Endoscopic Procedures Control Using Speech Recognition

Simão Afonso, Isabel Laranjo, Joel Braga, Victor Alves, José Neves

***Abstract* —** In this paper it is presented a solution for replacing the current endoscopic exams control mechanisms. This kind of exams require the gastroenterologist to perform a complex procedure, using both hands simultaneously, to manipulate the endoscope's buttons and using the foot to press a pedal in order to perform simple tasks such as capturing frames. The last procedure cannot be accomplished in real-time because the gastroenterologist needs to press an additional programmable button on the endoscope to freeze the image and then press the pedal to capture and save the frame. The presented solution replaces the pedal with a hands-free voice control module and it is capable of running on the background continuously without human physical intervention. This system was designed to be used seamlessly with the MyEndoscopy system that is being tested in some healthcare institution and uses the PocketSphinx libraries to perform real-time recognition of a small vocabulary in two different languages, namely English and Portuguese.

***Keywords*—**Automatic Speech Recognition, Hidden Markov Models, PocketSphinx, SphinxTrain, Endoscopic Procedures

I. Introduction

NOWADAYS it is accepted by most healthcare professionals that information technologies and informatics are crucial tools to enable a better healthcare practice. The Pew Health Professions Commission (PHPC) recommended that all healthcare professionals should be able to use information technologies [1]. The technological evolution has led to an enormous increase in the production of objective diagnostic tests and a decrease on the reliance of more subjective problem solving methods, which should increase the quality of the service provided, and can even be seen as a consequence of the increased accountability of healthcare institutions in relation to the legislation [2].

EsophagoGastroDuodenoscopy (EGD) and Colonoscopy occupy relevant positions amongst diagnostic tests, since they combine low cost and good medical results. The current endoscopic exams require the gastroenterologist to perform a complex procedure using both hands simultaneously to manipulate the endoscope's buttons and using the foot to press the pedal in order to perform such simple tasks as capturing frames. The last procedure cannot be accomplished in real-time because the gastroenterologist needs to press an additional programmable button on the endoscope to freeze the image and then press the pedal to capture and save the frame [3]. This approach to the problem is not optimal and raises several new issues, such as limiting the movements of everyone involved and requiring the gastroenterologist to perform a complex procedure, distracting him/her from the task at hand. A new hands-free interface that allows for a richer control scheme would solve some of the existing snags.

A novel approach to this problem consists of adding a voice recognition module to the system, providing a hands-free control. This module, called *MIVcontrol*, will be integrated into the device called *MIVbox* (more details are given in section 3).

The main goal of the *MIVcontrol* module is to create a simple speech recognition system for recognizing a very small vocabulary of simple pre-determined commands. The recognized commands are used to control the *MIVacquisition*, creating a hands-free control system that should be able to replace the current solution. This system can perform frame capturing in real-time, without the need to use any extra buttons.

The system should be speaker-independent and have a very low error rate, even on noisy environments, and it should be able to capture audio from a microphone continuously, so that it can run in the background without human intervention. This will require automatic word segmentation, to make recognition possible.

The rest of the paper is organized as follows: in section 2 it is presented a review of related work in the area of speech recognition, from its theoretical foundations to practical systems already being used. In section 3 is outlined the overall system architecture and how it integrates with the *MyEndoscopy* system. In section 4 is presented specific details about the implementation of the solution, whereas in section 5 the methodology used in the study is exhibited. Finally, the results and their assessment are presented in section 6 and 7, followed by conclusions in section 8.

II. Related Work

Automatic Speech Recognition (ASR) is a process by which a computer processes human speech, creating a textual representation of the spoken words. This process has two main areas of study, i.e. discrete speech and continuous speech. Discrete speech is useful for the creation of voice command interfaces, while continuous speech, also known as dictation, mimics the way two humans communicate. Though the ultimate objective of having a system capable of

This work is funded by ERDF - European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT - *Fundação para a Ciência e a Tecnologia* (Portuguese Foundation for Science and Technology) within project PEst-OE/EEI/UI0752/2014.

S. Afonso, I. Laranjo, J. Braga and J. Neves are in the Computer Science and Technology Center (CCTC), University of Minho, Braga, Portugal (simaopoafonso@gmail.com, isabel@di.uminho.pt, jneves@di.uminho.pt, joeltelesbraga@gmail.com)

V. Alves is in the Computer Science and Technology Center (CCTC), University of Minho, Braga, Portugal (corresponding author to provide e-mail: valves@di.uminho.pt).

recognizing everything anyone can say in multiple languages has yet to be achieved, research has been focused on smaller-scale approaches [4].

