A Mobile Agents based Distributed Speech Recognition Engine for Controlling Multiple Robots

Mayank Gupta, Pulkit Verma, Tuhin Bhattacharya, Pradip K. Das
Department of Computer Science & Engineering
Indian Institute of Technology Guwahati
Guwahati, India
{mayank.2013, v.pulkit, b.tuhin, pkdas}@iitg.ernet.in

ABSTRACT

Interaction with a robot has been an active area of research since the inception of robotics. Talking to a robot has always been considered the most natural way to communicate with it. But it is not always possible to have a full-fledged, standalone speech processing engine to be present on a robot or on a single machine. A dedicated system to convert the commands from audio to text is needed. However, as the number of commands and robots increases, it becomes necessary to eliminate all the single-point failure points in the system. Thus, distributed speech engine comes into picture. Also users may want to talk to the robot in different languages.

The approach proposed in this paper is distributed, fault tolerant and scalable, such that any new recognition algorithm or language support can be added and used without any changes to the existing system.

The work has been demonstrated on a freely available mobile agents based Internet of Things platform. However, any platform can be used.

Categories and Subject Descriptors

1.2.7 [**Artificial Intelligence**]: Natural Language Processing - speech recognition and synthesis; 1.2.9 [**Robotics**]: autonomous vehicles, operator interfaces.

General Terms

Algorithms, Experimentation

Keywords

Robot, Human-machine interaction, Speech recognition, Distributed Services, Mobile Agents

1. INTRODUCTION

Man-machine interaction has been an important area of work for many decades now. Speech being a natural way of communication among humans is far more intuitive and easier to use for people in applications like controlling robots, hands free operation of devices and systems, etc.

In hazardous environments the robots may need to be controlled remotely. It can also be beneficial when the computer or robot specific knowledge of robot operators is limited but they are experts in the area where robots are being used. It can also be useful for differently abled people who are required to control these robots. A very obvious way is to 'call' the robots using a telephone and 'tell' them the task to be performed. s

This paper proposes a distributed and scalable speech recognition engine. It handles requests from multiple clients to control a set of robots. This removes the chance of any single point failure. Mobile Agents are used to autonomously distribute the recognition task among multiple machines which hosts the the speech recognition engine. Since the system is distributed, the load capacity can be increased by increasing the number of recognition engines therein making the system scalable. Thus, virtually any number of robots and users can be supported. Also the system supports commands in multiple languages. The support for new languages can be added dynamically without changing the existing system.

AIR '15, July 02 - 04, 2015, Goa, India

DOI: http://dx.doi.org/10.1145/2783449.2783477.

2. RELATED WORK

The initial approaches to achieve this task were aimed at using voice driven control signals. The VoiceBot [5] was developed using this approach where voice was used to specify the control signals which in turn manipulated a real 3 dimensional robotic arm. But this system did not take proper speech commands as input to the system.

Some of the works were aimed at enabling robots to have their own voice recognition systems [11]. But in these systems the recognition module had to be replicated on each robot thereby increasing the overall resource usage and draining the robot batteries. Also each robot had to be equipped with sufficient computation power to carry out recognition tasks hence increasing the cost and complexity of each robot.

A possible solution to this problem is to have a dedicated speech recognition engine which converts the voice commands into text format understood by the robots. This concept was used in ActivMedia PeopleBot [8] where simple oral commands to control a mobile robot were used. It used a separate speech recognition engine and robotic control unit. This system was restricted to simulator hence its workability in real world scenarios was not known.

Interfaces to control LEGO® NXT Mindstorm® robots have also been proposed. Jha and Nair [6] developed "Logic Programming Interface" for controlling and programming multiple LEGO® NXT robots. The major problem with this system was that it required some basic familiarity with Prolog. This problem was solved in [1] where voice based input was used to control the LEGO robots. But it worked for Mexican Spanish voice and was developed to control a single robot.

A better approach was proposed in [12] where a client-server architecture for controlling multiple robots using voice commands was used. But the problem with these kind of systems is that they are not scalable and cannot handle high loads. As the number of robots increases, these systems tend to slow down, resulting in low performance of the system. And since these systems are dependent on one server, they are susceptible to single point node failures, which may render the whole system useless.

Speech remote control system [13] has also been proposed which is distributed. It is based on web service and automatic speech recognition. But in this system also the core server recognizing the utterances is single and thus is prone to single point node failure.

