Multi-Modal Features for Real-Time Detection of Human-Robot Interaction Categories

Ian R. Fasel
The University of Arizona
Department of Computer Science
Tucson, AZ 85721-0077
ianfasel@cs.arizona.edu

Masahiro Shiomi
Applied Telecommunications Research Institute
Kyoto, Japan
m-shiomi@atr.jp

Pilippe-Emmanuel Chadutaud
Applied Telecommunications Research Institute
Kyoto, Japan
pe-chadu@atr.jp

Takayuki Kanda
Applied Telecommunications Research Institute
Kyoto, Japan
kanda@atr.jp

Norihiro Hagita
Applied Telecommunications Research Institute
Kyoto, Japan
hagita@atr.jp

Hiroshi Ishiguro
Applied Telecommunications Research Institute
Kyoto, Japan
ishiguro@atr.jp

ABSTRACT

Social interactions unfold over time, at multiple time scales, and can be observed through multiple sensory modalities. In this paper, we propose a machine learning framework for selecting and combining low-level sensory features from different modalities to produce high-level characterizations of human-robot social interactions in real-time. We introduce a novel set of fast, multi-modal, spatio-temporal features for audio sensors, touch sensors, floor sensors, laser range sensors, and the time-series history of the robot's own behaviors. A subset of these features are automatically selected and combined using GentleBoost, an ensemble machine learning technique, allowing the robot to make an estimate of the current interaction category every 100 milliseconds. This information can then be used either by the robot to make decisions autonomously, or by a remote human-operator who can modify the robot's behavior manually (i.e., *semi-autonomous* operation [5]). We demonstrate the technique on an information-kiosk robot deployed in a busy train station, focusing on the problem of detecting interaction breakdowns (i.e., failure of the robot to engage in a good interaction). We show that despite the varied and unscripted nature of human-robot interactions in the real-world train-station setting, the robot can achieve highly accurate predictions of interaction breakdowns at the same instant human observers become aware of them.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Robotics

ICMI-MLMI'09, November 2–4, 2009, Cambridge, MA, USA.

General Terms

Algorithms, Experimentation, Human Factors

Keywords

Human-Robot Interaction, Multi-Modal Features

1. INTRODUCTION

For a robot, engaging in social interaction is much like any other control problem – it must contintually use its sensors to update its current beliefs about the state of the world and rapidly adapt its behavior accordingly. In contrast to industrial robots, the information about the world that is most salient to a *social* robot can be quite complex. Social interactions unfold over time at multiple timescales, and result in changes across multiple sensory modalities. For instance, the distances between humans and the robot at different points in time, the pattern of human touches to the robot, and the pattern of human speech relative to the sequence of robot behaviors are all useful for understanding the nature of a social interaction. Thus, while a social robot needs to make use of sensor data at relatively short time-scales to make instantaneous choices of action, it also needs to integrate information from longer time scales, including its own actions and previous beliefs, to provide higher-level *understanding* of the interactions as they unfold. This might be regarded as a form of *meta-cognition*, since the robot first uses sensors to choose instantaneous behaviors using it's underlying cognitive architecture, then combines sensor information with information about its own past behaviors in a different way to make higher-level hypotheses about it's own social behavior.

In this paper we present a technique for combining several different types of robot sensors, as well as information about the robot's own past behaviors, across multiple time-scales to produce a continually updated (every 100ms) assessment of the current social interaction. This interaction evaluation can then be used by the robot to adapt its behaviors from moment to moment, or by a remote human operator who can intervene if the robot cannot make good use of this information by itself. Our approach to information integra-

tion is inspired by recent work in vision and audio processing for real-time detection and understanding of faces, facial features, and facial expressions [20, 13, 12], and auditory scenes and emotional speech [1, 14]. We introduce a large library of simple, fast features over the "raw" robot sensors, and use GentleBoost [4] to select and combine a subset of these features into a strong classifier. Our features are defined across multiple sensory modalities, rather than only vision or only audio, and extend over multiple time-windows. Because we do not know *a priori* what modalities are most useful for understanding social interaction, or what time-scales, we construct the features in such a way that the learning algorithm can be agnostic about the underlying sensor types and time-scales, and can thus automatically discover which types of sensors and which temporal windows are most useful by virtue of which features are selected for solving the problem at hand.

We test our approach on a social robot located in a train station designed to provide humans with information about the station and surroundings through social communication. The robot behaves fully autonomously, and engages in unscripted interactions with passing humans. In such a noisy, uncontrolled setting as a train station, it is not uncommon for a robot to make mistakes during interactions with humans (due to e.g., failures in the speech-to-text processing system, or faces that are not detected by the vision system, etc.). Because the robot itself does not know when it has made a mistake, human-robot interactions can rapidly deteriorate. Thus a highly useful initial goal for interaction categorization, which we pursue in this paper, is to give the robot the ability to determine at each instant if it is currently engaging in a successful interaction with the human. We use this determination to enhance *semi-autonomous* operation, in which a covert remote-operator can switch to remote-control mode if he determines that the robot has encountered a situation that it cannot handle by itself. In [5], semi-autonomous operation was studied extensively and it was shown that one human could control several robots simultaneously and recover most problematic human-robot interactions transparently. However one of the greatest difficulties was that the human operator had to determine by themselves whether or not the robot needed help. Our method allows the robot to automatically signal to the operator when it thinks it needs help, reducing the cognitive load on the operator and improving the quality of human-robot interactions.

