Towards a Robust Interactive and Learning Social Robot

Robotics Track

Michiel de Jong
Carnegie Mellon University
Pittsburgh, USA
mdejong@andrew.cmu.edu

Kevin Zhang
Carnegie Mellon University
Pittsburgh, USA
klz1@andrew.cmu.edu

Aaron M. Roth
Carnegie Mellon University
Pittsburgh, USA
aaronr1@andrew.cmu.edu

Travers Rhodes
Carnegie Mellon University
Pittsburgh, USA
traversr@andrew.cmu.edu

Robin Schmucker
Karlsruhe Institute of Technology
Karlsruhe, Germany
robin.s@online.de

Chenghui Zhou
Carnegie Mellon University
Pittsburgh, USA
chenghuz@andrew.cmu.edu

Sofia Ferreira
Instituto Superior Técnico
Lisbon, Portugal
asofiamferreira@hotmail.com

João Cartucho
Instituto Superior Técnico
Lisbon, Portugal
joaocartuchoo@gmail.com

Manuela Veloso
Carnegie Mellon University
Pittsburgh, USA
mmv@cs.cmu.edu

ABSTRACT

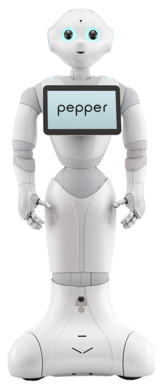
Pepper is a humanoid robot, specifically designed for social interaction, that has been deployed in a variety of public environments. A programmable version of Pepper is also available, enabling our focused research on perception and behavior robustness and capabilities of an interactive social robot. We address Pepper perception by integrating state-of-the-art vision and speech recognition systems and experimentally analyzing their effectiveness. As we recognize limitations of the individual perceptual modalities, we introduce a multi-modality approach to increase the robustness of human social interaction with the robot. We combine vision, gesture, speech, and input from an onboard tablet, a remote mobile phone, and external microphones. Our approach includes the proactive seeking of input from a different modality, adding robustness to the failures of the separate components. We also introduce a learning algorithm to improve communication capabilities over time, updating speech recognition through social interactions. Finally, we realize the rich robot body-sensory data and introduce both a nearest-neighbor and a deep learning approach to enable Pepper to classify and speak up a variety of its own body motions. We view the contributions of our work to be relevant both to Pepper specifically and to other general social robots.

KEYWORDS

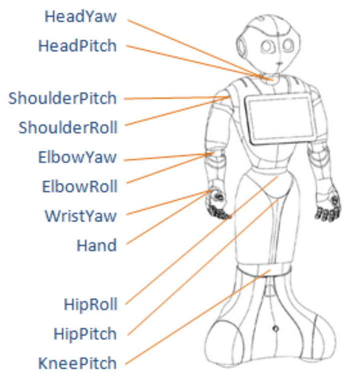
Human-Robot Interaction; Robot Autonomy; Robot Machine Learning; Pepper; Service Robot; Social Robot

ACM Reference Format:
Michiel de Jong, Kevin Zhang, Aaron M. Roth, Travers Rhodes, Robin Schmucker, Chenghui Zhou, Sofia Ferreira, João Cartucho, and Manuela Veloso. 2018. Towards a Robust Interactive and Learning Social Robot. In *Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), Stockholm, Sweden, July 10–15, 2018,* IFAAMAS, 9 pages.

Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. André, S. Koenig (eds.), July 10–15, 2018, Stockholm, Sweden. © 2018 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.



(a) Pepper the robot



(b) Pepper's joints labeled

Figure 1: Pepper. Photos: SoftBank/Aldebaran Robotics

1 INTRODUCTION

The development of service robots that can aid humans is an active area of study [2, 6, 15, 27, 28, 30]. As these robots are increasingly deployed in real world situations [8, 17, 24], it is critical that the robots can robustly interact with the people they encounter.

In this work, we focus on the social interaction capabilities of service robots. We leverage the Pepper robot platform (Figure 1a), a service robot developed by SoftBank/Aldebaran Robotics and designed for social interaction [31]. SoftBank Robotics provides default perception and interaction capabilities through its proprietary NAOqi framework. We extend those capabilities to facilitate more flexible and robust human-robot interaction. The techniques we use to extend Pepper's capabilities could be applied to any robot.

To strengthen Pepper's perception capabilities, we extend the inbuilt software with state-of-the-art external perception libraries. For vision, we use the OpenPose human pose detection package and the You Only Look Once (YOLO) and Faster R-CNN object recognition packages [3, 19, 21]. For speech, we combine NAOqi's speech recognition with the Google Cloud Speech platform to improve accuracy and allow for general speech input [18, 23].

We empirically evaluate these systems on Pepper and characterize their strengths and weaknesses. We analyze the performance of YOLO and Faster R-CNN networks on objects that do and do not map intuitively to categories in the COCO label set, to simulate real-world scenarios where the robot is asked to identify classes of objects on which it has not been trained. We propose a learning algorithm where Pepper associates novel objects with the combination of class labels they most resemble.

For the speech modality, we compare cloud-based recognition with the inbuilt speech software. Google Cloud Speech is capable of more general speech recognition, but requires a stable internet connection and incurs additional processing time. We combine the cloud-based and inbuilt speech systems and use results from the latter when cloud-based speech is unreliable or slow to process. We train a semantic parser to map recognitions to robot commands, and use parsed recognitions from Google Speech to improve the inbuilt speech recognition over time.

Even with these improvements, we recognize that individual perceptual modalities have limitations. To improve the robustness of human-robot interaction, we add additional input modalities in the form of an interface on the onboard tablet, a phone application, and external microphones. Pepper monitors input reliability to estimate the likelihood of error of the input from that modality and proactively seeks input from different modalities if one modality appears to be of low quality. In this manner, we improve the robustness of the overall social interaction.

For successful interaction, Pepper should not only sense and understand human input, but also be able to verbalize its own experience. In a step towards this goal, Pepper learns to classify motions it is undertaking in order to articulate them to the user.

