a model of the social state on the basis of incoming observations. The SSR incorporates an engagement model for coordinating interactions with the users in the scene, and a task model for interactions with individual users (involving getting a user’s order and serving the requested drink for example). The input observations are interpreted in terms of communicative actions, which are taken from a multi-dimensional dialogue act taxonomy underlying a recently developed ISO standard for dialogue act annotation [9]. In a multi-party, situated setting, agents need to be able to initiate, maintain, and end engagements with other agents. For both the bartender system and simulated users, we use a finite state engagement model driven by dialogue acts and gaze behaviour (see Fig. 2).

If a user conveys an intention to engage with the system (i.e., bids the system for attention), this is represented with the dialogue act attentionFeedbackElicitation, which changes the engagement state from NON-ENGAGED to USR_BIDD_ATT. Only after the system has accepted the bid for attention (through an attentionAutoPositive act), user and system are considered to be ENGAGED. The system may however choose to continue an interaction with another, engaged, user, and so the engagement state remains USR_BIDD_ATT, unless the user stops bidding and the state transitions back to NON-ENGAGED. This makes the model different from Bohus and Horvitz’ engagement model, which only distinguishes between the ENGAGED and NOT-ENGAGED states. Once an interaction is closed, the agents can simply look away if they want to disengage.

![Fig. 2. Finite state engagement model.](image)

The maintained social state contains models for each user in the social scene. Every such user model contains information about the user’s engagement state, their location (e.g., are they standing at the bar?), the user goal (what kind of drink do they want?), and whether they have been served a drink yet.

3.2. Social skills executor

The Social Skills Execution (SSE) component determines the system’s behaviour, based on the current social context. The output actions include both abstract communicative and non-communicative acts, as well as descriptions of their multimodal realisations. The generated communicative actions have the form of dialogue acts from the taxonomy mentioned above [9], and are associated with combinations of modalities to use for realisation, for example, a greeting can be realised by combinations of speech (“Hello”) and nodding (robot head movement). In the current system, non-communicative actions are limited to putting a bottle with a particular type of
drink in front of the user (i.e., serving a drink). In Section 4, Table 4, the supported actions are listed.

The decision making process consists of three main stages: 1) **social multi-user coordination**: managing the system’s engagement with the users present in the scene (e.g., accept a user’s bid for attention, or proceed with an engaged user), 2) **single-user interaction**: if deciding to proceed with an engaged user, generating a high-level response to that user, in the form of a communicative act or physical action (e.g., serving a drink) and 3) **multi-modal fission**: selecting a combination of modalities for realising a chosen response (e.g., speech and/or head gestures). One advantage of such a hierarchical design is that strategies for the different stages can be developed independently. Another is that it makes automatic policy optimisation more scalable. Note that the hierarchy of decision-making is followed in each time-frame, and that the realisation of an action has a certain duration, measured as a number of time-frames. The IM therefore not only processes input signals on a frame-by-frame basis, but also makes a decision about what to do in every frame. This is also the case for the Simulated Environment discussed below.

### 3.3. Multi-user simulated environment

To provide a testing environment for the Interaction Manager, we developed a multi-user simulated environment (MUSE). Not only can this environment be used for testing and evaluating the IM, we also use it for training action selection policies of the Social Skills Execution (SSE) component of the IM. The MUSE allows us to rapidly explore the large space of possible states in which the SSE will have to select actions. A reward function that incorporates individual rewards from all simulated users in the environment is used to encode preferred system behaviour in a principled way. A simulated user assigns a reward if they are served the correct drink, and gives penalties associated with their waiting time and various other forms of undesired system responses (see Section 4 for the definition of reward). All of this provides a practical platform for evaluating different strategies for social behaviour and also paves the way for automatic optimisation of policies, for example by using reinforcement learning techniques, as we will discuss in Section 4.

The simulation environment replaces the vision and speech processing modules in the actual robot bartender system, which means that it generates 1) vision signals in every time-frame, and 2) speech processing results, corresponding to sequences of time-frames where a user spoke. The vision observations contain information about users that have been detected, where they are in the scene, whether they are speaking, and where their attention is directed to. Speech processing results are represented semantically, in the form of dialogue acts. The SSR fuses the vision and speech input, for example to associate an incoming dialogue act with a particular user.