A. Theoretical Foundations

Aymen et al. [5] presented the theoretical foundation of Hidden Markov Models (HMM) that underpin most modern implementations of automatic speech recognition. The authors present the distinction between speech recognition, which aims to recognize almost anyone's speech, and voice recognition, which creates systems trained to particular users. The model is constructed based on a large *corpus* of recorded speech, annotated with the respective transcription. The HMM requires three different sub-models:

1) The acoustic model consists of different features for each utterance the system recognizes;
2) The lexical model tries to identify sounds considering the context;
3) The language model identifies the higher-level characteristics of speech, such as words and sentences.

The HMM searches the model for similar patterns that fit into the given audio input, producing probable matches. The HMM's advantages over previous learning algorithms consists of easy implementation on a computer and automated training without human intervention. This stems from the fact that it is assumed that in short-time ranges the process is stationary, vastly reducing the computational effort [5].

B. Implementations

There are several HMM implementations, but the most advanced are the HTK Toolkit [6] and the CMU Sphinx system [7].

Hidden Markov Model Toolkit (HTK) is a set of libraries used for research in automatic speech recognition, implemented using HMM. The HTK codebase is owned by Microsoft, but managed by the Cambridge University Engineering Department. Since HTK has been largely abandoned, since the last release (v3.4.1) was made in 2009, the CMU Sphinx system is getting more attention from the speech recognition community [6].

The original SPHINX was the first accurate Large Vocabulary Continuous Speech Recognition (LVCSR) system, using HMM as its underlying technology, that managed to be speaker independent [7]. The next version, SPHINX-II, was an improved version that was both faster and more accurate, created by most of the same authors, using HMM as its underlying technology. It was developed, from the beginning, as an open source project, creating a community around it [8]. The next version, SPHINX-III, is an offline version of the previous systems, with a different internal representation to allow for greater accuracy. The signals go through a much larger amount of pre-processing before they even reach the recognizer [7]. Current hardware is capable of running the recognizer for SPHINX-III in almost real-time, but it is not suitable to processing in such conditions. SPHINX-4 is a complete rewrite to create a more modular and flexible system that can accept multiple data sources elegantly. It is a joint venture with Mitsubishi Electric Research Laboratories and Sun Microsystems, using the Java programming language. As with the third version, its intended use is offline processing, not real-time applications [9]. *Vertanen* [10] tested both the HTK and the Sphinx systems with the Wall Street Journal (WSJ) corpus and found no significant differences in error rate and speed. This conclusion is corroborated by other researchers [11].

Huggins-Daines et al. [12] optimized CMU Sphinx II for embedded systems, primarily those with ARM architecture. To balance the loss of precision required in other optimizations, the CMU Sphinx III Gaussian mixture model was back-ported. They managed to have a 1000-word vocabulary running at 0.87 *times real-time* on a 206 MHz embedded device, with an error rate of 13.95% [12]. "*Times real-time*" is a notation that indicates the amount of time required to process live data. In this case, the system can process 1 second of data in 0.87 seconds, which makes it suitable to real-time recognizing. This work has lead to the creation of the PocketSphinx project, an open source initiative to continue this work. This project is in active development, and it has bindings for C and Python [13].

C. Practical systems

Vijay [14] studied the problem of phonetic decomposition in lesser-studied languages, like Native American and Roma language variants, using the PocketSphinx system. While the system does not implement the complex rules of these languages, it is possible to leverage the existing system to recognize unknown languages, using a relatively simple lookup table that maps sounds to phones [15]. *Varela et al.* [15] adapted the system to the Mexican Spanish language. The authors created a language and an acoustic model, based on an auto-attendant telephonic system, and achieved an error rate of 6.32% [15]. The same process was followed for other languages, like Mandarin [16], Arabic [17], Swedish [18]. These examples show that the PocketSphinx system is flexible enough so that it is relatively easy for people with phonetics training to extend it to other languages.