2 RELATED WORK

We identify four primary research directions relevant to our approach: the study of social service robots, the improvement and analysis of the effectiveness of individual input modalities, the improvement of robustness through the combination of inputs across modalities, and the classification and verbalization of robot actions.

A significant number of research groups and firms have worked on developing autonomous service robots, leading to robot platforms such as the CoBots [30], the PR2 [2], and many others. RoboCup@Home is a competition that focuses exclusively on autonomous service robots and features robot platforms from many different participating universities [6, 15, 27, 28]. Starting in 2017, RoboCup@Home operates a league exclusively for the Pepper robot, focusing on social interaction [22].

With regard to individual modalities, large bodies of work from computer vision and speech recognition focus on the development of increasingly accurate pose detection, object detection and speech recognition algorithms. We view our contribution as being the integration of these state-of-the-art algorithms into a robotic setting, evaluating their performance in that setting, and improving the robustness of the algorithms to the unique set of challenges generated by running those algorithms on a robot operating in the wild.

At the 2017 RoboCup@Home competition, the University of Amsterdam team (UvA@Home) used Google Cloud Speech for speech recognition on Pepper [6]. As they note, one disadvantage of the Google platform is that it is not possible to constrain the search space of possible utterances. We overcome this limitation by ensembling the cloud-based speech with the constrained inbuilt speech software and by using results from cloud-based speech to inform the appropriate constraints.

Previous work has investigated using alternate inputs to control robots. Stückler et al. describe an application for a handheld device that can extensively control a service robot [27]. Our work with Pepper is distinct in that Pepper has a built-in tablet, naturally allowing for screen-based user interaction.

A variety of works provide evidence that combining input from different modalities can improve the quality of input processing. Pinto et al. combine vision stimuli together with tactile and manipulation stimuli in a deep learning framework to improve a robot's object classification capability [16]. A previous RoboCup inspired work combined audio and visual input to localize objects on the Aibo robot platform [4].

Oviatt describes how users will naturally switch to a different input modality when they encounter interpretation errors with their initial modality [12]. We build upon this work by having the robot monitor input methods and actively suggest an alternative input method when the robot perceives the user's attempted input method to be unreliable. Our approach thus shifts some of the burden of monitoring communication reliability away from the human agent.

Finally, there has been work on the verbalization of robot experience [13, 25] and classification of robot actions [29] in various domains. We draw from both literatures in that we classify the robot's own kinematic movement to better communicate its actions.

3 VISION

As a social robot, Pepper will be faced with a variety of interactions in diverse environments. Processing and analyzing visual inputs to understand these environments is critical for its successful operation. We leverage state-of-the-art human pose and object detectors in order to provide Pepper with more and better information about its environment, resulting in a broader range of visual stimuli to which Pepper can respond.

3.1 Human Recognition

The inbuilt NAOqi operating system by Aldebaran provides access to algorithms for a variety of human-related vision tasks, such as human detection and tracking, face recognition, and gaze detection.

However, some tasks require more detail than the inbuilt algorithms provide (for example, detecting whether a person is speaking or in what direction they are pointing). We incorporate the OpenPose real-time pose detection library, which provides the pixel locations of select body keypoints for each person in an image [3].

Our architecture consists of a master ROS node on Pepper and two ROS nodes on an external server. One external node processes images from Pepper's front camera through OpenPose's vision pipeline and publishes a list of humans and body keypoints. The other external ROS node examines the body points from each person

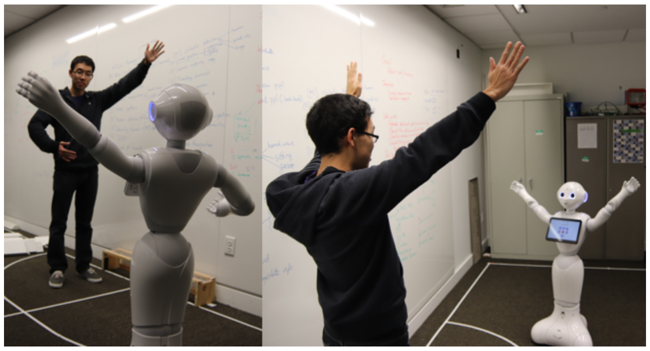


Figure 2: Pepper mimicking a human's arms.

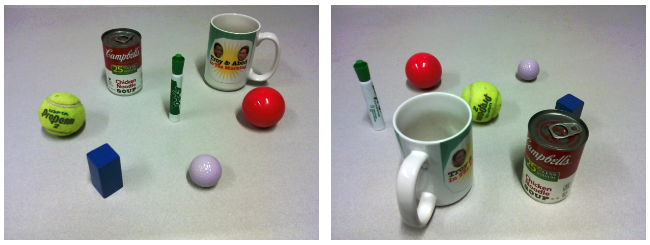


Figure 3: Example images of everyday objects from our object recognition experiments.

Table 1: A summary of class scores for Faster R-CNN and confidence levels for YOLO across 50 images.

