A Whisper ROS Wrapper to Enable Automatic Speech Recognition in Embedded Systems

Andrés A. Ramírez-Duque
School of Computing Science
University of Glasgow
Glasgow, UK
Andres.Ramirez-Duque@glasgow.ac.uk

Mary Ellen Foster
School of Computing Science
University of Glasgow
Glasgow, UK
MaryEllen.Foster@glasgow.ac.uk

ABSTRACT

Automatic Speech Recognition (ASR) is a technology that aims to automatically identify patterns in human speech and transcribe them into text. The performance of modern ASR systems is rapidly increasing, but most current systems require significant computing resources to run and often make use of cloud computing, making them difficult to deploy in the embedded context that is often necessary for an interactive robot. OpenAI has recently released Whisper, a robust and open-source speech recognition system; in this paper, we show how a lightweight Whisper model can be integrated into the ROS ecosystem using high-performance inference functions, enabling automatic speech recognition to run offline on embedded hardware, giving the potential for ASR to be integrated into a much wider set of HRI contexts.

CCS CONCEPTS

• **Computer systems organization → Embedded systems**; *Redundancy*; Robotics.

KEYWORDS

Speech Recognition, ROS, High-performance computing, Embedded hardware

ACM Reference Format:
Andrés A. Ramírez-Duque and Mary Ellen Foster. 2023. A Whisper ROS Wrapper to Enable Automatic Speech Recognition in Embedded Systems. In *Proceedings of HRI 2023 Workshop on Human-Robot Conversational Interaction (HRCI 2023).* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The recent rapid development of conversational AI systems, particularly those making use of a massive training approach [8], has the potential to contribute significantly to social robotics, by providing the opportunity for such robots to support fluent, open-domain conversation. Recent, rapid developments in natural language processing and speech processing that could be applied in HRI include GPT-3 [6] and ChatGPT[1] by OpenAI, along with Vall-E [18], Wav2Vec [3] and Whisper [16]. It could be said that a point is approaching where a distinction could be drawn between "before" and "after" research on conversational robotics.

However, although the potential for such systems is growing, it is still necessary to put these off-the-shelf models into practice and fine-tune a series of low-level details to develop physically embodied agents with conversational skills close to those of humans. In principle, conversational robots could be developed using similar techniques to those used in embodied virtual agent implementation [2]. However, the physical embodiment of robots presents a particular challenge: specifically, the embedded hardware often used in robots has some limitations on computing capabilities that make their development difficult. These limitations, such as the computational cost, the motherboard processor, the range of the sensors, and the limitations of Internet access in crowded spaces, must be taken into account when designing conversational robots.

Modern ASR systems tend to require significant processing power, either on the local computer (often using a GPU) or using cloud-based resources. However, the robots that could most benefit from the addition of ASR are often equipped with low processing capacities without access to a GPU and cannot easily be updated; these robots may also be deployed in contexts where internet access is unreliable or nonexistent. Deploying a large machine-learning model such as Whisper requires a hardware-focused approach.

We present such a solution: we have integrated Whisper [16] with the Robot Operating System (ROS) [15] using a high performance inference package to enable offline Automatic Speech Recognition (ASR) using embedded hardware. The package has been implemented and tested on a Raspberry Pi and is able to provide near-real-time speech recognition on this embedded platform. The implemented software can be freely downloaded from Github.

2 AUTOMATIC SPEECH RECOGNITION

Automatic Speech Recognition (ASR) is a technology that aims to automatically identify patterns in human speech and transcribe them into text [1]. Traditional methods focused on manual feature extraction and conventional techniques such as Gaussian Mixture Models (GMM) [19], and Hidden Markov Models (HMM) [11]. Modern techniques use Deep Neural Networks (DNNs) [10] and Transformers [17]. The textual transcription of the spoken utterance may provide a more natural human-robot interaction, but it also increases the complexity of the robot deployment.