Harvey et al. [4] researched how ASR systems could be integrated with their project aimed at developing a device to help the elderly, both inside and outside the home. The authors identified the following challenges associated with ASR systems used for voice command interfaces [4]:

1) Important differences between users;
2) Similarity between certain sounds;
3) Short words provide less data for the system to analyze, which may lead to increased error rates;
4) Different recognition languages lead to variable error rates using the same system.

Specifically to their project, the authors found that medical conditions, which frequently affect the elderly, create different speech patterns and their tolerance to errors is quite low. With that in mind, the authors leveraged the Sphinx library for its maturity and features. Focusing on the creation of models and general optimization tasks, the authors managed to create a multilingual system that has a 2-second processing time on embedded systems, with error rates above 70% [4].

Kirchhoff et al. [19] suggested other methods to improve the ASR systems' performance. One proposal consists of replacing the current feature-extraction algorithms with others specially designed to discriminate certain sound classes depending on the intended use, or through the use of noise reduction algorithms, which can improve the data

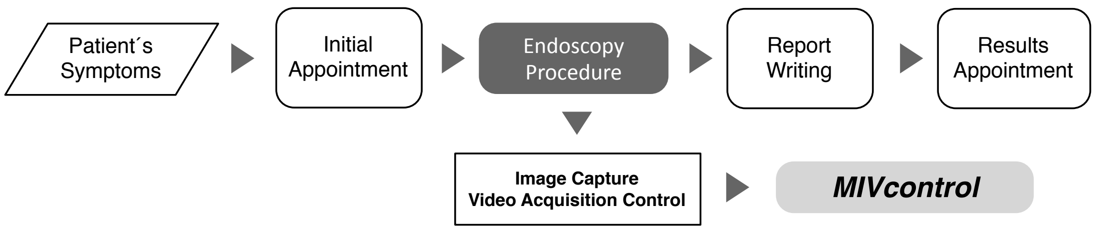


Fig. 1 Workflow for a gastroenterology medical appointment

analysis and increase the models' accuracy, using the same data collection routines. A different approach is collecting more out-of-band information to increase the amount of data available to the system. This might include different processing front-ends for feature extraction (although this may be of limited use) or even non-acoustic data, such as visual information [19].

III. Architecture

As referred before, the gastroenterologist needs to use a pedal to capture and save the frame, and with the proposed solution the pedal is replaced with a hands-free voice control module, called *MIVcontrol*. This module was developed to tackle the problems that healthcare professionals face when performing an endoscopic procedure. This module is part of the *MIVbox* device, which is integrated in the *MyEndoscopy* system.

MyEndoscopy is the name of the global system developed by *Laranjo et al.* [21], which groups several *MIVboxes*, which are scattered by various healthcare institutions. The main goal of *MyEndoscopy* is to link different entities and standardize the patient's clinical process management, to promote the sharing of information between different entities [21].

The *MIVbox* device has a web-based distributed architecture and it is capable of acquisition, processing, archiving and diffusion of endoscopic procedure results [21]. The main goal of the *MIVcontrol* module is to replace the pedal, currently used by gastroenterologists to capture interesting frames, by voice commands that interact directly with the *MIVacquisition* module. The MIVacquisition module receives the video directly from the endoscopic tower and provides it to all the MIVbox modules [20].

In **Fig. 1** is presented a simple workflow describing the moments that occur in a gastroenterology medical appointment, in the healthcare institution, that results in an Endoscopy Procedure. The *MIVcontrol* module can be seamlessly integrated into the current workflow by allowing the gastroenterologist to control the *MIVacquisition* module by using voice commands during the endoscopic procedure.

In **Fig. 2** is presented the overall system architecture, as a component of the *MIVbox* device. The *MIVcontrol* module uses the live audio streaming from a microphone to recognize commands and send them to the *MIVacquisition* module.

The process that leads to the creation of a model is presented on **Fig. 3**. It uses a corpus of pre-labeled audio data to create a speech model that can be used in the *MIVcontrol* module. This speech model is a combination of acoustic and language models.