	Faster R-CNN				YOLO		
	Q1	Q2	Q3		Q1	Q2	Q3
Mug							
cup	99.86	99.91	99.96	cup	88.53	90.65	91.97
refrigerator	2.45	5.22	8.83	sink	0.09	0.29	1.22
bottle	0.87	2.40	4.79	bed	0.10	0.29	0.54
vase	0.00	0.81	1.58	laptop	0.09	0.21	0.39
Can							
bottle	76.49	85.17	90.01	bottle	24.29	40.71	61.73
cup	13.36	33.31	44.99	cup	1.04	3.69	16.03
refrigerator	1.73	4.42	8.60	bed	0.06	0.20	0.41
vase	0.67	1.78	3.41	sink	0.08	0.17	1.22
Tennis ball							
sports ball	69.71	85.90	94.79	sports ball	3.57	27.84	64.23
apple	7.42	15.05	33.26	apple	3.78	15.17	38.29
refrigerator	1.73	4.42	8.46	orange	3.55	10.51	20.95
orange	0.13	1.11	2.04	bed	0.07	0.20	0.41
Block							
refrigerator	1.93	3.84	8.46	cell phone	0.42	1.35	3.54
cell phone	0.00	2.09	10.10	book	0.23	0.82	2.35
book	0.70	1.40	3.33	remote	0.09	0.26	0.85
tie	0.13	1.02	2.23	knife	0.08	0.23	0.82
Golf ball							
sports ball	36.04	50.65	60.14	sports ball	13.52	27.39	50.61
mouse	6.61	10.89	15.95	mouse	2.98	12.72	25.91
apple	7.26	9.32	13.11	apple	0.30	1.44	5.92
refrigerator	3.04	5.82	8.90	orange	0.11	0.47	1.14
Red ball							
apple	50.99	60.11	72.49	apple	30.29	44.43	57.34
sports ball	6.95	11.44	16.31	orange	14.06	25.13	43.51
mouse	4.17	6.15	12.64	sports ball	0.57	2.10	8.00
refrigerator	1.73	4.42	8.46	mouse	0.24	0.51	1.18
Marker							
bottle	76.74	86.95	91.60	bottle	17.71	29.44	45.89
toothbrush	11.03	20.92	46.98	toothbrush	7.75	20.68	36.86
refrigerator	2.83	5.46	8.83	remote	0.29	1.15	2.77
cup	0.68	1.71	3.24	spoon	0.20	0.55	1.35

and processes the data into useful semantic information for the robot to incorporate in its decision making.

For example, in order to determine whether an individual is attempting to signal or gesture to the robot, we determine the direction that they are facing based on their hip and shoulder coordinates and calculate their arm and shoulder angles using the locations of various parts of the arm such as the shoulder, elbow, and wrist. We use a similar approach to enable Pepper to imitate an individual's arm motions in real time, as shown in Figure 2. Furthermore, we can save the arm angles for future use, which could allow Pepper to learn human gestures in the future.

In addition to processing two-dimensional pose information, we also map body keypoints from the 2D image to 3D coordinates in real space using the point cloud generated by Pepper's depth camera. We then extract 3D semantic features about each human, such as their height and their distance from the robot. We also use a combination of the 2D and 3D information to determine whether humans in each frame are standing or sitting.

Finally, OpenPose also provides face and hand tracking, which allows us to extract even more semantic features about each individual. For example, we can determine whether a person is pointing at something, whether they are smiling, in which direction they are looking, and whether their mouth is open. By integrating OpenPose with Pepper, we generate more in-depth information about humans that enhances Pepper's interactive capabilities.

3.2 Object Recognition

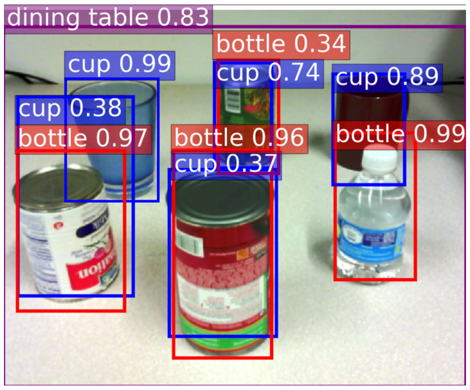
As a social service robot, Pepper should be able to effectively respond to commands that require identifying objects, such as "find me a mug." In order to do so, Pepper needs to be able to recognize generic classes of objects like "mug."

Pepper comes with a keypoint-based object detection algorithm. However, deficiencies of keypoint-based object detection algorithms motivate us to integrate additional algorithms. Most importantly, keypoint algorithms recognize particular instances of objects (e.g., this mug) rather than general classes of objects (e.g., a mug) [23].

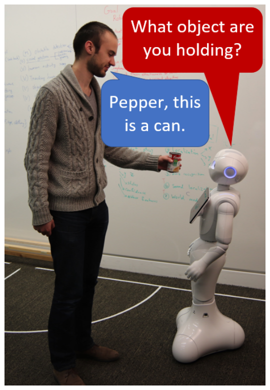
Neural networks, on the other hand, are trained to detect generic classes of objects. We analyze the performance of You Only Look Once (YOLO) and Faster R-CNN on Pepper [1, 19, 21].[1]

In a real-world service environment, Pepper will need to interact with arbitrary classes of objects. We are interested, therefore, in investigating the response of the neural networks to novel objects. We analyze how neural nets that have been trained on the COCO

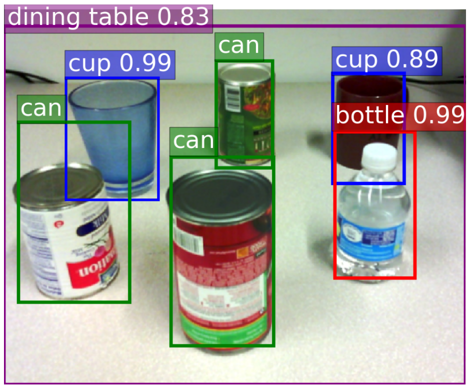
[1]We also ran the pre-trained YOLO9000 [20] model with 9,000 image classes on our images but found that that model did not do as good a job differentiating our particular different everyday objects and returned less confident results.



(a) Initial object detection



(b) Person shows Pepper new class of objects



(c) Object detection after learning new class

Figure 4: Pepper learns to associate can with combination of "cup" and "bottle" classes.

dataset[11] classify objects that do not intuitively correspond to COCO labels.

To empirically measure whether the networks are effective in distinguishing a variety of arbitrary, everyday objects, we have Pepper take images of seven objects in 50 different configurations (examples in Figure 3), and we hand-label the center point of each object in each image. Table 1 presents the first, second, and third quartiles of the confidence scores for the four classes with the highest median confidence for each object. [2]

We first consider the objects which intuitively belong to a COCO class (i.e., mug: "cup," tennis ball: "sports ball," golf ball: "sports ball"). For those objects, we find that the median assigned confidence is highest for the matching class labels, for both YOLO and Faster R-CNN.