Mobile Agents can provide all the major functionalities that are found in distributed architectures. They can move from one node to another and also make the system dynamic, adaptable, robust and scalable. Also using mobile agents relieves the task of considering the functional equality of all distributed computing nodes. This makes the development of the system more logical. Due to these advantages mobile agents have been implemented in a large number of areas [10].

Application of mobile agents in the field of robust teleoperation [3] has already been explored in the field of nuclear decommissioning. Mobile Agents have also been used in ALLIANCE architecture [4] which extended the adaptability and fault tolerance in simulated and real networked robots. Framework for cooperative control of multi-robots using mobile agents [17] has also been proposed recently. All these approaches using mobile agents work with usual text-based command interfaces.

The goal of this paper is to present a simple voice based control system for multiple robots which is distributed, scalable and robust. The voice commands used are highly intuitive like ACQUIRE, RELEASE, FORWARD, BACKWARD, etc. When used in conjunction with corresponding robot ID and client ID, they perform the specific tasks corresponding to them.

3. PROPOSED APPROACH

3.1 Components

The proposed system has five components as shown in Figure 1.

1. **Robot**: It is the robotic device present in an arena and to be controlled by the robot controller. We have used LEGO® NXT Mindstorms® robot for the purpose of the experiments. Robots are identified by their unique robot IDs.

2. **Robot Controller**: It is directly connected to the robots and acts as an interface for them. It gets the commands from the control server and performs corresponding operations on the robots. The authentication of command source is also done here.

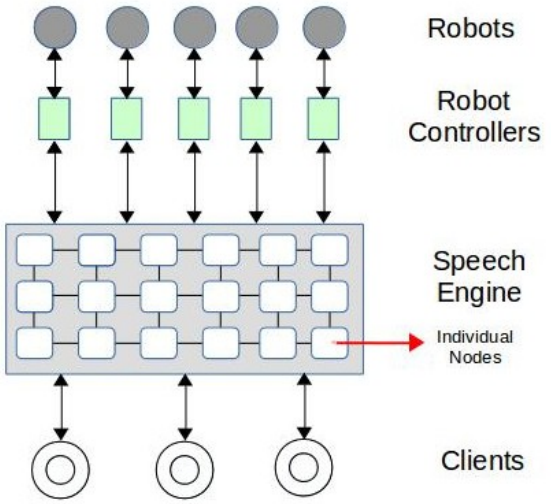

Figure 1: Layered Architecture showing relationships between different components

3. **Control Server**: It is selected randomly from the nodes that are part of the speech engine. It distributes the recognition task. It sends the command to the robot controller and simultaneously gives a positive/ negative score to the model of the speech engine.

4. **Speech Engine**: Here different speech recognition algorithms work in parallel for a particular input signal and sends the output probabilities to the control server. It can cater to the requests from a large number of clients simultaneously. Also, different recognition methods can be run in parallel to improve the accuracy of the system over time.

   Support for multiple languages is also provided here. All languages will have separate models hosted on different nodes and mobile agents are sent to each one of them. The model which gives the maximum similarity score is chosen.

5. **Client**: It receives the speech signal from the user and sends it to the speech engine. It also confirms the command just before performing it upon the robots. Each client has a unique user ID for authentication purposes.

3.2 Steps

The user speaks a command like "Acquire One" to his phone or any other speech interface. This command is sent via the network to the speech engine.

Each new command is sent to a random node within this engine. This node acts as the control server for that particular request. Bramson et al. [2] proves this kind of random distribution of load to be optimal. The control server performs the task of parallelizing the recognition based on word models. Since speech recognition algorithms are model-based wherein we have a model for each word in the vo-

