Controlling a Mobile Robot with Natural Commands based on Voice and Gesture

A.R. Fardana[1], S. Jain[1], I. Jovancevic[1], Y. Suri[1], C. Morand[1] and N.M. Robertson[1]

***Abstract*— This paper presents a real-time system for the control of a small mobile robot using combined audio (speech) and video (gesture) commands. Commercial hardware is used based on open-source code. Gesture is recognised using a dynamic time warp (DTW) algorithm using skeleton points derived from the RGB-D camera of the Kinect sensor. We present the integration of a faster parallel version of the DTW algorithm. Speech is recognised using a reduced-vocabulary HMM toolkit. Audio beam forming is exploited for localisation of the person relative to the robot. Separate commands are passed to a fusion centre which resolves conflicting and complementary instructions. This means complex commands such as "go there" and "come here" may be recognised without a complex scene model. We provide comprehensive analysis of the performance in an indoor, reverberant environment.**

I. INTRODUCTION

The development of robots has been significant in production, including factories [16]. The expectation is high for the development of intelligent robot systems that work cooperatively with human beings in daily life and in medical treatment and welfare. Smooth interfacing of human beings with robots is essential for the operation of robots by people in daily life. Anyone should be able to operate robots easily by giving instructions to the robot using gestures and voice instructions, just as people naturally communicate with each other. This interaction has been the subject of extensive research in recent years. As a result, an intelligent manipulator system using tracking vision has been developed [10]. The control algorithm for a service robot through the hand over task was proposed [1]. Human actions are utilized in human-robot interaction [13], [4]. The intelligent house for physically impaired people using hand pointing gestures has been developed [13]. The voice controller that can operate a robot by a voice instruction using a fuzzy control method was presented by Yoshidome [17]. This project presents an easy-to-build robot system using gesture and voice instructions. The goals of this work are to (a) build a robust, real-time and open system for controlling the motion of a robot by human speech and gesture commands. (The Robot we are using is the Turtlebot Roomba equipped with a Microsoft Kinect for both speech and gesture input); (b) integrate audio and gesture commands intelligently to achieve better recognition or enhanced functionality.

*This work is supported by the EU Commission under the 7th Framework Program (Grant FoF.NMP.2010-1-260101-LOCOBOT, www.locobot.eu).This work was supported by

[1] A.R. Fardana, S. Jain, I. Jovancevic, Y. Suri, C. Morand and N.M. Robertson are with VisionLab, Heriot-Watt University, Edinburgh, UK

Fig. 1. The successful recognition of "Go there" requires complementary audio and video inputs to be recognised by the system. Our setup is a Turtlebot with on-board computer and Kinect sensor.

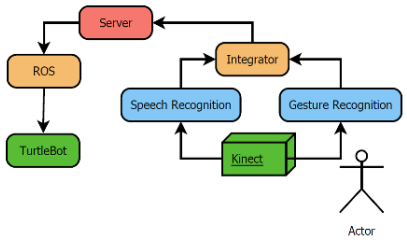

Fig. 2. System architecture

A. Contributions

In summary this paper makes the following research contributions. We develop a complete system for controlling the motion of a robot by human speech and by human gesture commands with low-cost off-the-shelf products. We present an improved version of DTW for faster gesture recognition based on parallelisation of the original DTW algorithm based on [5]. The robot can move to speaker source direction and perform the required action by exploiting the pointing angle from skeleton points alone i.e. no world model is required. We have integrated speech and gesture commands and successfully tested on the Turtlebot.