We next consider objects like can, red ball, and marker which do not have an intuitive label in the COCO set.[3] We find that Pepper nevertheless consistently assigns certain classes to those objects regardless of orientation. We call those classes "proxy classes." We define a proxy class for an object to be a class label that is consistently triggered more strongly by that object than by other objects in the environment. For example, Table 1 shows that the "toothbrush" class (the second most strongly triggered class by markers) is a proxy class for markers. Pepper is able to identify markers as the only objects in our images that "look like" toothbrushes. Of course, if the environment also contains an actual toothbrush, or some other object that "looks like" a toothbrush, then the "toothbrush" class would not be sufficient to distinguish markers from that other object.

In cases where a single proxy class is not sufficient, combinations of classes can be used to differentiate objects. We propose an algorithm in which the robot searches for objects that "look like" a specific combination of target classes. For example, Figure 4 displays a demonstration in which Pepper is shown a can and learns that the neural networks classify the "can" object as a combination of "bottle" and "cup" classes. It then searches for objects that look like both a "bottle" and a "cup" and is thereby able to identify cans and also to distinguish them from bottles and cups. We also note the robustness of these proxy labels across different object instances in this demonstration.

We propose that by combining information across classes in this manner, any of our everyday objects can be differentiated from the others. We believe that algorithms of this type will be useful in the (what we believe is inevitable) social scenario where a robot is asked to interact with a novel generic object class that it has not previously been trained on.

4 SPEECH

As speech recognition algorithms have improved, speech has become an increasingly important modality for human-robot interaction. We have conducted experiments to identify the strengths and weaknesses of Pepper's inbuilt speech recognition and augmented the existing software with cloud-based speech recognition to improve the accuracy and robustness of Pepper's speech interaction. Furthermore, we have developed the capability for Pepper to learn to better recognize speech as it experiences social interactions.

4.1 Experiments

We conducted a number of experiments to measure speech recognition accuracy. To ensure that our experiments are reproducible and valid comparisons across services, we used a studio microphone to record test sentences and then replayed these sentences through a high fidelity speaker located at the height of a human mouth, facing Pepper.

The sentences are in the domain of plausible robot commands and queries. Examples of sentences in the test corpus are "Bring me a cup" and "What were you doing Tuesday evening at 5 p.m.?" The accuracy metric used throughout these experiments is the proportion of sentences recognized with zero errors.

Recognition accuracy does not give the entire picture. The speech algorithms also return a confidence level for their results. Therefore,

[2]The "dining table" class is removed from Table 1 since both neural nets assign that class a large bounding box around all the objects (correctly identifying that these objects are resting on a table).
[3]The 80 COCO class labels can be found at http://cocodataset.org

Table 2: Accuracy measurements of inbuilt speech recognition. The baseline vocabulary contains 10^5 sentences while the small vocabulary contains only the 11 sentences to be recognized. Displayed results are over 33 recognitions per individual for five individuals.

Vocabulary size	Recognition accuracy	Median (1Q, 3Q) confidence (correct)	Median (1Q, 3Q) confidence (incorrect)
Baseline	0.69	0.66 (0.62, 0.69)	0.57 (0.52, 0.61)
Small	0.93	0.64 (0.58, 0.68)	0.38 (0.37, 0.41)

we also show the median, 1st and 3rd quartiles of the distribution of confidence levels conditional on correct and incorrect recognitions.

4.2 Inbuilt Speech Recognition

Pepper comes built-in with the speech recognition software Nuance Vocon, a compact speech solution for embedded systems, which is accessed through NAOqi.

The user inputs a vocabulary of phrases to be recognized. Whenever the software recognizes audio that matches a phrase in the vocabulary, it returns that phrase along with a confidence level. Only exact matches to phrases are recognized.

4.2.1 Base Accuracy. Table 2 displays the recognition accuracy of the inbuilt speech software for our baseline vocabulary and a tiny vocabulary consisting only of the sample sentences. Recognition accuracy is considerably lower for the larger grammar, as the software faces a less constrained problem and has more opportunities to make errors. A common error is leaving out part of the utterance ("get the cup" rather than "get the cup from the kitchen," or "grab the jar" rather than "grab the yellow jar").

The distributions of confidence levels for correct and incorrect recognitions overlaps considerably, though confidence does have some predictive power: the third quartile for incorrect recognitions is approximately equal to the first quartile for correct recognitions. We face a trade-off between acting on incorrect recognitions and throwing away valid recognitions.

4.2.2 False Positives. When users say something that is not in the vocabulary, the inbuilt speech recognition software sometimes still erroneously matches it to a phrase in the vocabulary. Such false positive recognitions are in principle not a problem, as long as associated confidence levels are sufficiently low to distinguish false positives from accurate recognitions.

Table 3 shows the confidence levels for false positive recognitions by similarity to the vocabulary. For sentences with at least two words different from any sentence in the vocabulary, confidence levels of false positives lie well below those of accurate recognitions and can be easily separated. For sentences only a single word different there is considerable overlap between distributions.

4.2.3 General Speech. The inbuilt software's inability to recognize phrases that are out of vocabulary and its sensitivity to false positives present a serious challenge. As a social robot, Pepper will likely be exposed to a variety of social interactions, and commands or queries can be expressed in a large number of ways.

Table 3: False positive rates of inbuilt recognition by similarity of test sentence to closest sentence in vocabulary. Displayed results are over 15 recognition attempts per category per individual for five individuals

Difference from grammar	Median (1Q, 3Q) confidence
1 word difference	0.58 (0.53, 0.63)
2 word difference	0.43 (0.35, 0.49)
No similarity	0.33 (0.30, 0.36)

One possible method to reduce false positives and improve the capacity of the robot to understand a variety of speech is to create a very large vocabulary. Other than increasing errors between known sentences, this also creates problems with loading the vocabulary. Every time a user launches a recognition script, the vocabulary has to be reloaded, and this process is quite time consuming: our vocabulary for the "retrieve object" class of robot commands contains approximately 10^6 sentences and takes slightly under two hours to load. General purpose grammars could be much larger than that, and loading time is approximately linear in vocabulary size, so simply extending the vocabulary is not a viable solution.