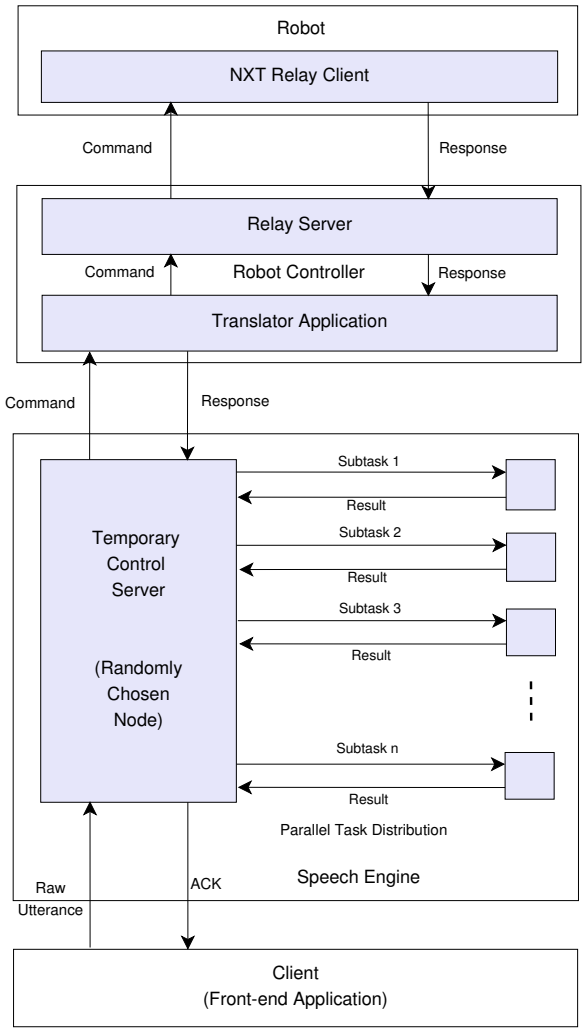

Figure 2: Flowchart of a typical request

cabulary, the task is well suited for distribution across a network.

Mobile agents carry the spoken phrase/command as a payload to different nodes of the recognition engine, each of which finds the similarity between the spoken word and the models that are hosted on that node. This similarity is quantified as the probability that the spoken word was generated using that model.

The maximum probabilities at each node are then returned to the control server, which compares them and decides the most probable command. This is sent back to the user for verification purposes. If the user replies positively, a positive score/feedback is sent to the model and the command is interpreted and the corresponding opcodes are forwarded to the robot controller. If the user feedback is negative, the model is given a negative score and the next most probable model is chosen as the candidate and the user is asked for confirmation again and so on.

For a request with positive feedback, the recognized command is forwarded to the corresponding robot controller.

Figure 2 shows the control flow for a typical request from a client.

3.3 Degree of Parallelization

Ideally the time taken to recognize a command can be decreased by decreasing the number of models hosted at each node. This will bring down the processing time at each node. But since the agents take some finite time to travel across the nodes, the performance benefit achieved at each node is nullified by this agent's travel time. Hence to achieve the maximum performance the processing time at each node should be sufficiently greater than the agent's travel time.

In a serial system where all models are present on a single machine, the processing time t_s is calculated as:

$$t_s = m \times t_1 \tag{1}$$

where, t_1 is the processing time to compare a single model with spoken word and m is the number of models (size of the vocabulary).

In a parallel system with k models per node, the processing time to compare all the models with the spoken word t_p is given as:

$$t_p = (\frac{m}{k} \times t_2) + (k \times t_1) \tag{2}$$

where, t_2 is the average time taken by an agent to travel to its destination node and return. It depends on the network bandwidth and machine architecture.

To achieve $t_p < t_s$, we need to ensure that:

$$(\frac{m}{k} \times t_2) + (k \times t_1) < m \times t_1 \tag{3}$$

Since in a parallel environment $k << m$,

$$\frac{m}{k} \times t_2 < m \times t_1 \tag{4}$$

Hence

$$t_2 < k \times t_1 \tag{5}$$

This implies that the processing time at each node must be sufficiently greater than the time taken by a mobile agent to travel across the network.

3.4 Authentication

Any user must use the ACQUIRE command to establish a connection with a robot. This connection will be alive until the user issues the RELEASE request. The robot is acquired by the client whose request reaches the robot controller first. Only this client is now eligible to send commands to the corresponding robot. The robot controller uses this user ID to authenticate the future commands.

If for any command the authentication fails, an error message is sent back to the user. If it passes, the task based on the command is performed on the robot and a feedback is sent to the corresponding client.

3.5 Robot Controller

The robot controller is a combination of three components working together to connect the robots with the speech engine.

1. **Translator Application**: It is a Java-based application that acts as a communication link between the relay server and the speech recognition engine. The translator continuously listens to the speech engine and gets

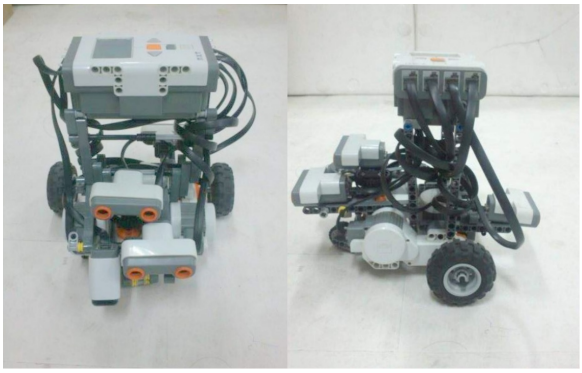

Figure 3: The LEGO® NXT Mindstorm® Robot

the text equivalent of the spoken command. It then parses the command, generates the equivalent opcodes and sends to the relay server. It also sends back the response it collects from the relay server back to the control server.