II. SYSTEM DESIGN FRAMEWORK

We introduce briefly the system design, which is illustrated in Figure 2. The user initiates a speech or/and gesture command according to a pre-defined ontology. The Kinect senses the audio signal (4 microphone beam forming array) or/and skeleton joint angles (RGB-D camera plus SDK algorithm [14]) and commands are sent to their respective modules. Speech and gesture recognition modules are run

on separate threads. Each module sends its command to the Integrator (Primary/Complementary Integrator). If the speech command is "Go", then the Complementary Integrator (CI) is called otherwise the Primary Integrator (PI) is used to combine the audio and video confidences. The Primary Integrator waits for the other command (from different modality, speech/gesture) for 3 seconds. If it receives the other command then it decides which command to execute depending on the confidence levels and whether or not commands from both modalities are recognized as same commands. If the other command does not arrive, it decides the action with one command only. Speech and Gesture run in Windows under Visual C++ environment. The Robot Operating System (ROS) runs on Linux on the robots. The Integrator sends a command (linear and angular velocity) over a wireless network. The client is the integrator module and HTTP server (also running Linux). The server receives the command and sends it to ROS and ROS executes command on the robots. The two machines are connected via wireless router and communicate via server-client scripts.

III. Automatic Speech Recognition (ASR)

The common way to recognize speech is the following: take waveform, split it into utterances vs. silences then try to recognize what is being said in each utterance. To do so, we want to take all possible combinations of words in the training corpus and try to match them with the sampled audio, choosing the best matching combination. We briefly re-state the main components of the speech recognition algorithm.

A. Feature extraction

The first step in attaching semantic meaning to the sequence of acoustic observations O is to convert an analog audio signal into a digital representation. During this analog-to-digital conversion, the amplitude of the signal is measured at fixed time intervals and translated to a floating point number. Because the information in this sequence of numbers is highly redundant, it is transformed into a reduced representation so that the relevant information is maintained but the data is less redundant. This step is called feature extraction [9]. The statistical model most often used to calculate the likelihood, is the Hidden Markov Model (HMM) [8]. (Although Neural Networks have also been popular [15].) An HMM consists of a finite number of states that are connected in a fixed topology. The input of the HMM, the feature vectors, are called observations. Each HMM state can 'emit' an observation from the observation sequence O with a certain probability defined by its Probability Distribution Function (PDF). The first observation must be emitted by a state that is defined to be one of the initial states. After this observation has been processed, one of the states that is connected to the initial state is chosen to emit the next observation. The probability that a particular transition from one state to another is picked, is modelled with the transition probability. Eventually all observations are emitted by a state that is connected to the state that emitted the previous observation and finally, final observation should be emitted by one of the final states. Since the actual path taken to create a specific state sequence is unknown to a theoretical observer therefore this type of Markov Model is called a Hidden Markov Model. Typically a three-state, left-to-right HMM topology is used to model phonemes.

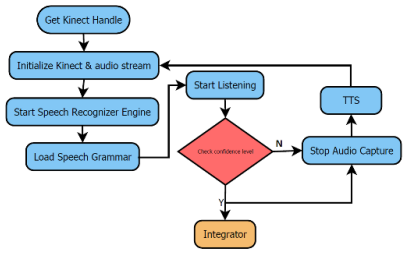

Fig. 3. We use the MSR speech recognition software on the detected audio stream.

In ASR, the probability distribution functions of the HMMs are often Gaussian Mixture Models (GMM). A GMM is a continuous function modelled out of a mixture of Gaussian functions where the output of each Gaussian is multiplied by a certain weight w. The Gaussian weights sum up to 1 and the Gaussian functions themselves are defined by their mean vector and covariance matrix. The Gaussian mixture model converts the feature vector into observation PDF model. The *a priori* probability P(W) where W is a sequence of words is calculated using a n-gram language model [7]. In n-gram models, for each possible sequence of n-1 words, the probability of the next word is stored. Because obtaining these statistics is only possible when a vocabulary is defined before creating n-gram model. In a typical system like LVCSR, HMM Lexicon (vocabularies) are defined that consist of more than 50,000 words. Some systems even use vocabularies with more than 300,000 words. With these large vocabularies the risk is minimized of not recognizing a word. The Viterbi algorithm decodes the observation using HMM lexicon and recognize individual words [12]. Finally these words are combined using N-gram grammar to comprehend running speech sentence. A more extensive discussion on Viterbi Decoder can be found [9].