In addition to semi-autonomous operation, the ability to automatically determine interaction categories, and in particular the *goodness* of interaction we focus on here, has a wide range of other possible uses. For instance, by treating the *good* vs. *bad* classification as an intrinsic reward signal, a robot can employ a reinforcement learning algorithm to autonomously modify its interaction policy to maximize the expected long-term number of *good* interactions with humans.

In the remainder of the paper we briefly discuss previous work on human-robot interaction classification, describe our new set of multi-modal features, and then describe our machine learning approach for selecting and combining a subset of features. We then present experiments on synthetic data, a laboratory setting, and the real-world train station setting. We then discuss the results in detail and then finally describe future improvements and future applications.

2. CLASSIFYING HUMAN-ROBOT INTERACTION

Evaluating the character of social interactions tends to be intuitive for humans, but what features are actually used by humans to make these judgements? In [19], researchers introduced the humanoid robot QRIO into a preschool classroom and asked human coders to watch videos and score in real-time the "goodness of interaction" on a scale of 1 to 5. Despite the fact that coders were not given any explicit instructions about exactly what "good interaction" should entail, multiple human coders had high agreement on the same videos. *Post hoc* analysis of videos later showed that the touching behavior by the children (in particular, touching of the robot's arms) could predict with high accuracy the human coders' judgements (correlation of 0.79). This led the authors to speculate that modern machine learning methods might be used to enable robots to autonomously classify the goodness of interactions using only their touch sensors. Another important lesson was that human observers did not have a good *a priori* notion of what sensory features would predict interaction success. Thus, it might also make sense for a machine learning algorithm to simultaneously select the most useful sensory features as it is learning how to best combine them.

A variety of methods to automatically analyze and categorize human-robot interactions have been used in previous work [17, 8, 16]. Most of these have been offline methods, i.e., analysis is performed after the interactions have finished. However a few systems have recently been developed for online use. In [3], Francois et al. used self-organizing maps to preprocess touch sensor data for classifying autistic childrens' interaction styles with an AIBO robot dog into "strong" versus "gentle" categories, allowing it to adapt its behavior to the interaction style. Hammer et al. [7] developed a neural-network based method for classifying play behavior on a specially designed touch-sensitive playground, and could categorize children's behaviors into eight different categories very accurately. This was then used to adapt the playground to encourage more play [7]. Finally, Ruvolo et al. [14, 15] used audio to classify, in real-time, the "mood" of a classroom into "crying" vs. "singing / dancing" vs. "everything else" for a child-care robot.

In this paper, we explore an approach to real-time classification of human-robot interactions that is similar to the approach of [14, 7], but we use multiple sensory modalities simultaneously instead of audio or touch alone. Our method has two phases: a training phase and a run-time phase. For training, high-fidelity recordings of all sensor data during a series of human-robot interactions are recorded and stored in a database. A human coder then watches videos of the interaction sequences and marks the instant in time that interaction appears to be going bad (otherwise the entire sequence is labeled *good*). These labeled sequences are then used to train a classifier. At run-time, a set of running statistics are updated every 100ms, and a new *good* vs. *bad* prediction is computed based on the most recent N seconds of sensor data. The outline of the system is shown in Figure 1.

3. MULTI-MODAL FEATURES FOR INTERACTION CLASSIFICATION

For these experiments we use the humanoid robot RoboVie-II [9], which has a wide array of sensors, both on its own

Train-Time algorithm

Given dataset of interaction sequences and human labels:

1) Scale, align, and compute cumulative-sums on example interaction data (described below).

2) Use GentleBoost [4] to choose a small set of features from all features across all sensor types, and combine selected features into a single classifier.

Run-Time algorithm (repeat every 100 miliseconds):

1) Update cumulative-sum vectors for all sensors.

2) Recompute feature responses on updated sensors.

3) Classify current moment as *good* or bad interaction using learned classifier.

Figure 1: Outline of the learning and runtime algorithms. At runtime, feature updates require only a handful of lookup, add, and multiply operations.

body and within the environment (see Figure 5). In our experiments we used five types of sensors: touch, sound level, laser range finder, pressure sensitive floor, and robot behaviors (treated by our algorithm as a sensor). These sensors are asynchronous and run at different sampling rates. The time between samples for any given sensor is not guaranteed to be constant. This poses a challenge to supervised machine learning algorithms, which typically need training vectors to be aligned with respect to the label and of fixed size. Therefore, our first challenge is to find a representation which (a) converts asynchronous sensor readings into fixed-size vectors, and (b) does so in a way that can be accomplished with minimal computation for real-time operation. After describing the general way of representing sensors suitable for the learning algorithm, we will describe the individual sensors and the features defined over them.