Furthermore, we would like Pepper to learn about its environment and how humans prefer to interact with it, and for that we need Pepper to be able to recognize previously unanticipated speech.

4.3 Cloud-based Speech Recognition

Google Cloud Speech is a cloud-based streaming speech recognition software platform [7]. Users connect to the service and send streaming audio, and the service returns transcription results in real-time, along with a confidence level. Unlike the inbuilt service, users do not specify the speech they expect to receive. Google's speech algorithm consists of a deep neural network that has been trained on a large amount and variety of speech from Google users, and is able to recognize general speech [18].

To improve accuracy and help recognize speech from any angle, we separately stream audio from each of Pepper's four differently positioned microphones to Google, and use the recognition result with the highest confidence.

4.3.1 Comparison. Table 4 displays a comparison of accuracy between the cloud-based speech and inbuilt recognition. The cloud-based speech recognition achieves slightly higher accuracy with less overlap between confidence levels of correct and incorrect recognitions.

The cloud-based speech recognition service allows for more general speech and appears to perform somewhat better, but it also requires an active internet connection with sufficient speed for streaming audio, which may not always be available. Moreover, cloud-based speech recognition incurs additional processing time, though with a fast internet connection we have found this latency to be <1s.

Table 4: Accuracy comparison of speech recognition systems. Displayed results are over 33 recognition attempts per category per individual for five individuals.

Recognition software	Recognition Accuracy	Median (1Q, 3Q) confidence (correct)	Median (1Q, 3Q) confidence (incorrect)
Inbuilt	0.69	0.65 (0.61, 0.68)	0.57 (0.51, 0.61)
Cloud-based	0.73	0.97 (0.94, 0.98)	0.89 (0.81, 0.94)
Combination	0.85	0.90 (0.75, 1.0)	0.75 (0.60, 0.87)

4.4 Dual Approach

The different voice recognition systems use very different approaches. As a result, their recognition errors are not strongly correlated, which implies that we might gain additional accuracy by ensembling them.

We run the inbuilt and cloud speech speech software simultaneously and use a separate processing module that combines the recognitions. Using experimental data from the previous sections, we estimate accuracy functions $p_{\text{inbuilt}}(\text{confidence})$ and $p_{\text{cloud}}(\text{confidence})$. For each utterance, we return the recognition result from the service with the highest implied accuracy.

Table 4 shows that the combined approach yields a significant gain in accuracy relative to both the cloud-based and inbuilt speech recognition software individually.

4.5 Parsing

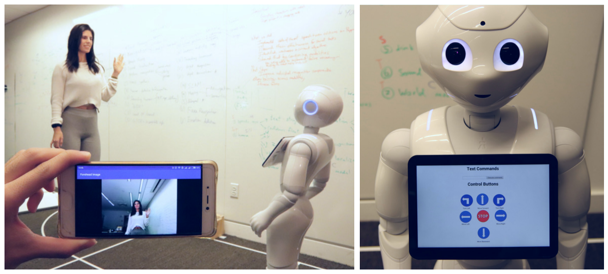
After an utterance has been transcribed to text, the next step is to match that text to a robot command. Our general approach is to use a frame semantic parser. Each recognized sentence is broken down into constituent parts, which are labeled. Then, we attempt to match the labeled sentence to one of a number of semantic frames that correspond to different robot commands.

We train a conditional random field to label sentences [10, 14]. Initially, we create a training vocabulary of the type of sentences we expect. Rather than enter sentences individually, we generate sentences from a context-free grammar (CFG) [26]. A CFG consists of a set of terminals and a set of production rules to turn those terminals into sentences. We train the conditional random field on a large number of generated sentences, after which the conditional random field is able to generalize to parsing related sentences. For example, the parser is able to learn that the sentence “get me the banana from the fridge” matches the frame “retrieve object” and that “banana” and “fridge” are the object and location of the object, even if it has never seen the object or the location before.

4.6 Learning Vocabulary

We use the information from cloud-based recognition to improve the inbuilt speech recognition over time by learning new phrases that Pepper encounters. Whenever Pepper hears a sentence containing terminals with which it is not familiar, we add those terminals to the database to be added to the grammar for the next session, in addition to the raw sentence.

For example, if Pepper hears “get me the banana from the fridge”, “banana” would be added to the list of objects and “fridge” to the list of locations for the “retrieve object” frame. In this way, Pepper can learn to understand many new sentences from few interactions.



(a) Pepper’s mobile application **(b) Pepper’s tablet**

Figure 5: Examples of alternate inputs to Pepper.

Because vocabulary size is constrained by loading time and large-grammar recognition performance, we keep track of the frequency with which Pepper encounters sentences and terminals. Then, before each session, the most probable sentences corresponding for the semantic frames for interactions we expect to encounter during the session are loaded into the vocabulary.

Storing all the recognized sentences also allows us to improve the parser over time, as we manually annotate sentences that the parser was not able to successfully match to a robot command.

5 MULTI-MODALITY

In previous sections we described Pepper’s existing vision and speech functionality, and outlined a number of improvements to the capability and robustness of those systems. Even with the improvements, there will always be occasions where those input methods fail, or are inappropriate. It is important for Pepper to be robust to those situations and still be able to interact with users, if in a more limited way. For that reason, we have developed a number of alternate input methods. In addition, we also continually monitor the quality of the various input methods. When one input method is unreliable, Pepper actively requests input through a different modality from the user.

5.1 Alternate inputs

5.1.1 Tablet. Pepper has a built-in android tablet on its chest that can accept touch inputs from users (Figure 5b). When speech or vision inputs are unavailable or unreliable, Pepper actively instructs users to input their commands through the touch screen, either by typing instructions or pressing a button. While somewhat more inconvenient than speech, the availability of this input method improves the robustness of overall communication.

5.1.2 Mobile Application. Many individuals now carry smartphones with internet capabilities. To make use of this, we developed an android mobile application that allows a user to remotely control Pepper via an interactive and intuitive interface.