B. Implementation and evaluation

Microsoft Speech Recognition (MSR) is freely-available. This has many advantages: it supports many language packs both for speech recognition and speech synthesis; its grammar and vocabulary are much more advanced; it requires no training for the models and its performance is much better than existing available open source software. Figure 3 shows how Speech Recognition is performed in our system. The Kinect handle is requested from the main application. Using Kinect handle we initialize the audio stream and start capturing audio. Once it is successful, we start the speech recognition engine. Once the engine is started, we load the grammar. Once we are ready to listen to the user, we wait for the commands to arrive. Once we receive commands, the MSR gives us the recognized word with its confidence. If the

	Forward	Left	Right	Faster	Slower	Stop	Backwards	Go	Unknown
Forward	1	0	0	0	0.2	0	0	0	0
Left	0	0.8	0	0	0	0	0	0	0
Right	0	0	1	0	0	0	0	0	0
Faster	0	0	0	0.8	0	0	0	0	0
Slower	0	0	0	0	0.7	0	0	0.05	0
Stop	0	0	0	0	0	0.75	0	0	0
Backwards	0	0	0	0	0	0	0.95	0	0
Go	0	0	0	0	0	0,1	0	0.9	0
Unknown	0	0.2	0	0.2	0.1	0.15	0.05	0.05	0

Fig. 4. The confusion matrix for speech command recognition. 4 users of different, non-UK nationality were used in the evaluation.

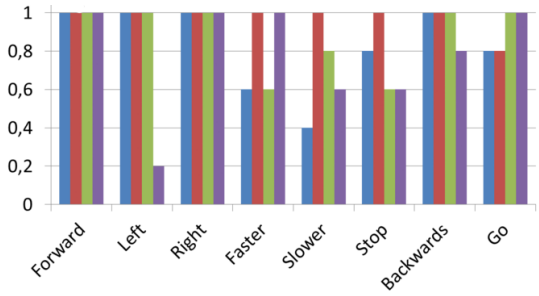

Fig. 5. Inter-person accuracy for speech recognition. Each bar for each command represents an individual's average accuracy.

confidence greater than a predefined threshold (0.5), we send the command, its confidence and source angle of user to the Integrator. Otherwise if confidence less than threshold, then we first stop the audio capture and the robot says "Again Please" (synthesised) and then the user should repeat the command.

In reverberant environments sound source localisation and recognition is problematic [11]. However this merely justifies the use of gesture in combination. In a normal office environment ($T_{60} < 0.2s$) we record 4 people saying each command 10 times. The confusion among speech commands for 4 users is shown in Figure 4. The inter-person variability is shown in Figure 5. Overall recognition rate is 85%.

IV. Gesture Recognition

We propose that our gesture commands can be distinguished by the specific position of the hand with respect to the head. The input for gesture recognition is the hand trajectory and the hand can be tracked in various ways. A common way to do it is by using joints: hands, shoulders, elbows, head, etc. In order to do that, we need estimation of joints positions in image. Shotton *et al.* propose estimation of joints based on single depth image [14]. Their method uses decision trees for per-pixel classification to body parts and then Mean-Shift [6] for estimation of joints. In this work, we are using 2D joint information shown in Figure 6. In each frame, we extract 2D position of: left hand joint, right hand joint, left shoulder joint, right shoulder joint and head. Gesture is represented by temporal vector A of (x, y) positions with respect to coordinate frame shown below. In each frame, hand (x, y) position is recorded.

Video segmentation is the process of determining frame sequence in which one full gesture is contained. (Segmentation of the gesture in video should not be confused with image processing segmentation where we segment image regions.) Gesture segmentation is done by imposing constraint: starting and ending position of the hand should be resting position: both arms lying along the body. It is position shown in Figure 6, (*left*).