3.1 Time-scaling

The first step for putting different sensors into a common representation is to synchronize them according to a single reference point on a common clock. We use a simple re-sampling technique for this, essentially low-pass filtering samples which arrive more frequently than the base rate, and interpolating samples which arrive less frequently than the base rate. The result of this procedure is a vector Q_s for each sensor s, where the first element represents the most recent sensor data, and each subsequent element represents an estimate of the sensor value at fixed intervals of time. Our re-sampling method is as follows:

1. For each sensor s, set a sampling interval k.
2. For each sensor s, initialize random-access vector Q_s of size `max_length` with all zeros.
3. Repeat every 100 milliseconds:
   - For each sensor s:
     (a) If new samples arrived since last update, insert most recent sample into the front of Q_s. Otherwise, copy the first value in Q_s and insert it into the front of Q_s.
     (b) Delete the last element of Q_s.

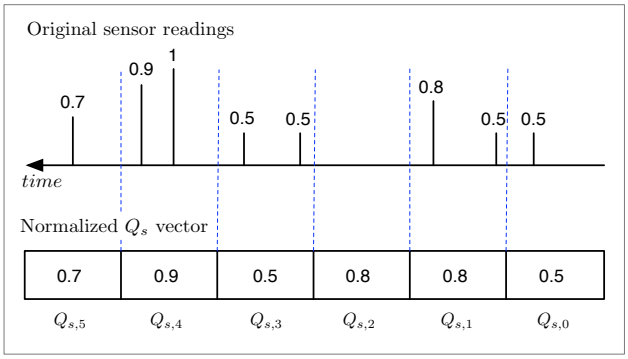

Figure 2: Converting asynchronous sensor readings into fixed length vectors. The first element $Q_{s,0}$ is always the most recent value of the sensor. Here we show time from right to left.

The procedure is illustrated in Figure 2. Note that it is not necessary for each sensor to use the same sampling interval. In our experiments we use 100ms intervals for most sensors, and 500ms intervals for the others.

3.2 Extracting training data from labeled sequences

With the representation described above, it is simple at runtime to select the most recent N seconds of data across all sensors and present it to a classifier which takes as input a set of fixed-size vectors. For training the classifier, we must now align all of the training sequences with respect to a detection window defined by the time each interaction was labeled as *good* or *bad*, and the number of seconds that we wish to include before and after the label when determining the interaction category. The before and after intervals are tunable parameters chosen manually or through cross-validation. In Section 6 we discuss the effects of different before and after intervals.

To illustrate the alignment of training data, Figure 3 shows an example in which the detection window is fixed to include one second after the label and three seconds before the label. The figure shows a six second sequence which the human coder has labeled bad at the fourth second. In this example, the first two sensors have been time-scaled with a sampling interval of .5 seconds, and the third with a sampling interval of 1 second. For the first two sensors, the detection window includes elements 2 through 9, while for the third sensor, elements 1 through 4 are included. Trained with samples extracted using this detection window, the resulting classifier output at runtime can be interpreted as meaning "Based on the last four seconds of data, the interaction one-second ago is *bad* / *good*".

3.3 Features

We can now describe the features we use for classification. Our features are very similar to the box-filters used in [20] for detecting objects in images, and the spatio-temporal box filters described in [14] for classifying audio scenes. In our case, a feature has three parameters: an index s indicating which sensor it refers to, a start time i, and an end time j. A feature response is then computed by taking the sum of values in sensor vector Q_s between the feature's start and

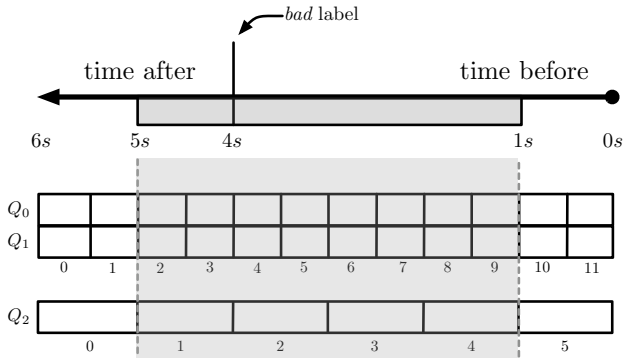

Figure 3: Extracting a 4 second *detection window* from a 6 second sequence labeled *bad* at the 4th second. In this example the detection window is defined as three seconds before and one second after the label. Note that time is depicted right to left.

end time. The start time and end time are always computed with respect to the start of the detector window. As shown in Figure 3.3, these features can be computed efficiently if, for each Q_s vector, a cumulative-sum vector C_s is stored instead. The sum of values in Q_s between i and j can be quickly computed by subtracting $C_{s,j}$ from $C_{s,i+1}$.

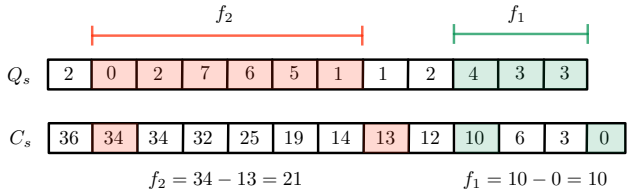

Figure 4: Two features, f_1 and f_2, defined on the same sensor s. Using the cumulative sum representation, the feature response can be computed efficiently. Above is an example (time-scaled) sensor vector Q_s. Below is the corresponding cumulative sum vector C_s.