[1]ChatGPT: https://openai.com/blog/chatgpt/

2.1 Whisper by OpenAI

Whisper is a robust speech recognition system developed by OpenAI using over 680,000 hours of supervised data collected from the web and trained through weak supervision techniques. Additionally, thanks to the massive database and the training techniques that they used, the model can be a multilingual and multitask ASR system [16]. The structure of the model itself is not new, as it uses an encoder-decoder transformer model [17]. The model was enhanced to add the multitask training format using a set of special tokens that serve as task specifiers or classification targets; as part of development, a sample of the previously transcribed text was fed back into the model so that it would learn from the context that accompanies the transcription.

OpenAI released this ASR system as open source and made the trained models available through a family of five models ranging from Tiny to Large, consisting of 39M to 1550M parameters respectively [16]. The official Whisper implementation[2] requires Python 3.9.9 and PyTorch 1.10.1 [14]. Despite the relatively small size of the Tiny model, the performance of this model on embedded platforms such as the Raspberry Pi is poor: it is not possible to perform transcriptions anywhere close to real-time.

2.2 High-performance Inference

Whisper.cpp [9] is an open-source framework that uses high performance coding to implement one of the currently best-known alternative methods for performing Whisper model inference. This implementation uses plain C/C++ structure code optimized to run the encoder-decoder model only using CPU resources. The approach features low memory usage, requires no allocated memory at run time, and integrates Arm NEON, AVX, and VSX intrinsic support depending on the processor architecture. It also integrates different Basic Linear Algebra Subprograms (BLAS) libraries for accelerated tensor operation routines.

To date, this minimalist implementation has incorporated some OpenAI model features, such as initial prompts, temperature support, Greedy decoding strategy, and BeamSearch [12]. The Whisper.cpp GitHub repository[3] provides a number of example applications, including a web browser, a system to generate karaoke-style transcriptions, a real-time transcription of the raw capture, and a basic voice assistant example.

3 WHISPER ROS WRAPPER

From a purely technical perspective, integrating an ASR system into ROS seems straightforward, as ROS is an inherently flexible ecosystem that supports multiple languages. However, depending on the target platform, the model requirements, and the final task of social interaction, porting an ASR may require a bit more effort and time. We have integrated Whisper.cpp into ROS using embedded platforms widely used by the developer community, such as the Raspberry Pi. These embedded systems are characterised by being portable and having a good processing vs power usage ratio and an adequate size that allows them to be discreet and easily integrated into commercial robotic platforms.

[2]OpenAI Whisper website: https://openai.com/blog/whisper/
[3]Whisper.cpp: https://github.com/ggerganov/whisper.cpp

3.1 ROS Service Integration

The first thing we set out to answer is what might be the best way to integrate a system like Whisper into ROS. The answer to the above question depends on the typical requirements of a robotic system and the interaction task. Normally, a conversational robot must simultaneously execute the reading of sensors (touch, cameras, LiDAR, microphones), the control of actuators, and various decision-making processes and communication interface or direct control with the user. In this context, the clear approach for integrating an ASR system into ROS would be a process that runs on-demand, that is, a service or action.

The implementation of ROS Service and Actions is simple: they act as a callback blocking function that is executed when a signal is triggered, but in ROS, this kind of process is enriched by the synchronous *RPC-style* communication that enables bidirectional connection with any ROS entity [15]. Services are used to run fast processes, so they are ideal for a robot that might need to switch tasks quickly. Thus, in this work we implement the inference process as a ROS service, making it accessible to all nodes and using a microphone to record audio samples.

Our implemented service stores speech samples in an audio buffer which can be filled via the ROS Topic */audio* using a standard *AudioDataStamped* message. The service also accepts audio samples coming directly from GStreamer audio pipelines without further ROS message conversion. In order to maintain compatibility with Whisper.cpp, it can also use SDL2[4] to provide low-level access to the audio. In all the above cases, the audio must be sampled at 16000 Hz. Once the transcription of the audio segment is finished, the service responds to the requested signal using the *std_srvs/Trigger* message to notify either *True/False* if the service is executed successfully and a *String* message containing the transcribed audio.