Users can give Pepper commands by selecting from a menu, typing the command, or through speech, connected to the Google Android cloud-based recognition platform. The user can also access what Pepper can see (Figure 5a). We communicate between modules that carry out tasks through ROS, so users can request any task the robot is capable of executing.

5.1.3 External Microphones. When operating in a controlled environment, we do not have to rely solely on the robot hardware. For example, a popular use for Pepper is as a greeter in stores. In such an environment, it is possible to deploy stationary microphones. We use conference room microphones to augment Pepper's onboard microphones in a similar fashion to Amazon Alexa, actively listening for sentences that start with "Pepper" or another command word. We find that combining the inbuilt speech, cloud-based speech, and external microphones improves recognition accuracy. In addition, this method allows users to control the robot when they are far away from the robot but close to one of the external microphones.

5.2 Failure-Tolerant Modality Selection

We have introduced a number of alternate modalities. If one modality fails, Pepper can still successfully interact through the remaining modalities. We continually keep track of the current reliability of input methods and request alternate input from the user as appropriate.

5.2.1 Speech. For speech, we monitor the audio stream quality to determine whether cloud-based speech recognition is likely to be reliable. If it is not reliable, speech recognition will be limited to phrases in the vocabulary. When a user attempts an interaction that is unlikely to succeed given the vocabulary, Pepper directs that user to use the tablet instead.

Another sign of poor quality speech recognition is when consecutive speech recognition results either exhibit low confidence or result in a parse that cannot be grounded. In this case Pepper can either direct the user to use the tablet, or to use vision to ground a command instead. Figure 6 demonstrates Pepper's response in a scenario where it is asked to count a particular kind of object, but Pepper has either not heard the object correctly or does not yet know its name. Pepper instead asks the user to show it an example of the object visually, after which it successfully completes the task.

5.2.2 Vision. For vision, we monitor the confidence levels of detected objects and human pose keypoints. If the robot is currently executing a task that requires knowledge of a person's arms, but the arms are being blocked by an obstacle, outside the field of view or given low confidence levels, the robot will actively prompt the human to step back within view, or request instructions through speech instead.

Another way that speech can help recover vision from failure is when the user asks Pepper to find or retrieve an object, but it does not detect the object with sufficient confidence. In that case, Pepper requests additional information from the user to aid in locating the object, such as the size or color ("the cup is small and red"), or its location relative to an object that is easier to identify ("it's on the table next to the bottle of water").

6 RECOGNIZING ACTIVITY PATTERNS

For effective human-robot interaction it is crucial that the robot is capable of communicating its current state. In addition, a user may want to learn what activities an autonomous robot previously performed while it was acting without supervision. To enable this, it is necessary for the robot to have a memory and possess a rudimentary awareness of its actions. We present a general, online approach for recognizing and categorizing activity solely based on joint angles. For classification, we applied and evaluated Long Short Term Memory (LSTM) [9] deep neural network and nearest neighbor based approaches. We trained these algorithms to recognize various motions that were executed by Pepper in the robotics simulator Gazebo.

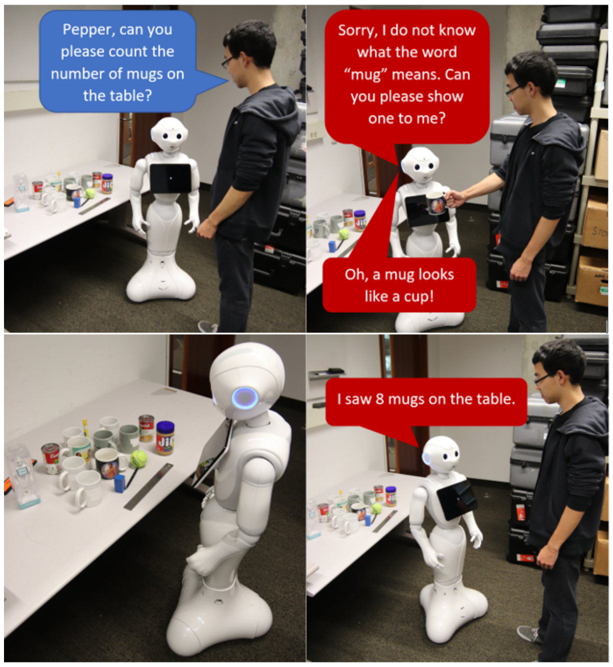

Pepper, can you please count the number of mugs on the table?
Sorry, I do not know what the word "mug" means. Can you please show one to me?
Oh, a mug looks like a cup!
I saw 8 mugs on the table.

Figure 6: Pepper multi-modality demo.

6.1 Learning Motion Patterns

First we define a motion sequence for a given robot with n measurable joint angles during a time frame $[t_i, t_{i+k}]$ for $i, k \in \mathbb{N}$ as $T_A(t_i, t_{i+k}) = \{M_i, M_{i+1} \ldots, M_{i+k}\}$ where $M_j = (x_1, \ldots, x_n)$ and x_h is the value of a specific joint for $h \in 1, \ldots, n$ at time $j \in [t_i, t_k]$. Therefore, we deal with classifying a time series with values in $\mathbb{R}^n$.

6.1.1 Similarity Between Motions. Our nearest neighbor based approach builds on previous work on behavior recognition in robot soccer [5]. As a similarity metric between motion sequences we use the Hausdorff metric in combination with the Euclidean distance d. Given two motion sequences T_1 and T_2 the Hausdorff distance is defined as:

$$H(T_1, T_2) = \max\{\max_{p \in T_1} \min_{q \in T_2} d(p, q), \max_{p \in T_2} \min_{q \in T_1} d(p, q)\} \tag{1}$$

We reduce the computational cost for classifying a motion sequence T with m measurements to $O(|C| * \max_{c \in C} |c| * m)$ by determining the centroid $c \in C$ for each class, averaging over a set of labeled training sequences. Using the Hausdorff distance allows us to measure the similarity between motion sequences even if they have different durations.