2. **Relay Server**: It is a Java-based component that connects the robot to the robot controlling host. It communicates with the robot using socket programming and takes the command from the translator application and performs the associated operations on the robot.

3. **NXT Relay Client**: It is the relay server counterpart that runs on the robot itself. It continuously communicates with the relay server, takes directions from it and acts accordingly. It also provides feedback to the relay server about the success or failure of the task.

4. EXPERIMENTS

4.1 LEGO® NXT Mindstorm® Kit

The LEGO® NXT Mindstorm® shown in Figure 3 is used as the robot for the pupose of experiments. Various motors and sensors are connected to a master control unit termed as brick. The brick is connected via Bluetooth® to the robot controller which can be any computing device. This connection is facilitated by the Bluetooth® port on the brick which connects directly to the Relay Server. Lejos NXJ firmware is installed on the brick which enables the usage of Java programming environment. For facilitating wireless communication between the brick and the robot controller, Lejos NXJ has Bluetooth® libraries.

4.2 Commands

For experimental purposes, the following set of commands were used:

1. Acquire ⟨Robot ID(s)⟩: This enables client to acquire the robots with robot IDs mentioned in the command.

2. Forward ⟨Robot ID(s)⟩: The robot(s) will move forward from their current position(s).

3. Backward ⟨Robot ID(s)⟩: The robot(s) will move backward from their current position(s).

4. Left ⟨Robot ID(s)⟩ ⟨degrees⟩: The robot(s) will rotate to their left (clockwise direction) by mentioned degrees.

5. Right ⟨Robot ID(s)⟩ ⟨degrees⟩: The robot(s) will rotate to their right (anti-clockwise direction) by mentioned degrees.

6. Stop ⟨Robot ID(s)⟩: The robot(s) will stop moving.

7. Release ⟨Robot ID(s)⟩: The connection between client and robot(s) is terminated.

8. Grab ⟨Robot ID(s)⟩: The robot(s) uses claws to grab the object right in front of it. It must be in the correct position to be able to grab the object.

9. Move ⟨Direction⟩ ⟨Robot ID(s)⟩: The robot(s) will turn in the direction specified and move forward from their current position(s).

4.3 Experimental Setup

We used an HMM toolkit based on [7] that has a predefined set of models. The toolkit calculates the percentage match between the spoken word with each of the models. The model with the highest match is chosen as the spoken word.

We took two instances of the toolkit. One of these compares the models with the words sequentially. Let us call this instance "Serial". As the number of possible words (models in the vocabulary) increases, the time taken by the speech engine should increase linearly, (the time complexity being $\mathcal{O}(n)$).

The other one called "Parallel" was tweaked to compare the word with just one model. So, we could run multiple instances of the engine in parallel - one for each model and return all the results and compare them. This should perform better than $\mathcal{O}(n)$.

The command to control a robot is of the following format: ⟨Operation⟩ ⟨Robot ID(s)⟩.

For example, consider the command "Forward One". If there are 'x' number of possible operations and 'y' number of robots, we need $\mathcal{O}(x + y)$ nodes to compute the result in constant time.

4.4 Mobile Agents

We have used Typhon - A Mobile Agents Framework for Real World Emulation in Prolog [9] to achieve totally distributed and decentralized parallelism. The mobile agent takes the observation sequence (the recorded audio in vector quantized form) as payload to different nodes. Each node calculates the received observation sequence's similarity with the model assigned to that node and returns the probability measure. The speech engine is an executable available at each node. Typhon calls this executable along with the payload as an argument and gets the result back and returns it to the requesting entity.

5. RESULTS

The performance was measured as the time taken to return the result for a given number of models.

Table 1 shows the time taken vs the number of models for the "Serial" approach, i.e. when the recognition is done without any parallelization. These values have been used as

Table 1: Vocabulary size vs average recognition time

No of Words	Average Time (msec)
1	80
20	112
50	138
100	180
500	578
1000	1082
2500	2566
5000	5020
10000	10 168

Table 2: Number of mobile agents vs Average time taken

No of Agents (m/k)	Average Time (msec)
5	75.8
10	145.2
20	300.6
30	439.8
40	585.2
50	751.2
60	907.6
70	1077.8
80	1268.2
90	1455.6
100	1577.4

t_s for further experiments instead of theoretical $k \times t_1$. As we can see, the average time taken by the engine to recognize the command increases linearly as the number of words in the vocabulary increases.