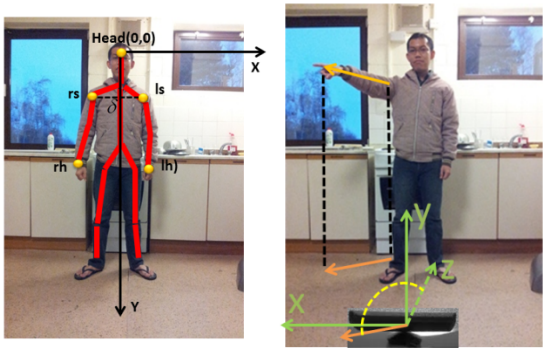

Fig. 6. Extracted joints for use in gesture recognition (left) and the direction vector via projection of arm vector onto the ground plane (right)

Once a gesture is segmented the next task is to compute distance between new gesture trajectory and each of the learned gestures we have in our database and match the new gesture with closest one. Any distance (Euclidean, Manhattan etc.) which aligns the i^{th} point on one time series with the i^{th} point on the other will produce a poor similarity score. A non-linear (elastic) alignment produces a more intuitive similarity measure, allowing similar shapes to match even if they are out of phase in the time axis and have different number of elements [2]. To do this we use Dynamic Time Warping (described below). This is needed because gesture will be done with different speed every time, even if done by the same person, and especially if done by *different* person. We compute confidence of this winning gesture and send this information to the Integrator module, if confidence is above the predefined threshold, which is empirically set. (Note that we are not using mirroring mentioned in the original work [2], i.e. if gesture is learned by using certain hand, it has to be done with same hand in order to be recognized. Our robot must recognize "Left" and "Right" as separate commands.)

A. DTW and speed-up via parallel processing

The computation cost of DTW is in quadratic time. On top of this is the time complexity of searching all possible matched sequences in database under DTW distance. Therefore to reduce computation time we implement a new parallel approach [5] which is novel in this context. To demonstrate the superiority of the proposed DTW parallelization, we conducted extensive experiments using synthetic data sets. For comparisons, we also implemented the sequential version of the DTW code. We measure the run time of both methods. Our results confirm (below) that the parallel code gives significant (factor 8 at least) performance improvement over sequential code on 8 cores Intel Core i7 microprocessor, at 3.1 GHz with 8 GB RAM.

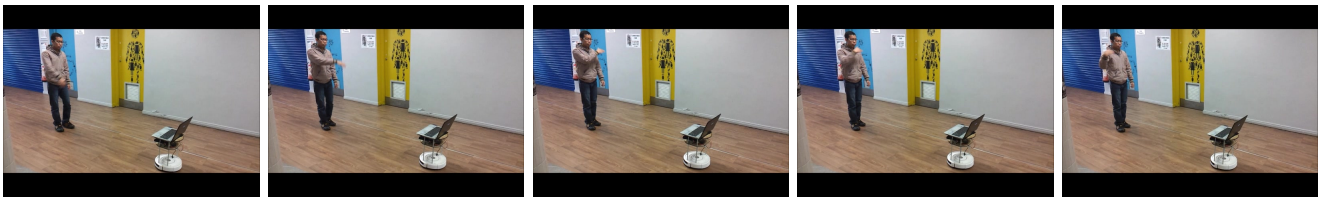

Fig. 7. Stopping the robot via gesture commands (*left to right*).

Dynamic time warping (DTW) is a time series alignment algorithm developed originally for speech recognition. It is also called non-linear, elastic alignment. It produces a more intuitive similarity measure, allowing similar shapes to match even if they are of different lengths. See Figure 8.