6.1.2 Motion Sequence Classification with LSTM. The correct classification for a sequence of motions can depend on the entire sequence (for example, moving from one location to another might be classified differently depending on whether or not the robot picked up an object beforehand). LSTM networks allow for information to persist through an entire observation sequence and can model this long-range dependency. The hidden state output of the LSTM after each observation input from a training sequence captures the underlying property that induced the sequence of actions. A relu layer and a softmax layer transform the hidden state into a probability distribution over all possible classes of actions.

6.1.3 Online Classification. We apply our algorithms iteratively on the observation in order to recognize partial motion sequences. Let t be the current time, $T_p = \{M_1, ..., M_j\}$ be the partial motion sequence that has been recorded since time t_1 and let t_k be the end of the motion so that $t_1 \leq t \leq t_k$.

In our nearest-neighbor-based approach we want to find the centroid $c \in C$ that is most similar to T_p. To compare with the partial motion sequence we limit the duration of c to that of T_p (in the case $|T_p| >= |c|$) and calculate $H(T_p, c_p)$ where $c_p = \{M'_1, ..., M'_k\}$.

In the LSTM approach, the LSTM can take in partial sequences with first n measurements and the missing $70 - n$ measurements filled in as zero vectors. By doing so, each training sequences of length 70 was transformed into 70 samples for the training process.

6.2 Collecting data

The robot has 20 degrees of freedom (DOF) (shown in Figure 1b). It has an omnidirectional base with 3 DOF, a body with 3 DOF, two humanoid arms with 5 DOF, two hands with 1 DOF, and a head with 2 DOF. We receive ROS messages with measurements of the 20 joint angles of Pepper at up to 50 hz and store this data in a lightweight database using SQLite.

6.3 Experimental Results

We first evaluated our models on 5 arm-based motions (shown in Figure 7, along with the initial starting position) and later expanded to a set of 20 arm-based motions. By using the robotics simulator Gazebo in combination with the control software MoveIt! we collected 50 samples for each motion. We used 700 samples for training and 300 for the test set. Each sample describes a 7-second-long motion sequence of Pepper with 10 measurements per second.

The data is arranged into a 3-dimensional tensor as an input to the LSTM. The first dimension is the number of sequences, the second dimension is the length of each sequence, and the third dimension is the number of features recorded at each time step. We used a batch size of 10.

We want to study the accuracy of our classification algorithms using the first l measurements from the beginning of the motion, for $0 < l \leq 70$, to see how much data is necessary to determine the class of actions (Figure 8). The nearest neighbor based approach is capable of recognizing most activities after a few measurements. The mixed length LSTM achieves similar prediction accuracy after around 10 readings. A few of our motions are identical until around the

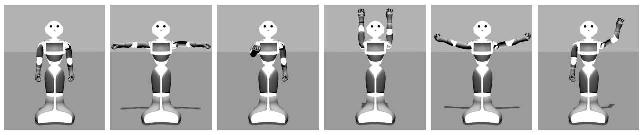


Figure 7: The leftmost image shows the initial starting position (standing). The next five images show end positions of the five movements in our first test.

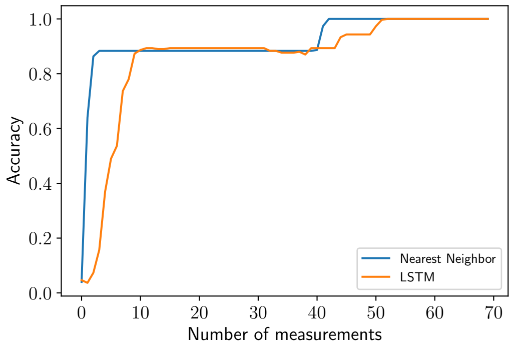


Figure 8: Accuracy over variable length of observation sequence.

40th measurement are classified correctly by the nearest neighbour algorithm soon after. The LSTM needs a few measurements more until to achieve the same result.

7 CONCLUSIONS AND FUTURE WORK

In this work we extend and evaluate Pepper's capability for human-robot social interaction. We integrate state-of-the-art vision and speech recognition systems on Pepper and we contribute an empirical analysis of their effectiveness, outlining areas of weakness and proposing improvements. We also contribute a learning algorithm to improve communication capabilities over time, updating speech recognition through social interaction.

As we recognize limitations of the individual perceptual modalities, we contribute a multi-modality approach to increase the robustness of human-robot social interaction. We add alternative input methods in the form of a mobile phone application, external microphones, and a custom interface on the onboard tablet, and we actively seek input from different modalities when one modality proves unreliable, providing robustness to the failure of individual components. To improve the transparency of the robot's behavior, we exploit the availability of rich robot body-sensory data and apply nearest-neighbor and deep learning algorithms to enable Pepper to classify and verbalize a variety of its own body motions.

We view the techniques presented in this work to be relevant both specifically to Pepper and to other social robots.

REFERENCES

[1] Marko Bjelonic. 2017. YOLO ROS: Real-Time Object Detection for ROS. (2017). Retrieved October 30, 2017 from https://github.com/leggedrobotics/darknet_ros

[2] J. Bohren, R. B. Rusu, E. Gil Jones, E. Marder-Eppstein, C. Pantofaru, M. Wise, L. MÃűsenlechner, W. Meeussen, and S. Holzer. 2011. Towards autonomous robotic butlers: Lessons learned with the PR2. In *2011 IEEE International Conference on Robotics and Automation*. 5568–5575. https://doi.org/10.1109/ICRA.2011.5980058

[3] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. 2017. Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. In *CVPR*.

[4] Dragos Datcu, M Richert, T Roberti, W De Vries, and LJM Rothkrantz. 2004. AIBO Robot as a soccer and rescue game player. *Proceedings of GAME-ON 2004* (2004), 45–49.

[5] Can Erdogan and Manuela Veloso. 2011. Action Selection via Learning Behavior Patterns in Multi-Robot Systems. In *Proceedings of the International Joint Conference on Artificial Intelligence*. Barcelona, Spain.