The ROS service provides three modes of operation: in the first one, it runs as a streaming method for a period of T seconds and transcribes any available audio samples in the input. In the second mode, the service listens passively until it recognises a passphrase to trigger detection and transcribe a predetermined (N) number of words (N output tokens). In the final mode, the service is used to transcribe and match keywords contained in a vocabulary and respond with a command associated with each keyword.

Additionally, we include a function in the service that allows us to reset and relaunch the inference process to modify some sensitive parameters, such as the number of tokens used by the encoder to represent the audio context. It is important to note that in embedded platforms such as the Raspberry Pi, controlling the context sizes changes the encoder-decoder sizes and therefore the inference time. This could allow us to balance features such as precision and computational cost at run-time, which may be advantageous depending on the phase of the human-robot interaction that requires ASR inference: for example, if the system is expecting an answer to a Yes/No question, then the system could be reconfigured with a smaller vocabulary.

For this work, the service was implemented on a Raspberry Pi 4 8GB (ARM Cortex A72). The ROS package was built by compiling

[4]Simple Direct Media Layer: https://www.libsdl.org/

the Whisper.cpp module using Neon-Arm[5] routines and OpenBlas[6] to optimise performance. To initially test the performance of the inference process, an 11-second audio sample was transcribed using the default amount of context tokens (1500 tokens), and load time, encoder time, and total inference time were measured. The ROS Service takes about 2000*ms* to load the model, 7600*ms* to code the sample, and around ten seconds to finish the whole inference process on Raspberry. Once the number of tokens was reduced to 768 and 512 tokens, the encoder ran two and three times faster, respectively. Thus, it is possible to use the ROS service implementation to transcribe three to seven seconds audio samples with near real-time behaviour. The developed package can be found on GitHub at https://github.com/andres-ramirez-duque/ros_stream.

3.2 Challenges

The result of this simplified implementation is promising, but the main purpose is to help as a proof of concept to analyse the challenges that remain to provide robots with robust ASR systems like Whisper that are able to run on embedded hardware to support the development of conversational robots. A remaining challenge for this overall goal is to address the timing control and turn-taking among the speakers, including the robot: the current service allows us to take care of the resources, but it requires an additional strategy that executes the service and triggers it at the correct time.

In addition, the current model does not yet implement other typical ASR functions, such as word-level timestamps, speech activity detection [5], speaker change detection, voice overlay detection, and diarization [4]. In the current state, it is not possible to identify and transcribe what each person is saying separately. The above limits implementation in complex scenarios such as crowded spaces.

4 CONCLUSION

We have presented the development of a Whisper ROS wrapper composed of a lightweight Whisper model and high-performance inference functions to enable automatic speech recognition on embedded hardware, with the goal of supporting conversational robot systems. We have described the characteristics of the inputs, methods, and outputs of the ROS service and have demonstrated that it is capable of real-time recognition; we have also outlined some remaining challenges. We consider that this implementation could be used as a proof of concept for future research, and are currently exploring integrating this server into a child-robot system designed for use in paediatric emergency departments [7, 13].

ACKNOWLEDGMENTS

The authors wish to acknowledge the SSHRC-UKRI Canada-UK Artificial Intelligence Initiative (UKRI grant ES/T01296/1) for financial support of this project.

REFERENCES

[1] Hanan Aldarmaki, Asad Ullah, Sreepratha Ram, and Nazar Zaki. 2022. Unsupervised Automatic Speech Recognition: A review. *Speech Communication* 139 (4 2022), 76–91. https://doi.org/10.1016/j.specom.2022.02.005

[2] Merav Allouch, Amos Azaria, and Rina Azoulay. 2021. Conversational Agents: Goals, Technologies, Vision and Challenges. *Sensors* 21 (12 2021), 8448. Issue 24. https://doi.org/10.3390/s21248448

[3] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems* 33 (2020), 12449–12460.

[4] Hervé Bredin and Antoine Laurent. 2021. End-to-end speaker segmentation for overlap-aware resegmentation. In *Proc. Interspeech 2021*.