The simulated signals are the result of combining the output from the simulated users in the environment. Each simulated user is initialised with a goal, enters the scene, and starts bidding for attention. As with the SSE component, the simulated users also maintain a state and based on that state, generate responses at an abstract level as well as lower-level multimodal realisations of these responses. They do this on a frame-by-frame basis. Additionally, the simulated users start with a given patience level, which is reduced in every frame the user is bidding for attention or being served by the system. If a user’s patience has reduced to zero, s/he gives up and leaves the bar. However, it is increased with a given fixed amount when the system politely asks the user to wait, encoded as a pausing dialogue act. The behaviour of the sim-
ulated users is partly controlled by a set of probability distributions that allow for a certain degree of variation. These distributions have been informed by statistics derived from a corpus of human-human customer-bartender interactions [10].

Additionally, the MUSE provides feedback about the execution of robot actions. Since the execution of actions has a duration in this framework, it is relevant for the IM to know when actions have been completed (or interrupted). This type of information simulates the feedback that is provided in the actual bartender system by a robot controller.

4. POLICY OPTIMISATION IN SOCIAL INTERACTION

To set up automatic optimisation of strategies for social interaction, we designed two Markov Decision Processes (MDPs), corresponding to the social multi-user coordination and single-user interaction stages, discussed in Section 3.2. Both MDPs have their own state spaces $S_1$ and $S_2$, each defined by a set of state features, extracted from the social state that was estimated by the SSR, see Tables 1 and 3. They also have their own action sets $A_1$ and $A_2$, corresponding to the range of decisions that can be made at the two stages, see Tables 2 and 4, and two policies $\pi_1 : S_1 \rightarrow A_1$ and $\pi_2 : S_2 \rightarrow A_2$, mapping states to actions.

<table>
<thead>
<tr>
<th>Table 1. State features for the social multi-user coordination policy. For each user, 4 features are included in the state space, which means that the value function is defined over $4 \cdot 2^3 = 32$ states for 1 user, which increases to 1024 states for 2 users and 32,768 states for 3 users.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Index</strong></td>
</tr>
<tr>
<td>$4 \cdot (i - 1)$</td>
</tr>
<tr>
<td>$4 \cdot (i - 1) + 1$</td>
</tr>
<tr>
<td>$4 \cdot (i - 1) + 2$</td>
</tr>
<tr>
<td>$4 \cdot (i - 1) + 3$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2. Actions for the social multi-user coordination policy.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Index</strong></td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>$3 \cdot (i - 1) + 1$</td>
</tr>
<tr>
<td>$3 \cdot (i - 1) + 2$</td>
</tr>
<tr>
<td>$3 \cdot (i - 1) + 3$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 3. State features for the single-user interaction policy. In this case, there are $4 \cdot 7 = 28$ states.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Index</strong></td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 4. Actions for the single-user interaction policy, which correspond to possible dialogue acts, except for disengaging and serving a drink. The specific drink types required for two of the actions are extracted from the fully specified user goal in the social state maintained by the IM.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Index</strong></td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
</tbody>
</table>

5. EVALUATION

To evaluate this learning methodology, we ran 10 policy optimisations in interaction with the MUSE running 2 simulated time-frame, and is the sum of the rewards $R_i$ defined for each individual user $i$:

$$ R_i = 350 \cdot TC_i - 2 \cdot W_i - TO_i - SP_i $$

where $TC_i$ (Task Complete) is a binary variable indicating whether the user was able to order a drink and the drink was served, $W_i$ (Waiting) is a binary variable indicating whether the user was waiting for the system to prepare the drink, $TO_i$ (Task Ongoing) is a binary variable indicating whether the user is currently engaging with the system, and $SP_i$ (Social Penalties) is a binary variable indicating whether the user is interacting with the system, and has not served the correct drink yet, and $SP_i$ (Social Penalties) represents various social penalties, such as when the system turns its attention to another user while user $i$ is still talking to them.

The policies are encoded as functions that associate a value to each state-action pair; these so-called $Q$-values are estimates of the long-term discounted cumulative reward. Given the current state, the policy selects the action with the highest $Q$-value:

$$ \pi(s) = \arg \max_a Q(s, a) $$

Using a Monte-Carlo Control algorithm [11], the policies are optimised by running the IM against the MUSE and after each interaction sequence use the received reward signal to update the $Q$-values.
users. Each optimisation was carried out over 250k iterations, starting with an exploration rate of $\epsilon = 0.2$, discounted in every iteration with a factor 0.98. Each iteration corresponds to one session, by which we mean a complete scenario in which the two users enter the scene and try to order a drink, successfully or not. When updating the Q-function with the total reward obtained at the end of a session, a discount factor of $\gamma = 0.995$ is applied. After every 1000 iterations, the learned policy was saved and evaluated by running 2000 sessions with the MUSE, using the fully discounted exploration rate of $\epsilon = 0.001$.