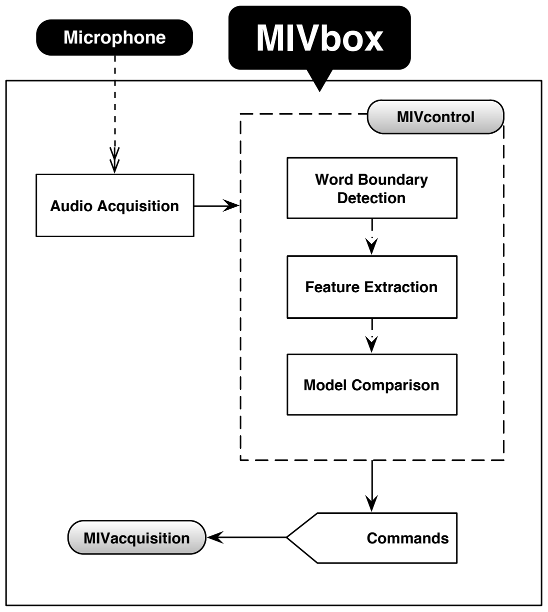


Fig. 2 *MIVcontrol* global architecture

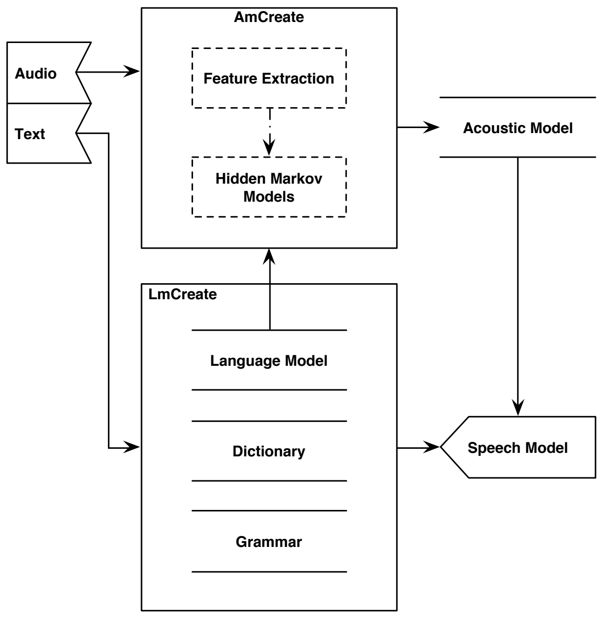


Fig. 3 *MIVcontrol* model training procedure

The process is split in two main sub-processes: creation of the textual model, named *lmCreate*, and the creation of the acoustic model, named *amCreate*. Since the acoustic model requires parts of the textual model, it must be generated last. The *lmCreate* process creates the textual model based on the textual data in the corpus, while the *amCreate* process analyzes the pre-recorded audio data using the same feature extraction steps used in the *MIVcontrol* module, and uses HMM to learn how to classify the commands contained in the language model.

IV. Implementation

The creation of the speech model used in the *MIVcontrol* module, from higher to lower level comprises three different phases, namely language model, dictionary and acoustic model.

A. Language Model

The language model is a high-level description of all valid phrases (i.e. combination of words) in a certain language. Statistical language models try to predict all the valid utterances in a language, by combining all the recognized words into every possible combination [22]. Context-Free Grammars are restricted forms of a language model, that restrict the recognized phrases to a predetermined set, and discard those that do not fit that model [23].

The decision to adopt a certain language model depends mostly on its intended application. While statistical language models are useful for open-ended applications, like dictation and general-purpose recognition, context-free grammars are suitable for specific applications, like command-and-control systems.

SphinxBase requires the grammar to be defined in Java Speech Grammar Format (JSGF), which is a platform-independent standard format to define context-free grammars, using a textual representation so that it can be human-readable [24]. The statistical language model is automatically created based on the command list.

B. Dictionary

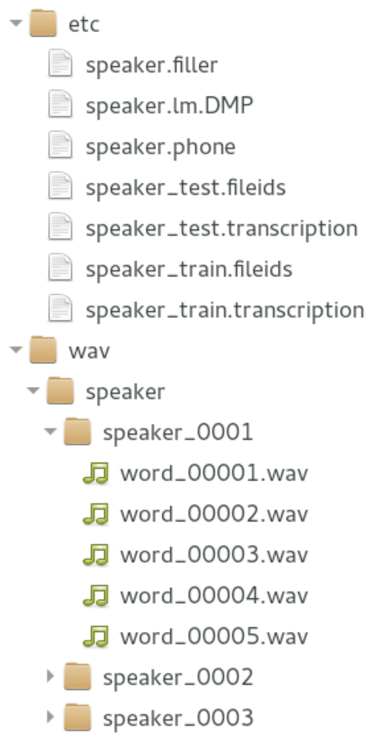
The dictionary is a map between each command and the phonemes it contains. A phoneme is defined as the basic unit of phonology, which can be combined to form words. Its internal representation consists of using the ARPAbet to represent phonemes as ASCII characters. The ARPAbet does not allow representing the entire International Phonetic Alphabet (IPA), but it is sufficient for small vocabularies, such as the one required by this application [25].