Table 2 shows the time taken by the "Parallel" approach vs the number of agents.

Table 3 shows the time taken for "Serial" (t_s) and "Parallel" (t_p) approaches calculated in milliseconds. From this table it can be inferred that as the number of models on a single node (k) decreases, there is a performance increase (t_p) as compared to the "Serial" approach. This trend is not followed if we decrease the value of k beyond a certain threshold. It can be explained by the fact that the time taken for mobile agents' traversal overshadows the gain achieved by the parallelization of the system.

The confusion matrix for the words recognized is shown in Table 4. Whenever the person controlling the robot speaks a command, the first task is to partition the command into different words. Once this is done, the words are converted to text individually using any algorithm.

The recognition accuracy of the HMM toolkit was found to be around 88%. However it is dependent on various factors like the algorithm used, the amount of data used for training, noise in the environment etc. and is not relevant to the purpose of this work. Any standard speech recognition API/Toolkit like HTK toolkit [18], CMU-Sphinx, Microsoft Speech API, etc. can be used in place of the HMM toolkit used here. Also, any number of algorithms can be used in speech engine in parallel. We just need to host the algorithm on some node and inform the speech engine about this new addition.

Table 3: Comparison of time taken by "Serial" (t_s) and "Parallel" (t_p) approaches

m	k	$\frac{m}{k}$	$\frac{m}{k} * t_2$	$k * t_1$	t_s	t_p
2500	250	10	145.2	337	**5020**	**482.2**
	50	50	751.2	138	**5020**	**889.2**
	25	100	1577.4	117	5020	1694.4
1000	100	10	145.2	180	**1082**	**325.2**
	50	20	300.6	138	**1082**	**438.6**
	20	50	751.2	112	1082	863.2
500	100	5	75.8	180	578	255.8
	50	10	145.2	138	578	283.2
	10	50	751.2	102	578	853.2
100	100	1	20.4	180	180	200.4
	20	5	75.8	112	180	187.8
	1	100	1577.4	80	180	1657.4

6. CONCLUSION

As expected, the time taken for serial conversion of speech-to-text commands grows linearly as the number of words increases. Whereas the same task takes less time after the number of words crosses a certain threshold if done in parallel. Many ways are possible to implement the parallel approach but this work uses mobile agents which ensures the scalability and decentralization of the system.

Any derived application can only be as accurate as the recognition algorithm implemented in the speech engine. But this approach supports adding new algorithms to the speech engine dynamically. Different nodes in the speech engine may host different recognition techniques [15] thereby improving the overall accuracy. Also, the positive/negative feedback mechanism ensures that better approaches are used more often than less accurate ones. For time-critical applications, we can use techniques that use fewer computational resources and time [14].

The feedback mechanism is necessary as speech utterances vary depending on pitch, volume, environment, etc. and fully accurate recognition algorithms are not yet available. Also, the machine learning approaches require feedback as part of the training mechanism to improve the accuracy of the system over time, hence the feedback is necessary. But in the current implementation, the feedback mechanism is a bottleneck and for applications that are not critical, it may be omitted.

Any new language can be added to the system dynamically just by hosting the new models of the commands.

The present model supports only one relay server at each robot controller node. Currently, work is underway to create a multi-threaded version of this relay server so that multiple robots can be connected to a single host. We also plan extensions to architecture for allowing more decentralization to increase robustness [16].

Data losses due to network errors have not been accounted for in this work. These losses may affect the usability of the system in noisy environments like in wireless ad hoc networks. A possible solution would be to implement reliability protocols between communicating nodes.

This work also does not handle the problems that may arise due to the inactivity of the client for a certain time. A solution might be to implement a timer that keeps track

Table 4: Confusion Matrix for Various Commands

Commands	ACQUIRE	FORWARD	BACKWARD	LEFT	RIGHT	STOP	RELEASE	GRAB	MOVE
ACQUIRE	9	1	0	0	0	0	0	0	0
FORWARD	2	7	1	0	0	0	0	0	0
BACKWARD	0	2	8	0	0	0	0	0	0
LEFT	0	0	0	8	2	0	0	0	0
RIGHT	0	0	0	0	8	2	0	0	0
STOP	0	0	0	0	0	10	0	0	0
RELEASE	0	0	0	0	1	0	9	0	0
GRAB	0	0	0	0	1	0	0	9	0
MOVE	0	0	0	0	0	0	0	0	10

of the inactivity duration. The choice of this threshold time is application dependent and is difficult to be agreed upon. Future work may include extending the proposed system to use natural language processing capabilities so that the robot-controlling person may talk to the robot in a more natural way rather than in a pure command-based method as described here.