![Plot](image-url)

**Fig. 3.** Results from a 2 user SSE policy optimisation in terms of average reward, showing learning curves for the best policy ($t07$) as well as the average over 10 optimisations, and the performance levels of the strategies that use random policies for one or both of the decision stages ($rnd1$, $rnd2$, and $rnd$), and of a hand-coded strategy ($hdc$).

Fig. 3 shows the training results in terms of average reward. The learning curve of the best policy found is shown ($t07$), as well the average performance over the 10 optimisations at different stages of training (avg). In addition to the learned policies, we also evaluated the system when running a random policy for one ($rnd1$ and $rnd2$) or both ($rnd$) of the action selection stages, and finally, a fully hand-crafted version of the SSE ($hdc$).

![Table](image-url)

**Table 5.** Performance of SSE strategies in terms of Success Rate (SR) and Average Reward (AR) with 95% confidence intervals (CI), evaluated over 5000 sessions.

The results indicate that the optimisation is effective and on average converges after about 60k iterations. After about 5000 iterations, the learned policies on average start to out-perform the hand-coded system. In noise-free conditions and high user patience levels (250 frames), the hand-coded system achieves a 100% success rate, which is equalled by the performance level of a policy that is optimised under these conditions. As the patience levels are reduced (to 175 frames), it becomes more difficult to hand-code an effective strategy for managing the users’ patience. The overall results are summarised in Table 5, in which we also listed an alternative hand-coded policy ($hdcNP$). In contrast to $hdc$, this policy does not include asking a second user to wait, before continuing to serve a user it was already interacting with. In this particular setting of the MUSE, the $hdc$ strategy is more successful (93% vs 87%). In an alternative setting where the impact of a pausing act by the system on the users’ patience levels is reduced (from 40 to 15 frames to be added to a user’s patience level upon receiving a pausing act), the alternative strategy $hdcNP$ is more successful (87% vs 81%). Using a policy that can be optimised, the best strategy for managing user patience is found automatically and generally outperforms the hand-coded strategies. In terms of success rate, the optimised strategy $t07$ achieves a relative improvement of 15% over $hdcNP$ and 7.5% over $hdc$.

We also did an experiment in which noise was added to the speech input by confusing the speech act types at various rates. Fig. 4 shows the performance of three strategies at confusion rates varying from 0% to 40%: the hand-coded ($hdc$) and trained ($t07$) policies from before, and in addition a policy $tra25$ that was trained at a confusion rate of 25%. The latter strategy clearly outperforms both $hdc$ and $t07$, particularly at higher confusion rates.

![Plot](image-url)

**Fig. 4.** Sensitivity of SSE strategies to noise (speech only).

6. CONCLUSION

We have presented a new framework for automatically optimising social skills in multi-user, multi-modal interactions. The main part of the framework is an Interaction Manager (IM) that processes audio-visual input on a frame-by-frame basis, maintains a model of the social state, and generates high-level communicative and non-communicative actions as well as combinations of modalities for realising them. The decision-making process features a hierarchy of two MDPs
with two policies that can be optimised using reinforcement learning.

The other part of the framework is a Multi-User Simulated Environment (MUSE) that provides a simulated audio-visual input stream, generated from the behaviour of multiple simulated users. The SE was developed to test and evaluate the IM, but also to automatically optimise action selection policies of the IM. This optimisation is based on the behaviour of the simulated users, including the reward/penalty signals they provide.

The policy optimisation results show that the method is effective and learned strategies generally outperform hand-coded strategies, depending on the settings of the simulated environment. When making the conditions of the interactions more challenging, for example by lowering the patience level of the simulated users or by adding noise to the input observations, the hand-crafted system starts to fail more frequently, and is outperformed by policies trained under these modified conditions.

In future work, for the system to be even more robust to noise, including noise in the vision input, the IM should take information about the uncertainty in the state estimation into account. The use of Partially Observable Markov Decision Processes (POMDPs) has become a well-established approach to designing interaction managers aimed at improving the robustness of systems to speech recognition errors [12]. The framework allows to explicitly represent the uncertainty arising from such errors and to use action selection policies that can be optimised in the face of this uncertainty, using reinforcement learning techniques. We plan to extend our MDP framework to a POMDP framework along the lines of this work. With regard to the MUSE, this requires further developing the error model for the audio-visual input to the IM.

In order to evaluate our IM in interaction with human users, it will be integrated into the robot bartender system, which currently has a simple rule based social state recogniser and a plan-based action selection module [13]. We will also use data collected in an evaluation of this initial system to make the data that the simulated environment generates more realistic, including the reward function that can be tuned to correlate better with user satisfaction. A system trained on this improved simulation should perform better when evaluated on real users.

7. REFERENCES