The cumulative sum representation of sensors is especially convenient for real-time updates. When new data arrives on sensor s, the cumulative sum vector can be updated by simply adding the incoming value and the value in the first element of C_s, then inserting the result into the beginning of C_s, and then deleting the last entry of C_s. Any features needed for classification can then be computed with two lookups and one subtraction.[1]

3.4 Sensor types

The robot had the following sensors:

Sound level: Two sound level sensors give an instantaneous reading of the sound intensity on two external microphones every 100ms.

[1]In order to avoid precision errors, C_s should be implemented using a double-buffer that is occasionally swapped. Otherwise the values in C_s will become so large relative to the values in Q_s that the difference is smaller than the floating-point precision.

Tactile sensors: 16 tactile sensors on the robot's body (arms, shoulders and chest) give a pressure reading at different locations of the robot's body every 10 ms.

Laser range finder: An Hokuyo URG-04LX infrared laser range finder, affixed to the robot, returns 683 distance estimates spaced evenly over a 240 degree range, with a radius of up to four meters, sampled about every 100ms. We binned the distance readings into $10^o \times 500$cm regions, resulting in 192 regions, and treat each region as a separate distance "sensor". The value for a distance sensor is computed as the total number of distance readings in that region.

Floor sensor: We installed a specially equipped floor which returns pressure readings within a fixed region around the robot. Similar to the laser range finder, this sensor was divided into 20×20 regions, resulting in 400 floor "sensors". Each floor "sensor" is the sum of the pressure readings in that region.

Behavior: In addition to the real sensors, we also treat the robot's own behavior as a type of sensor. The robot has over 200 core behaviors, each which can last between a few tenths of a second to several seconds. These behaviors were grouped into 6 behavior types: greeting-starting, greeting-leaving, guiding, talking, free-play, and idling. At any moment, the robot is running one of these behaviors. We therefore create six behavior "sensors", one for each behavior type. At each sampling interval, we set the sensor for a behavior type to 1 if the robot is currently executing a behavior of that type, and 0 otherwise.

We used a sampling interval of 100ms for all of these sensors, except for behavior we set the sampling interval to 500ms. This results in 616 sensors. For a 15 second detection window, there are about 5000 possible features for each sensor. Therefore for a 15 second detection window there are about 3 million possible features defined over all five sensor types.

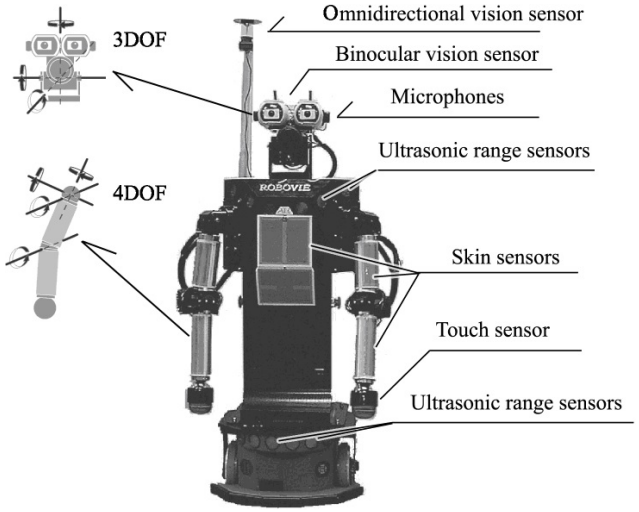

Figure 5: The RoboVie-II robot used in Experiments 2 and 3.

4. LEARNING

We use GentleBoost [4] to construct a strong classifier that combines a subset of all possible features. GentleBoost is a popular method for sequential maximum likelihood es-

timation and feature selection. At each round of boosting, a transfer function, or "tuning curve", is constructed for each feature which maps feature response to a real number in $[-1, +1]$. Each tuning curve is optimized using nonparametric regression methods to minimize a weighted exponential loss (see [13] for details on this procedure). At each round of learning, the feature plus tuning curve that yields the best improvement in the total loss is added into the ensemble, and the process repeats for a fixed number of rounds, or until performance no longer improves on a holdout set. In this way, GentleBoost simultaneously selects a subset of good features and builds a classifier out of them.

At each round of boosting, it is necessary to recompute the optimal tuning curve for each feature under consideration for being added to the ensemble, because the data samples are re-weighted, so old tuning-curves can't be re-used. Since brute-force search through all 3×10^6 possible features would be very expensive, we employ a search procedure known as Tabu Search [6] to speed up the process. First, a random set of n features are selected and evaluated on the training set, and are used to initialize the "tabu list" of features already evaluated in this round. The top $k \leq n$ of these features are then used as the starting points for a series of local searches.