[5] Hervé Bredin, Ruiqing Yin, Juan Manuel Coria, Gregory Gelly, Pavel Korshunov, Marvin Lavechin, Diego Fustes, Hadrien Titeux, Wassim Bouaziz, and Marie-Philippe Gill. 2020. pyannote.audio: neural building blocks for speaker diarization. In *ICASSP 2020, IEEE International Conference on Acoustics, Speech, and Signal Processing*.

[6] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. https://doi.org/10.48550/ARXIV.2005.14165

[7] Mary Ellen Foster, Patricia Candelaria, Lauren J. Dwyer, Summer Hudson, Alan Lindsay, Fareha Nishat, Mykelle Pacquing, Ronald P. A. Petrick, Andrés A. Ramírez-Duque, Jennifer Stinson, Frauke Zeller, and Samina Ali. 2023. Co-design of a Social Robot for Distraction in the Paediatric Emergency Department. In *Companion of the 2023 ACM/IEEE International Conference on Human-Robot Interaction*. Stockholm, Sweden. https://doi.org/10.1145/3568294.3580127

[8] Asbjørn Følstad, Theo Araujo, Effie Lai-Chong Law, Petter Bae Brandtzaeg, Symeon Papadopoulos, Lea Reis, Marcos Baez, Guy Laban, Patrick McAllister, Carolin Ischen, Rebecca Wald, Fabio Catania, Raphael Meyer von Wolff, Sebastian Hobert, and Ewa Luger. 2021. Future directions for chatbot research: an interdisciplinary research agenda. *Computing* 103 (12 2021), 2915–2942. Issue 12. https://doi.org/10.1007/s00607-021-01016-7

[9] Georgi Gerganov. 2023. Whisper.cpp. https://github.com/ggerganov/whisper.cpp.

[10] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, and Brian Kingsbury. 2012. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *IEEE Signal Processing Magazine* 29 (11 2012), 82–97. Issue 6. https://doi.org/10.1109/MSP.2012.2205597

[11] B. H. Juang and L. R. Rabiner. 1991. Hidden Markov Models for Speech Recognition. *Technometrics* 33 (8 1991), 251. Issue 3. https://doi.org/10.2307/1268779

[12] Jungo Kasai, Keisuke Sakaguchi, Ronan Le Bras, Dragomir Radev, Yejin Choi, and Noah A. Smith. 2022. Beam Decoding with Controlled Patience. https://doi.org/10.48550/ARXIV.2204.05424

[13] Alan Lindsay, Andrés A. Ramírez-Duque, Ronald P. A. Petrick, and Mary Ellen Foster. 2022. A Socially Assistive Robot using Automated Planning in a Paediatric Clinical Setting. In *Proceedings of the AAAI Fall Symposium on Artificial Intelligence for Human-Robot Interaction (AI-HRI)*. https://doi.org/10.48550/arXiv.2210.09753

[14] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, et al. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf

[15] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. 2009. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*.

[16] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2022. Robust speech recognition via large-scale weak supervision. *arXiv preprint arXiv:2212.04356* (2022).

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[18] Chengyi Wang, Sanyuan Chen, Yu Wu, Ziqiang Zhang, Long Zhou, Shujie Liu, Zhuo Chen, Yanqing Liu, Huaming Wang, Jinyu Li, Lei He, Sheng Zhao, and Furu Wei. 2023. Neural Codec Language Models are Zero-Shot Text to Speech Synthesizers. https://doi.org/10.48550/ARXIV.2301.02111

[19] Y. Zhang, M. Alder, and R. Togneri. 1994. Using Gaussian mixture modeling in speech recognition. *Proceedings of ICASSP '94. IEEE International Conference on Acoustics, Speech and Signal Processing* i, I/613–I/616. https://doi.org/10.1109/ICASSP.1994.389219

[5]Neon-Arm: https://www.arm.com/technologies/neon
[6]OpenBLAS: https://github.com/xianyi/OpenBLAS