Since the list of required commands is small, all the dictionaries used were created manually.

C. Acoustic Model

The acoustic model is trained using *SphinxTrain* and maps audio features to the phonemes they represent, for those included in the dictionary. The training performed by SphinxTrain requires previous knowledge of the dictionary and a transcription for each utterance, in order to map each utterance to its corresponding phonetic information. It also requires the data to be in a particular audio format. In order to minimize clerical errors and cut the time need to analyze the data to a minimum, all the technical considerations and index building were abstracted away in a script referred to as *amCreate*.

SphinxTrain requires the folder tree presented on **Fig. 4**, where "**model**" denotes the model name.



```
etc
    speaker.filler
    speaker.lm.DMP
    speaker.phone
    speaker_test.fileids
    speaker_test.transcription
    speaker_train.fileids
    speaker_train.transcription
wav
    speaker
        speaker_0001
            word_00001.wav
            word_00002.wav
            word_00003.wav
            word_00004.wav
            word_00005.wav
        speaker_0002
        speaker_0003
```

Fig. 4 Folder tree required by *SphinxTrain.*

The folder directory has two top folders, namely the *etc* and the *wav*.

The *etc* folder contains all the metadata and configuration parameters needed to train the acoustic model, as well as the dictionary. It contains both a list of all the phonemes used in the model and a list of filler phonemes, such as silences, that should be ignored. It also has a list of all the files to be used during both training and testing phases, as well as a mapping between each audio file and its corresponding transcription. This mapping corresponds to the labeled data to be the input to the HMM.

The *wav* folder simply contains all the collected data, as audio files, organized in subfolders by speaker identification, with a subfolder for each set of uttered commands.

The system processes continuous audio in real-time, splits it in commands and produces a line of text for each recognized command. If the spoken command is not recognized, an empty line is produced. The *MIVcontrol* module runs on the *MIVbox*.

The audio picked up by the microphone is stored in a memory buffer. The first pre-processing stage involves splitting the incoming audio into different utterances, or sets of words, by tracking silent periods between them. To account for noise present during recording, any audio with volume below a certain threshold is considered a silence.

Each segmented utterance then goes through a similar process. The audio is processed creating a set of features, and then the Semi-Continuous HMM finds the most likely utterance contained in its dictionary. This is the final output, corresponding to a command given to the system.

If there is Internet access and the data to be recognized is not sensitive, it is possible to use an online speech recognition service, such as the Google Speech API [26], as a fallback mechanism.

V. Methodologies

The parameters that have a bigger impact on the model's accuracy are the number of tied states used in the HMM and the number of Gaussian mixture distributions, so testing will focus on this parameters. Before testing begins, the data is split randomly between training and testing stacks, with the testing stack receiving 10% of the data. That same data is tested varying both the number of tied states in the HMM and the number of Gaussian distributions. The accuracy of the model is represented as a Word Error Rate (WER), which combines both false positives and false negatives into a single metric. This is done because this module acts as *middleware*, being used by other modules on a global system. The context where the commands are spoken can then be considered, which is not evaluated here. Furthermore, this methodology is consistent with the literature on the subject.

The results were obtained on a computer with 2 GB of RAM and an Intel Celeron CPU, with two cores and a clock speed of 1.10GHz. The operative system used was Fedora 19, 32 bits version. The compiler used was gcc v4.8.2, using PocketSphinx v0.8 and SphinxBase v0.8, both from the official repositories. The training used CMUcltk v0.7, compiled from source, and SphinxTrain v1.0.8, from the official repositories. The training data was collected with the built-in laptop microphone, in both noisy and quiet conditions, to better correspond to the concrete use-case.

VI. Results

The audio corpus in which the system was tested contained two languages, Portuguese and English, with a total of 1405 recordings, totaling 25 minutes of speech, recorded by 5 female and 7 male speakers. To test this model, *SphinxTrain* tried to predict the contents of the testing data using the model created with the training data. The main parameters that can be tweaked are the number of tied states in the HMM and the number of Gaussian mixtures distributions.