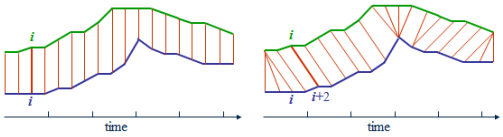

Fig. 8. DTW concept

It aims to align two sequences of typically different lengths by iteratively warping time axis. Warping is repeated until optimal match is found. The optimal match is the one that makes sum of differences between matched features minimal. We can interpret this algorithm as filling the table (matrix) and then finding the optimal path in matrix (optimal match). The matrix has dimensions $n \times m$ where n and m are lengths of two sequences. Sequences are arranged along bottom and left edge of the matrix. See figure 9.

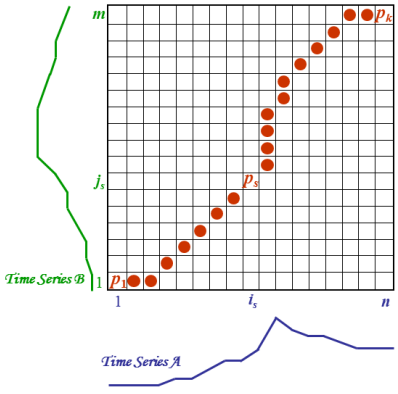

Fig. 9. Finding optimal path

The number of possible paths is exponentially growing with number of elements of sequences and dynamic programming is used to solve this search problem. The algorithm starts with filling the element $(1,1)$ with distance between first elements of two sequences $X(x_1,...,x_n)$ and $Y(y_1,...,y_m)$:

$$g(1,1) = d(y_1, x_1)$$

where $d(x_i, y_j)$ is difference between two elements. It is absolute difference in case when elements are scalars and for example Euclidean distance in case where elements are vectors. After filling the first element, rest of the elements are filled by using recursive formula:

$$g(i,j) = d(i,j) + min\{\, g\,(i,j-1)g(i-1,j-1)g(i-1,j)$$

Finally, DTW distance between sequences X and Y is given by

$$D(X,Y) = \frac{g(n,m)}{n+m}$$

Optimal path is found by tracing back from element (n,m) to element $(1,1)$. We move from element (i,j) to one of its three neighbors:$(i,j-1), (i-1,j-1)$ or $(i-1,j)$ which has lowest value g. Eventually we will reach element $(1,1)$. The original algorithm is filling one row at a time. It can be seen that filling table is computationally most demanding task in DTW algorithm. Hence it is useful to make parallel version of this phase. Since it is recursive algorithm and there is data dependency, it is not possible to make filling rows parallel. One row can be filled only after previous one is filled. Also, one element in a row can be filled only after previous one is filled.

However, by changing order of filling table elements, parallelization is possible. In figure 10(a) data dependency of table filling is shown. Since (i,j) depends only on elements $(i-1,j)$, $(i,j-1)$ and $(i-1,j-1)$, it is possible to compute simultaneously elements $(i+1,j)$ and $(i,j+1)$ once (i,j) is known. So, if computation flow is proceeding in diagonal direction, parallel DTW can be achieved. In each iteration, we can fill one whole diagonal of a matrix. Flow is depicted in figures 10 (b, c, d, e, and f). In first iteration, algorithm only calculates element $(1,1)$ which enables computing elements $(1,2)$ and $(2,1)$ in next iteration. So, one thread is active in first iteration, two threads are active in second iteration, etc. This is first phase of the algorithm where number of calculated elements in each iteration is equal to number of step. Maximum level of parallelization is achieved in step m when m threads are active. m threads stay active as long as number of elements in diagonal is m (see figure 10(e)). That is second phase where number of elements is m. In third phase from step $n+1$ to step $n+m-1$, number of threads is again changing through steps. In each $step$, number of elements is $m+n-step$.

As can be concluded from Figures 11 and 12 the speed up increases linearly with increasing number of cores.