From each starting filter, a set of new candidate features are generated by replicating the filter and slightly modifying its start and end time. If any of these features are not already in the tabu list, they are evaluated and then added to the list. If the best feature from this set improves the loss, it is retained and the local search is repeated until a local optimum is reached. After the local search has been completed for each of the initial k best features, the feature and tuning curve which achieved the greatest reduction in the loss function is added into the ensemble classifier.

The amount of time needed to train a classifier scales linearly with the number of examples. On a 2GHz laptop computer it takes about 20 minutes to train a 100 feature classifier on the datasets described in the Experiments section, using code written in Matlab.

5. EXPERIMENTS

We conducted three experiments with this system. The first two experiments can be regarded as pilot experiments. The third experiment is to test if the system can evaluate the goodness of interactions when deployed in the real-world.

Experiment 1:

The first experiment was a "toy problem", meant to test the general validity of the approach. We generated 200 sequences of synthetic interactions, in which each sensor type was generated with different parameters (e.g., frequency, location, and duration of touch) depending on whether it was a *good* or *bad* example. The synthetic data was designed to share some similarities to real data, but still be simple to generate. For example, to generate synthetic loudness data, we generated noisy square waves of slightly different frequency and amplitude depending on if they were a *good* or *bad* example.

Experiment 2:

This experiment was conducted at a laboratory using the RoboVie-II robot. The robot was set up to provide information and entertainment to humans who engage it. We placed floor sensors in the center of a 2×2 meter floor area, then placed the robot on the floor sensors. The loudness sensor and laser range finder were attached to the robot. Five participants were recruited to interact with the robot under varying conditions. Four kinds of interactions were performed:

1. The robot greets the participant, then the participant greets the robot (*good* interaction).
2. The robot greets the participant, but the participant ignores the robot (*bad* interaction).
3. The participant greets the robot, then the robot greets the participant (*good* interaction).
4. The participant greets the robot, but the robot ignores the participant (*bad* interaction).

Each participant performed each interaction five times, resulting in 100 example sequences. A human coder then labeled the instant that a *bad* interaction occurred in the *bad* interaction conditions. The goal was to be able to classify the instant at time that the human labeled an interaction as *bad* for the *bad* interactions.

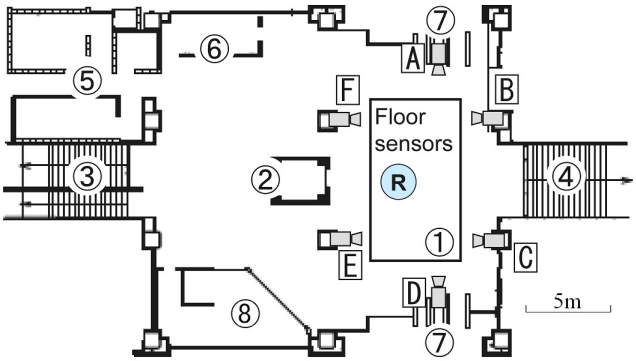

Figure 6: The train station setup in Experiment 3

Experiment 3:

This experiment was conducted at a terminal station for a railway line that connects residential districts with the city center. Station users are mainly commuters and students, however families often visited the station on weekends to see the robot. There are four to seven trains per hour. Most users go down the stairs from the platform after they exit a train. We introduced the RoboVie-II robot into the train station to act as an interactive information-kiosk. We set the robot and the sensors in front of the right stairway and informed visitors that the robot is a test-bed, with the capability of providing guidance about routes to users. We placed floor sensors in the center of a 4×8 meter floor area. The robot was placed on the floor sensors, and it also could move around. The loudness sensor and laser range finder were attached to the robot. The users could freely interact with the robot. RoboVie-II has a large number of possible behaviors, chosen using a sensor-driven behavior scheme described in detail in [8, 9]. In this experiment, the robot was set up so that a normal sequence of interactions would be:

1. Approach a person,
2. Greet and shake hands with the person,
3. Provide information (e.g., tell them about the new shopping mall nearby),
4. Guide the person if they want to go somewhere,
5. Play a game with the person if they don't want to go somewhere.

These behaviors are described in greater detail in [18]. For this experiment, we gathered 74 example sequences which included greetings by either the robot or the human. A human coder marked the moment when a *bad* interaction occurred. In those video sequences where no *bad* interaction occurred, we labeled a random time as *good*.

6. RESULTS AND DISCUSSION

For each experiment, we benchmark the system by training on a subset of the data, then testing on the remaining data. When testing, we measured the area under the ROC (AUC), which can also be interpreted as the correct classification rate on a two-alternative-forced-choice task (2AFC) [2] and does not depend on choosing a final threshold on the classifier's output.

The AUC scores we report are five-fold cross-validation results. For each experiment, we randomly divided the examples into five sets, roughly equal in size. Then for each set, we train on four of the subsets and test on the fifth. The graphs below show average AUC scores over all folds, with error bars representing the standard deviation of the mean. For each experiment, we trained for 100 rounds of boosting (thus selecting 100 features). We found training more features did not appreciably improve or decrease performance.