[6] Jonathan Gerbscheid, Thomas Groot, and Arnoud Visser. 2016. UvA@ Home 2017 standard platform proposal. *simulation* 8 (2016), 9.

[7] Google. 2017. Google Cloud Speech. (2017). Retrieved November 9, 2017 from https://cloud.google.com/speech/

[8] Nick Hawes, Chris Burbridge, Ferdian Jovan, Lars Kunze, Bruno Lacerda, Lenka Mudrová, Jay Young, Jeremy L. Wyatt, Denise Hebesberger, Tobias Körtner, Rares Ambrus, Nils Bore, John Folkesson, Patric Jensfelt, Lucas Beyer, Alexander Hermans, Bastian Leibe, Aitor Aldoma, Thomas Faulhammer, Michael Zillich, Markus Vincze, Muhannad Al-Omari, Eris Chinellato, Paul Duckworth, Yiannis Gatsoulis, David C. Hogg, Anthony G. Cohn, Christian Dondrup, Jaime Pulido Fentanes, Tomás Krajník, João M. Santos, Tom Duckett, and Marc Hanheide. To Appear. The STRANDS Project: Long-Term Autonomy in Everyday Environments. *IEEE Robotics and Automation Magazine* (To Appear).

[9] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[10] Thomas Kollar, Vittorio Perera, Daniele Nardi, and Manuela Veloso. 2013. Learning environmental knowledge from task-based human-robot dialog. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 4304–4309.

[11] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*. Springer, 740–755.

[12] Sharon Oviatt. 1999. Mutual Disambiguation of Recognition Errors in a Multimodel Architecture. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '99)*. ACM, New York, NY, USA, 576–583. https://doi.org/10.1145/302979.303163

[13] Vittorio Perera, Sai P Selveraj, Stephanie Rosenthal, and Manuela Veloso. 2016. Dynamic generation and refinement of robot verbalization. In *Robot and Human Interactive Communication (RO-MAN), 2016 25th IEEE International Symposium on*. IEEE, 212–218.

[14] Vittorio Perera and Manuela Veloso. 2017. Learning to Understand Questions on the Task History of a Service Robot. (2017).

[15] Luis A Pineda, Caleb Rascon, Gibran Fuentes, Arturo Rodrıguez, Hernando Ortega, Mauricio Reyes, Noé Hernández, Ricardo Cruz, Ivette Vélez, and Marco Ramırez. 2017. The Golem Team, RoboCup@ Home 2017. (2017).

[16] Lerrel Pinto, Dhiraj Gandhi, Yuanfeng Han, Yong-Lae Park, and Abhinav Gupta. 2016. The Curious Robot: Learning Visual Representations via Physical Interactions. *CoRR* abs/1604.01360 (2016). arXiv:1604.01360 http://arxiv.org/abs/1604.01360

[17] Agence France Presse. 2016. Robot receptionists introduced at hospitals in Belgium. (2016). Retrieved November 15, 2017 from https://www.theguardian.com/technology/2016/jun/14/

[18] Emil Protalinski. 2017. Google's speech recognition technology now has a 4.9% word-error-rate. (2017). Retrieved November 9, 2017 from https://venturebeat.com/2017/05/17/googles-speech-recognition-technology-now-has-a-4-9

[19] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 779–788.

[20] Joseph Redmon and Ali Farhadi. 2016. YOLO9000: Better, Faster, Stronger. *arXiv preprint arXiv:1612.08242* (2016).

[21] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *arXiv preprint arXiv:1506.01497* (2015).

[22] RoboCup. 2017. RoboCup@Home Leagues. (2017). Retrieved November 13, 2017 from https://www.robocup2017.org/eng/leagues_home.html

[23] SoftBank Robotics. 2017. NAOqi API and Pepper documentation. (2017). Retrieved October 30, 2017 from http://doc.aldebaran.com/2-5

[24] SoftBank Robotics. 2017. Who is Pepper? (2017). Retrieved November 1, 2017 from https://www.ald.softbankrobotics.com/en/robots/pepper

[25] Stephanie Rosenthal, Sai P Selvaraj, and Manuela M Veloso. 2016. Verbalization: Narration of Autonomous Robot Experience.. In *IJCAI*. 862–868.

[26] Michael Sipser. 2006. *Introduction to the Theory of Computation*. Vol. 2. Thomson Course Technology Boston.

[27] Jörg Stückler, Ishrat Badami, David Droeschel, Kathrin Gräve, Dirk Holz, Manus McElhone, Matthias Nieuwenhuisen, Michael Schreiber, Max Schwarz, and Sven Behnke. 2013. Nimbro@ home: Winning team of the robocup@ home competition 2012. In *RoboCup 2012: Robot Soccer World Cup XVI*. Springer, 94–105.

[28] Jörg Stückler, Ishrat Badami, David Droeschel, Kathrin Gräve, Dirk Holz, Manus McElhone, Matthias Nieuwenhuisen, Michael Schreiber, Max Schwarz, and Sven Behnke. 2013. *NimbRo@Home: Winning Team of the RoboCup@Home Competition 2012*. Springer Berlin Heidelberg, Berlin, Heidelberg, 94–105. https://doi.org/10.1007/978-3-642-39250-4_10

[29] Manuela Veloso, Nicholas Armstrong-Crews, Sonia Chernova, Elisabeth Crawford, Colin McMillen, Maayan Roth, Douglas Vail, and Stefan Zickler. 2008. A team of humanoid game commentators. *International Journal of Humanoid Robotics* 5, 03 (2008), 457–480.

[30] Manuela M Veloso, Joydeep Biswas, Brian Coltin, and Stephanie Rosenthal. 2015. CoBots: Robust Symbiotic Autonomous Mobile Service Robots.. In *IJCAI*. 4423.

[31] Sam Byford (The Verge). 2014. SoftBank announces emotional robots to staff its stores and watch your baby. (2014). Retrieved November 1, 2017 from https://www.theverge.com/2014/6/5/5781628/softbank-announces-pepper-robot