B. Computing direction from gesture

We now compute vector determined by shoulder and hand joint. The 3D vector in the Kinect frame-of-reference is

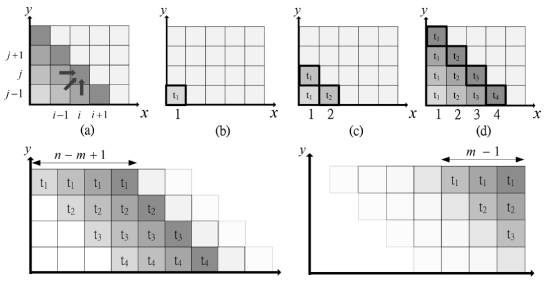

Fig. 10. Filling table for the DTW algorithm used in gesture matching.

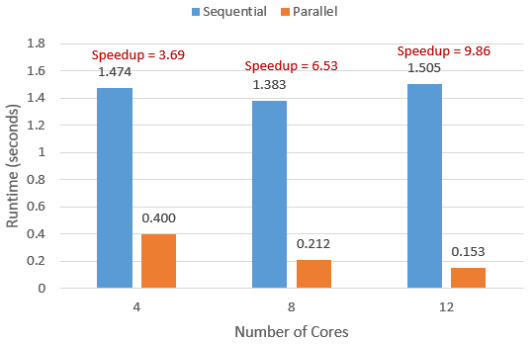

Fig. 11. The effect of increasing number of cores on runtime.

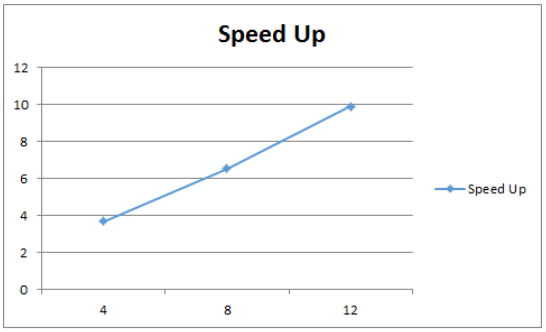

Fig. 12. Linear speed-up with varying numbers of cores.

	Forward	Left	Right	Faster	Slower	Stop	Backwards	Non-Gesture
Forward	0.85	0	0	0	0	0	0	0
Left	0.15	0.9	0	0	0	0.05	0	0
Right	0	0	1	0	0	0	0	0
Faster	0	0	0	0.95	0	0	0	0
Slower	0	0.05	0	0	0.95	0	0	0
Stop	0	0	0	0	0	0.8	0	0
Backwards	0	0	0	0	0	0	0.9	0
Non-Gesture	0	0.05	0	0.05	0.05	0.15	0.1	0

Fig. 13. The confusion matrix for gesture command recognition.

obtained, which for this purpose we can consider as the robot frame since it is mounted on the robot. Then we project this vector to the XZ plane, which is approximately parallel with the ground plane. The angle between positive part of the Z-axis and this new vector is computed, as shown in Figure 6, (*right/*). This is then the angle for which robot must rotate to be oriented in the direction the user wishes it. (For a negative angle, robot will rotate to the left and for positive angle it will rotate to the right.) We then rotate the robot for the required amount of time (angular velocity and angle are known). Another variation is to use elbow and hand joints, but hand-shoulder vector has shown better results in our experiments.

C. Evaluation of gesture recognition

Once more we ask 4 volunteers to repeat each gesture command 10 times, after a series of training exercises. The confusion matrix among gesture commands alone is shown in Figure 13 and the inter-person variation in Figure 14. In our experiments, overall accuracy was 89%. An illustrative example of the "Stop" command being successfully executed is given in Figure 7.