Overall we achieved excellent results on all datasets. First, in the toy data experiment, we verified that for each sensor type, if a classifier was trained using only sensors of that type, we could learn a detector that achieves very high (larger than 0.98) AUC within three or four features. Using all sensors combined also gave excellent performance. These results were mainly useful to demonstrate that the overall approach did not have obvious implementation errors and that our general approach was viable.

On the real robot datasets, we obtained very convincing results. In Figure 7, we show the result of trying different *before* and *after* values for the detection window for experiment 2. These show that in general, allowing the detection window to include more time prior to the *good* / *bad* tag is beneficial, although improvements level off after about 5 seconds. Including time after the *good* / *bad* tag beyond a few seconds does not appear to improve performance. Overall, the best performance on experiment was AUC of 0.94 with a detection window of 12 seconds after and 0 seconds before the tag.

In experiment 3, we achieved an AUC of 0.96 when using a detection window of three seconds before and three seconds after the tag. This means that in the train station, the robot could be aware that something is wrong within about 3 seconds from the time an interaction begins to break-down. If we used zero seconds after the tag, then best performance was still 0.94. Thus 94% of the time, the robot can determine that an interaction failure has occurred at the same instant that a human would make this determination.

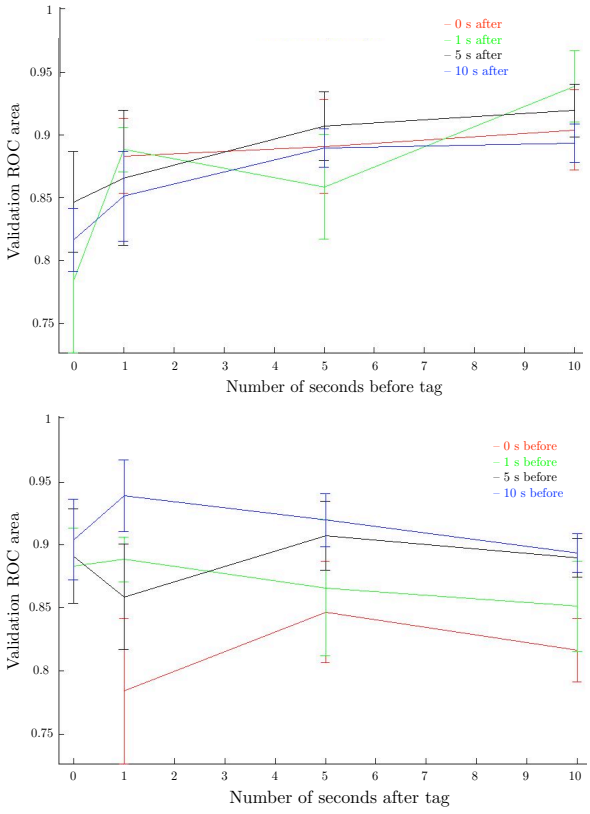

Figure 7: Top: Performance generally increases as the detection window uses more history. Bottom: Performance is improved slightly by including a few seconds after the failure label, but more time does not improve accuracy.

In Figure 8, we examine performance when only one sensor type is used at a time. These curves are shown for Experiment 2, using a detection window of zero seconds after the tag, and varying the time before the tag from 0 to 30 seconds. It is interesting to see that by themselves, the loudness and behavior sensors can achieve around 90% accuracy, using a detection window with about 5-15 seconds of history.

We were initially surprised that the touch sensors were not able to provide good results, however more careful observation of the videos revealed that subjects rarely touched the robot in either condition. We speculate that this may be due to the fact that subjects were college students interacting with a 4-foot tall robot, whereas many previous studies such as [19, 3], which showed touching behavior to be a *good* predictor of interaction categories, were focused on young children with smaller robots. Overall, the laser range finder and floor sensors were able to give decent performance, but were not as reliable as loudness or behavior. Because we tested them with the toy problem, we don't believe these features were fundamentally wrong. Rather, we believe this result merely shows that they are not reliable enough in real-world settings to give us high accuracy when used alone. When we used all sensor types combined, we saw a boost in performance to 0.945, showing that combining the sensors had a positive effect, achieving performance greater than any one sensor type alone.