The effect of the number of tied states in the HMM is shown on **Table 1**.

Table 1 Effect of the number of tied states on the WER

Number of Tied States	Errors	WER (%)
5	112	33.6
10	85	28.7
25	109	32.2
50	100	35.6
100	86	26.8
150	108	33.3

The effect of the number of Gaussian mixtures distributions on the error rate is shown on **Table 2**.

Since the trained model is small, the differences in processing time are negligible. That defined the optimal conditions for training 8 Gaussian mixture distributions with 100 tied states in the HMM. With this configuration, the system classified the Portuguese model with 11.22 % WER and the English model with 4.55 % WER.

Table 2 Effect of the number of Gaussians on the WER

Number of Gaussians	Errors	WER (%)
1	152	17.1
2	172	17.0
4	134	14.6
8	142	14.1

VII. Discussion

As a proof-of-concept, this system managed to create a voice recognizer for a very small vocabulary to be used as a command and control system, leveraging the capabilities of the CMU Sphinx project. It was created as an alternative to cloud-based solutions, such as Google Speech API. In a medical environment, cloud-based solutions pose certain challenges that might degrade their performance, such as increased communications security, a need to keep recurring costs on non-medical equipment to a minimum, and also privacy and legal reasons on systems that deal with sensitive data. Having a system that can be installed inside the healthcare institutions' network without external dependencies is a plus for the reasons presented above.

PocketSphinx was based on work done for SPHINX II, which was not designed as a real-time recognizer. With all the optimizations it has received, it is possible to use it in real-time with acceptable performance, even in underpowered computers.

VIII. Conclusion and Future Work

In summary, this paper presents an automatic speech recognition system designed specifically to solve a problem that affects gastroenterologists. The system is capable of running on the background continuously without human physical intervention, and so it is capable of replacing the pedal and buttons commonly used in current endoscopic systems. It was designed to be used seamlessly with the MyEndoscopy system that is being tested in some healthcare institutions.

The next step will involve improving the integration with the *MyEndoscopy* system, including a more robust testing phase, which is facilitated by the fact that PocketSphinx is a cross-platform library.

To increase the usefulness of the system, it is important to collect more data, particularly with different voice features. It is also possible to apply newer training algorithms to the same data, and test how they affect the models created. The *SphinxTrain* suite was created using CMU Sphinx recognizers, but there are more recent projects that are able to produce models compatible with PocketSphinx-based recognizers. Those newer systems may generate better models with the same data.

Reference

[1] E. H. O'Neil, "Recreating Health Professional Practice for a New Century," San Francisco, CA, 1998.

[2] N. Summerton, “Positive and negative factors in defensive medicine: a questionnaire study of general practitioners.,” *BMJ*, vol. 310, no. 6971, pp. 27–29, Jan. 1995.

[3] J. M. Canard, J.-C. Létard, L. Palazzo, I. Penman, and A. M. Lennon, *Gastrointestinal Endoscopy in Practice*, 1st ed. Churchill Livingstone, 2011, p. 492.

[4] A. P. Harvey, R. J. McCrindle, K. Lundqvist, and P. Parslow, “Automatic speech recognition for assistive technology devices,” in *Proc. 8th Intl Conf. Disability, Virtual Reality & Associated Technologies*, Valparaíso, 2010, pp. 273–282.

[5] M. Aymen, A. Abdelaziz, S. Halim, and H. Maaref, “Hidden Markov Models for automatic speech recognition,” in *2011 International Conference on Communications, Computing and Control Applications (CCCA)*, 2011, pp. 1–6.

[6] S. Young, G. Evermann, D. Kershaw, G. Moore, J. Odell, D. Ollason, V. Valtchev, and P. Woodland, “HTK FAQ.” [Online]. Available: http://htk.eng.cam.ac.uk/docs/faq.shtml. [Accessed: 03-Feb-2014].

[7] K.-F. Lee, H.-W. Hon, and R. Reddy, “An overview of the SPHINX speech recognition system,” *IEEE Trans. Acoust.*, vol. 38, no. 1, pp. 35–45, 1990.

[8] X. Huang, F. Alleva, H.-W. Hon, M.-Y. Hwang, K.-F. Lee, and R. Rosenfeld, “The SPHINX-II speech recognition system: an overview,” *Comput. Speech Lang.*, vol. 7, no. 2, pp. 137–148, Apr. 1993.