V. Joint use of gesture and speech commands

The software implemented so far does not return a distribution over all possible commands for speech and gesture. For a larger vocabulary this would be impractical on a real-time setup. We thus propose two ways to fuse commands coming from speech and gesture, using what probability information that is available, while retaining performance. The two fusion strategies are as follows: (a) Primary Integrator : this integrator makes command recognition more robust when command overlap i.e. the same command is given both in gesture and speech. It cross verify the commands from both modules; (b) Complementary Integrator : here one module acts as a *complementary* action to the other, as shown in Figure 1. Direct pairing between speech and gesture commands: is achieved for Forward, Backward, Left, Right, Slower, Faster and Stop commands. In Primary integration we compare information coming from two modules. Gesture module gives distribution: confidences for all commands present in database. However, the speech module returns only the maximum likelihood (ML) result (command with highest confidence) and its confidence. So we take the ML, or winning, command from both modules and compare them. We may also integrate speech and gesture commands in a complementary fashion e.g. "Go there": saying "Go" and pointing in a specific direction ("There"). We assume here that speech command is given before gesture command. This prerequisite was necessary to avoid synchronization conflict

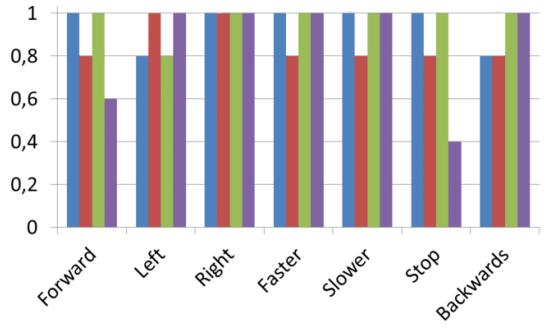

Fig. 14. Inter-person accuracy for gesture recognition. Each bar for each command represents an individual's average accuracy over 10 trials.

Test case	Expected outcome
1. User says, “Go”, then gestures in direction	Moves to the specified direction
2. User gestures in direction, then says “Go”	Robot executes command according to the gesture
3. User gestures and speaks same command	Robot fuses the commands and executes
4. User gestures and speaks different command	Winner-takes-all and execute
5. User gives single command	Robot executes the command

TABLE I

THE TEST CASES FOR COMBINED GESTURE AND SPEECH RECOGNITION ON THE PLATFORM. IN OUR EXPERIMENTS WITH 4 PARTICIPANTS WE ACHIEVE 82% SUCCESS IN RECOGNITION OF JOINT COMMANDS.

between speech and gesture commands.

VI. EXPERIMENTATION, RESULTS AND EVALUATION

The average execution time - using the trained algorithms with no optimisation - for a speech command is 2.2s and for gesture 3.3s. When gesture computation is parallelized, execution time is reduced dramatically to an average of 0.25s. Successful operation of the robot requires the user to be with $\pm 60^o$ of the centre of the microphone array and with 4m of the camera. We now wish to consider accuracy and in particular whether speech and gesture facilitate improved recognition or increased capability over a single modality. As shown in Figures 4 and 13, recognition accuracy is good. Inter-person variability is high for speech (shown in Figure 5) and less so for gesture (Figure 14). The test cases are shown in Table I. Accuracy is 82% over all of these cases. We show in Figure 15 that conflicting commands may be resolved correctly in the majority of cases by the combination of audio and video.

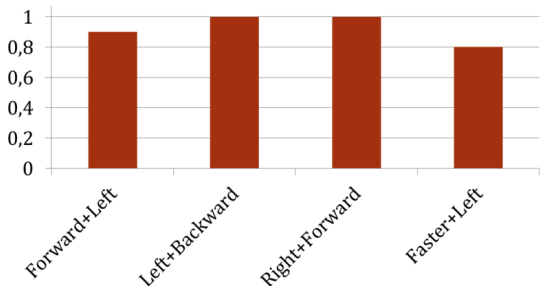

Fig. 15. Percentage accuracy of the correct command recognition in the presence of conflicting speech and gesture commands.

VII. FUTURE WORK

There are many avenues for future development, including developing a probabilistic inference model which uses an instruction “grammar”, e.g. the Start command must appear before the Stop command etc. Noise reduction for audio module and making it more robust to high reverberation of the type found in e.g. factory environments is necessary but this may require de-coupling audio and video from the Kinect sensor. Directional instructions are currently associated to the “Forward” and “Go there” command only, and there is scope for widening the use of this function.