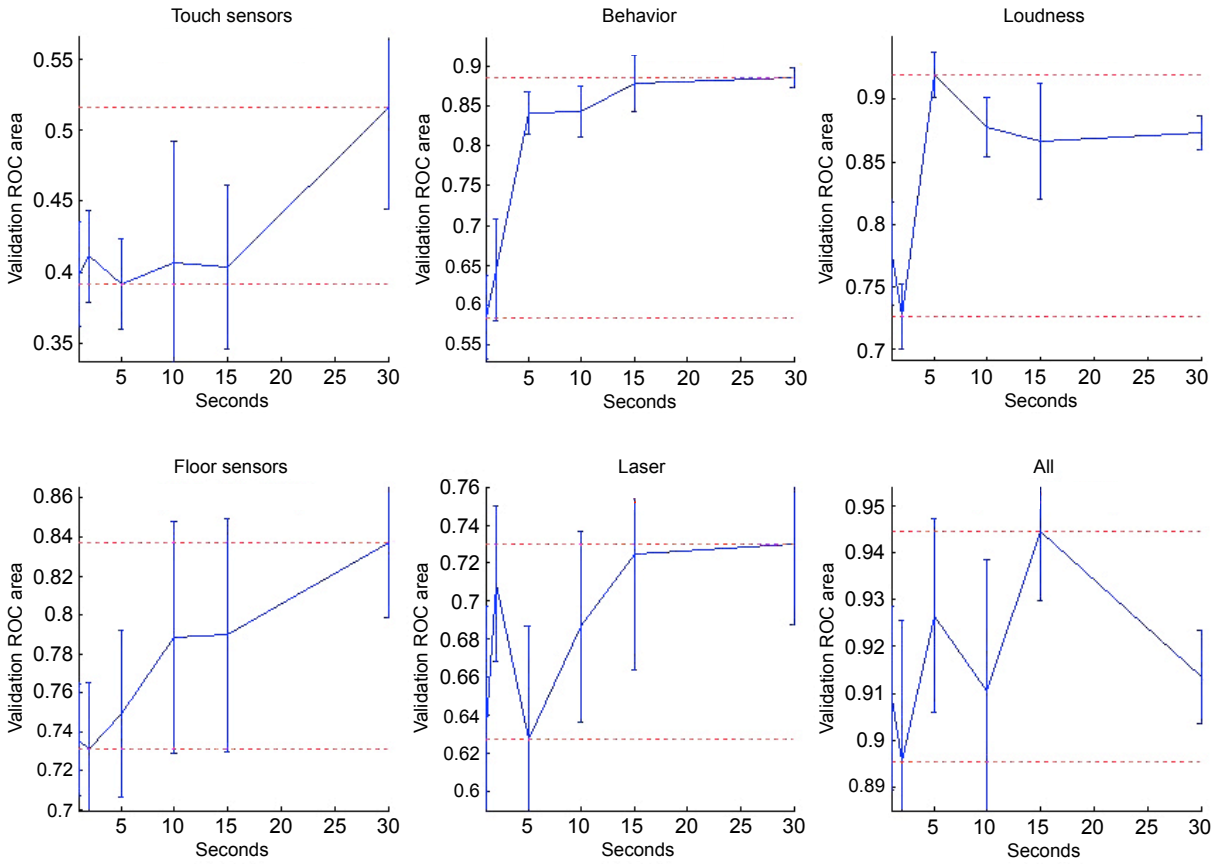

Figure 8: Area under the ROC (AUC) curves for each individual sensor as time before is increased from zero to 30 seconds. Error bars show standard error. Detectors were trained with 100 features. Loudness and Behavior sensors each perform quite well when used independently. Other sensors do less well – indeed the touch sensors seem to carry almost no information at all. This is due to the fact that subjects rarely touched the robot in the experiment. Combining features from all sensor types results in the highest overall performance.

For Experiments 2 and 3, the features that were selected most often were the loudness and behavior sensors, although floor and laser sensors were chosen as well. This is an interesting result, as we had expected the laser range finder to provide better prediction ability. From the videos it appears that while the laser range sensor could often predict that if a human is not in front of the robot then there is not a *good* interaction going on, on the other hand, it is often the case that a person is standing right in front of the robot when a *bad* interaction happens.

7. CONCLUSIONS AND FUTURE WORK

This paper presents a novel approach to combining information from multiple modalities in order to give real-time (every 100ms) predictions of the current interaction category. We introduced a new set of features across several sensory modalities that can be computed extremely fast (just a few operations per feature), and which can be combined together with a machine learning approach to achieve higher accuracy than using any one of the sensor types alone. The learning algorithm automatically discovered that temporal features of the loudness and robot behavior time-series signal were the most useful in predicting whether a greeting interaction was *good* or bad, and the best predictions could be made using information from the 6-12 seconds preceding the interaction breakdown. This is somewhat in contrast to the work of [19, 3], which showed touching behavior to be the best predictor of *good* vs. *bad* interactions.

While we have achieved good results on this early work, we believe we can get even better performance by improving the features. First, vision and audio applications that used similar summation-type features (e.g., [20, 14]) included additional operations after computing the sums over subwindows, such as taking differences between adjacent subwindows (facilitating detection of local contrasts) and repeating features in time and computing summary statistics over repetitions (thus facilitating detection of periodic events). It seems likely that we could also benefit from using features that capture periodic regularities, since good interactions often involve periodic "turn-taking". In addition, we would like to use more sophisticated features from vision and audio, although for these sensor types we would likely not use the raw features but the output of some other system – for instance, by treating the results of an automatic facial action unit coding system (as in [11]) as "sensors" and constructing temporal features over them.

A future goal for this work is to categorize many different aspects of human-robot interaction, such as the five phases described in Experiment 3 (approach, greet, provide information, guide, play). There is ample evidence this method will perform as well for these cases as for the greeting phase we studied in this paper. First, the greeting phase already contains a rich variety of human-robot activities, including turn-taking, and the other interaction categories contain the same kinds of behaviors. Extending the features to explicitly account for periodic variations as described above will also enhance the ability to recognize turn-taking features of cooperative interaction. Although the current method is binary, we can produce an accurate N-way classifier using the procedures described in [10, 12, 14, 15] for face and speech emotion classification. In our case, we will train binary detectors for all *good* vs. *bad* detectors for each category, as well as for all *good* vs. *bad* detectors for all possible *one* vs. *all other* category splits, and then combine these various detectors using multinomial logistic ridge-regression.