[9] P. Lamere, P. Kwok, E. Gouvea, B. Raj, R. Singh, W. Walker, M. Warmuth, and P. Wolf, “The CMU SPHINX-4 speech recognition system,” in *IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP 2003), Hong Kong*, 2003, vol. 1, pp. 2–5.

[10] K. Vertanen, “Baseline WSJ Acoustic Models for HTK and Sphinx: Training recipes and recognition experiments,” *Cavendish Lab. Univ. Cambridge*, 2006.

[11] G. Ma, W. Zhou, J. Zheng, X. You, and W. Ye, “A comparison between HTK and SPHINX on chinese mandarin,” in *IJCAI International Joint Conference on Artificial Intelligence*, 2009, pp. 394–397.

[12] D. Huggins-Daines, M. Kumar, A. Chan, A. W. Black, M. Ravishankar, and A. I. Rudnicky, “Pocketsphinx: A Free, Real-Time Continuous Speech Recognition System for Hand-Held Devices,” *2006 IEEE Int. Conf. Acoust. Speed Signal Process. Proc.*, vol. 1, pp. I–185–I–188, 2006.

[13] D. Huggins-Daines, “PocketSphinx v0.5 API Documentation,” 2008. [Online]. Available: http://www.speech.cs.cmu.edu/sphinx/doc/doxygen/pocketsphinx/main.html. [Accessed: 20-Feb-2014].

[14] V. John, “Phonetic decomposition for Speech Recognition of Lesser-Studied Languages,” in *Proceeding of the 2009 international workshop on Intercultural collaboration - IWIC ’09*, 2009, p. 253.

[15] A. Varela, H. Cuayáhuitl, and J. A. Nolazco-Flores, “Creating a Mexican Spanish version of the CMU Sphinx-III speech recognition system,” in *Progress in Pattern Recognition, Speech and Image Analysis*, vol. 2905, A. Sanfeliu and J. Ruiz-Shulcloper, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 251–258.

[16] Y. Wang and X. Zhang, “Realization of Mandarin continuous digits speech recognition system using Sphinx,” *2010 Int. Symp. Comput. Commun. Control Autom.*, pp. 378–380, May 2010.

[17] H. Hyassat and R. Abu Zitar, “Arabic speech recognition using SPHINX engine,” *Int. J. Speech Technol.*, vol. 9, no. 3–4, pp. 133–150, Oct. 2008.

[18] G. Salvi, “Developing acoustic models for automatic speech recognition,” 1998.

[19] K. Kirchhoff, G. A. Fink, and G. Sagerer, “Combining acoustic and articulatory feature information for robust speech recognition,” *Speech Commun.*, vol. 37, no. 3–4, pp. 303–319, Jul. 2002.

[20] J. Braga, I. Laranjo, D. Assunção, C. Rolanda, L. Lopes, J. Correia-Pinto, and V. Alves, “Endoscopic Imaging Results: Web based Solution with Video Diffusion,” *Procedia Technol.*, vol. 9, pp. 1123–1131, 2013.

[21] I. Laranjo, J. Braga, D. Assunção, A. Silva, C. Rolanda, L. Lopes, J. Correia-Pinto, and V. Alves, “Web-Based Solution for Acquisition, Processing, Archiving and Diffusion of Endoscopy Studies,” in *Distributed Computing and Artificial Intelligence*, vol. 217, Springer International Publishing, 2013, pp. 317–24.

[22] P. Clarkson and R. Rosenfeld, “Statistical language modeling using the CMU-cambridge toolkit,” in *5th European Conference on Speech Communication and Technology*, 1997, pp. 2707–2710.

[23] A. Bundy and L. Wallen, “Context-Free Grammar,” in *Catalogue of Artificial Intelligence Tools*, A. Bundy and L. Wallen, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1984, pp. 22–23.

[24] A. Hunt, “JSpeech Grammar Format,” 2000.

[25] R. A. Gillman, “Automatic Verification of Hypothesized Phonemic Strings in Continuous Speech,” Arlington, Virginia, 1974.

[26] B. Ballinger, C. Allauzen, A. Gruenstein, and J. Schalkwyk, “On-Demand Language Model Interpolation for Mobile Speech Input,” *Elev. Annu. Conf. Int. Speech Commun. Assoc.*, no. September, pp. 1812–1815, 2010.