REFERENCES

[1] Agah A, Tanie K Human interaction with a service robot: mobile-manipulator handing over an object to a human Proc. of the IEEE Int. Conf. on Robotics and Automation, pp 575-580, 1997

[2] P.Barratini, C.Morand, N.M.Robertson A Proposed Gesture Set for the Control of Industrial Collaborative Robots Proc. 21st IEEE Int. Symp. Robot and Human Interactive Communication, Paris, France, September 2012

[3] Bien ZZ, Park K, Bang W, Stefanov DH LARES: An Intelligent Sweet Home for Assisting the Elderly and the Handicapped CWUAAT 2002, pp 43-46, 2002

[4] B.Burger,I.Ferrane, F.Lerasle, G.Infantes Two-handed gesture recognition and fusion with speech to command a robot Autonomous Robots Journal, Volume 32 Issue 2, February 2012, Pages 129-147. Kluwer Academic Publishers Hingham, MA, USA

[5] J. Chang and M.Y. Yeh Monitoring multiple streams with dynamic time warping using graphic processors Workshop Notes of the International Workshop on Parallel Data Mining, pp. 11-19, 2011

[6] Comaniciu, D., Peter M Mean Shift: A Robust Approach Toward Feature Space Analysis IEEE Transactions on Pattern Analysis and Machine Intelligence (IEEE) 24 (5): 603-619, 2002

[7] Christopher D. Manning, Hinrich Schetze Foundations of Statistical Natural Language Processing MIT Press: 1999.

[8] B. H. Juang; L. R. Rabiner Hidden Markov Models for Speech Recognition Technometrics, Vol. 33, No. 3, pp. 251-272, 1991

[9] D.Jurafsky and J.H. Martin Speech and Language Processing: An Introduction to Natural Language Processing Computational Linguistics, and Speech Recognition. Prentice-Hall, New Jersey, 2000

[10] Kawarazaki N, Kashiwagi N, Hoya I, Nishihara K Manipulator Work System Using Gesture Instructions Journal of Robotics and Mechatronics, 14 (5): 506-513; 2002

[11] Ulrich Klee, Tobias Gehrig and John McDonough Kalman Filters for Time Delay of Arrival-Based Source Localization EURASIP Journal on Advances in Signal Processing 2006, 2006:012378 doi:10.1155/ASP/2006/12378

[12] Luk, R.W.P.; R.I. Damper Computational complexity of a fast Viterbi decoding algorithm for stochastic letter-phoneme transduction IEEE Trans. Speech and Audio Processing 6 (3): 217-225, 1998

[13] Pavlovic VI, Sharma S, Huang TS Visual interpretation of hand gestures for human-computer interaction: A review IEEE Trans. on Pattern Analysis and Machine Intelligence, 19 (7): 677-695, 1997

[14] Shotton J, Fitzgibbon A, Cook M, Sharp T, Finocchio M, Moore R, Kipman A, Blake A Real-Time Human Pose Recognition in Parts from Single Depth Images Computer Vision and Pattern Recognition (June 2011)

[15] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang Phoneme recognition using time-delay neural networks IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 37, pp. 328-339, 1989

[16] Christian Wögerer, Harald Bauer, Martijn N. Rooker, Gerhard Ebenhofer, Alberto Rovetta, Neil Robertson, Andreas Pichler LOCOBOT - Low Cost Toolkit for Building Robot Co-workers in Assembly Lines ICIRA (2), 2012, pp. 449-459, http://dx.doi.org/10.1007/978-3-642-33515-0_45

[17] Yoshidome T, Kawarazaki N, Nishihara K A Robot Operation By A Voice Instruction Including A Quantitative Expression Proc. Of The 5TH Franco-Japanese Congress and 3RD European-Asian Congress On Mechatronics, pp 123-126, 2001