What constitutes *good* vs. *bad* interactions is an inherently ambiguous concept, which is why we did not attempt to coach the human labelers about the definition, but allowed them to make their own interpretation. Much like recognizing faces, even though we don't consciously know the rules our brain is using to recognize interaction categories, people do indeed "know it when they see it", as demonstrated in [19]. We believe that this makes the problem even more suited to feature-selection / data-driven approach than face-detection appeared to be in the early days of computer vision. Nevertheless, the problem of automatic understanding of human-robot interaction is amenable to many different approaches and we believe there is considerable research yet to be done in this area.

8. REFERENCES

[1] S. Chu, S. Narayanan, C.-C. J. Kuo, and M. J. Matarić. Where am I? Scene recognition for mobile robots using audio features. In *IEEE International Conference on Multimedia & Expo (ICME)*, 2006.

[2] C. Cortes and M. Mohri. Auc optimization vs. error rate minimization. In S. Thrun, L. Saul, and B. Scholkpf, editors, *Advances in Neural Information Processing Systems, 16*, Cambridge, MA, USA, 2004. MIT Press.

[3] D. Francois, D. Polani, and K. Dautenhahn. On-line behaviour classification and adaptation to human-robot interaction styles. In *Proc. 2nd ACM/IEEE International Conference on Human-Robot Interaction (HRI07)*, pages 295–302, Washington DC, USA, March 9-11 2007.

[4] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 28(2):337–374, 2000.

[5] D. F. Glas, T. Kanda, H. Ishiguro, and N. Hagita. Simultaneous teleoperation of multiple social robots. In *Proceedings of the 3rd ACM/IEEE international conference on human-robot interaction*, Amsterdam, The Netherlands, 2007.

[6] F. W. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.

[7] F. Hammer, A. Derakhshan, Y. Demazeau, and H. H. Lund. A multi-agent approach to social human behaviour in children's play. In *IAT '06: Proceedings of the IEEE/WIC/ACM international conference on Intelligent Agent Technology*, pages 403–406, Washington, DC, USA, 2006. IEEE Computer Society.

[8] T. Kanda, H. Ishiguro, M. Imai, and T. Ono. Body movement analysis of human-robot interaction. In *In Proc. Int. Joint Conf. on Artificial Intel ligence (IJCAI)*, pages 177–182, 2003.

[9] T. Kanda, H. Ishiguro, T. Ono, M. Imai, and R. Nakatsu. Development and evaluation of an interactive humanoid robot. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2002.

[10] G. Littlewort, M. Bartlett, I. Fasel, J. Susskind, and J. R. Movellan. An automatic system for measuring facial expression in video. *Computer Vision and Image Understanding, Special Issue on Face Processing in Video*, 24(6):615–625, 2006.

[11] G. Littlewort, M. Bartlett, I. Fasel, J. Susskind, and J. R. Movellan. An automatic system for measuring facial expression in video. *Image and Vision Computing.*, in press.

[12] G. Littlewort, M. S. Bartlett, C. J, I. Fasel, T. Kanda, H. Ishiguro, and J. R. Movellan. Towards social robots: Automatic evaluation of human-robot interaction by face detection and expression classification. In S. Thrun, L. Saul, and B. Schoelkopf, editors, *Advances in neural information processing systems*, volume 16, pages 1563–1570. MIT Press, Cambridge, MA, 2004.

[13] J. R. Movellan and I. R. Fasel. A generative framework for real time object detection and classification. *Computer Vision and Image Understanding*, 2005.

[14] P. Ruvolo, I. R. Fasel, and J. R. Movellan. Auditory mood detection for social and educational robots. In *ICRA*, pages 3551–3556, 2008.

[15] P. Ruvolo and J. R. Movellan. Automatic cry detection in early childhood education settings. *Proceedings of ICDL*, pages 204–208, 2008.

[16] T. Salter, F. Michaud, K. Dautenhahn, D. Létourneau, and S. Caron. Recognizing interaction from a robot's perspective. In *Proc. 14th IEEE Int. Workshop on Robot and Human (RO-MAN)*, pages 178–183, 2005.

[17] B. Scassellati. Quantitative metrics of social response for autism diagnosis. In *Proc. 14th IEEE Int. Workshop on Robot and Human Interactive Communication (RO-MAN)*, 2005.

[18] M. Shiomi, D. Sakamoto, T. Kanda, C. T. Ishi, H. Ishiguro, and N. Hagita. A semi-autonomous communication robot: a field trial at a train station. In *HRI*, pages 303–310, 2008.

[19] F. Tanaka, A. Cicourel, and J. R. Movellan. Socialization between toddlers and robots at an early childhood education center. *Proceedings of the National Academy of Science*, 194(46):17954 x17958, 2007.

[20] P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision*, 2002.