7. ACKNOWLEDGMENTS

The authors wish to acknowledge UNICEF India and the DST, Government of India, for the funding provided under their FIST scheme, which greatly aided in the work reported herein.

8. REFERENCES

[1] D. Be, C. Gonzalez, M. Escalante, M. Garcia, C. Miranda, and S. Gonzalez. Wireless control LEGO NXT robot using voice commands. In *International Journal on Computer Science and Engineering (IJCSE)*, volume 03, pages 2926–2934, August 2011.

[2] M. Bramson, Y. Lu, and B. Prabhakar. Randomized load balancing with general service time distributions. *SIGMETRICS Perform. Eval. Rev.*, 38(1):275–286, June 2010.

[3] L. Cragg and H. Hu. Application of mobile agents to robust teleoperation of internet robots in nuclear decommissioning. In *Industrial Technology, 2003 IEEE International Conference on*, volume 2, pages 1214–1219 Vol.2, Dec 2003.

[4] L. Cragg and H. Hu. Mobile agent approach to networked robots. *The International Journal of Advanced Manufacturing Technology*, 30(9-10):979–987, 2006.

[5] B. House, J. Malkin, and J. Bilmes. The VoiceBot: A voice controlled robot arm. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 183–192, New York, NY, USA, 2009. ACM.

[6] S. Jha and S. Nair. A logic programming interface for multiple robots. In *Emerging Trends and Applications in Computer Science (NCETACS), 2012 3rd National Conference on*, pages 152–156, March 2012.

[7] B. H. Juang and L. R. Rabiner. Hidden markov models for speech recognition. *Technometrics*, 33(3):251–272, 1991.

[8] P. Liu, A. Chan, R. Chen, K. Wang, and Y. Zhu. Voice based robot control. In *Information Acquisition, 2005 IEEE International Conference on*, pages 5 pp.–, June 2005.

[9] J. Matani and S. Nair. Typhon - a mobile agents framework for real world emulation in prolog. In C. Sombattheera, A. Agarwal, S. Udgata, and K. Lavangnananda, editors, *Multi-disciplinary Trends in Artificial Intelligence*, volume 7080 of *Lecture Notes in Computer Science*, pages 261–273. Springer Berlin Heidelberg, 2011.

[10] D. Milojicic. Mobile agent applications. *IEEE Concurrency*, 7(3):80–90, 1999.

[11] O. Rogalla, M. Ehrenmann, R. Zollner, R. Becher, and R. Dillmann. Using gesture and speech control for commanding a robot assistant. In *Robot and Human Interactive Communication, 2002. Proceedings. 11th IEEE International Workshop on*, pages 454–459, 2002.

[12] K. Shrivastava, N. Singhal, P. K. Das, and S. B. Nair. A speech recognition client-server model for control of multiple robots. In *Proceedings of Conference on Advances In Robotics*, AIR '13, pages 69:1–69:6, New York, NY, USA, 2013. ACM.

[13] B. Tan. A distributed speech romote control system based on web service and automatic speech recognition. In X. Wan, editor, *Electrical Power Systems and Computers*, volume 99 of *Lecture Notes in Electrical Engineering*, pages 771–778. Springer Berlin Heidelberg, 2011.

[14] P. Verma. Resource Usage Analysis for Speech Recognition Techniques. Master's thesis, Department of Computer Science & Engineering, Indian Institute of Technology Guwahati, India, 2015.

[15] P. Verma and P. K. Das. i-Vectors in speech processing applications: A survey. *International Journal of Speech Technology*, 18(4):529–546, 2015.

[16] P. Verma, M. Gupta, T. Bhattacharya, and P. K. Das. Improving services using mobile agents-based IoT in a smart city. In *2014 International Conference on Contemporary Computing and Informatics*, page 107–111. IEEE, 2014.

[17] C. Yokoyama, M. Takimoto, and Y. Kambayashi. Cooperative control of multi-robots using mobile agents in a three-dimensional environment. In *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*, pages 1115–1120, Oct 2013.

[18] S. Young. The HTK hidden markov model toolkit: Design and philosophy. *Entropic Cambridge Research Laboratory, Ltd*, 2:2–44, 01 1994.