Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG

Bachelorthesis

Multi channel speech command recognition for humanoid robots

Mehrkanalige Sprachbefehlserkennung für humanoide Roboter

Thomas Walther

September 13, 2018

MIN-Fakultät

Fachbereich Informatik

Studiengang: Software System Entwicklung

Matrikelnummer: 6539274

Erstgutachter: Prof. Dr.-Ing. Timo Gerkmann

Zweitgutachter: Marc Bestmann

Betreuer: Dr.-Ing. Martin Krawczyk-Becker

Abstract

Speech recognition for humanoid robots is a not well explored topic in the RoboCup to this date. This thesis uses the Kaldi framework for the speech recognition. For an easy implementation on the different operation systems of the robots a self created python tool is used. The training of the speech recognition is extended with different audio modifications of the training data that are explained in detail. The results of the Word-Error-Rate, Accuracy, Insertions, Substitutions and Deletions that Kaldi provides get analyzed and evaluated. It is shown that the speech recognition with the chosen modifications is able to work in noisy environments and can be used for further researches and for practical usage. The capabilities of the python tool are explained and there is a short introduction of the configuration files.

Contents

List of Figures

1 Introduction

Besides the economic use of robots they are very present in today's scientific life. The technologies that are needed to evolve the robots to a new level to be more human-like are given with neural networks and much smaller, and they are more efficient processing units. The RoboCup Federation [1] has an objective to beat the most recent world champion in soccer in 2050 with autonomous robots. The normal rules for humans are aimed at, so the robots are not allowed to benefit from non-human possibilities like wireless communication networks or shared information. They will just have two ears, two eyes and the ability to speak, so they can communicate human-like.

Until now the robots can walk, recognize their surroundings and identify goals, balls and sometimes robots. First attempts for audio recognition are made to recognize the whistle of the referees but no robot in the RoboCup can speak or understand even simple speech commands.

To overcome the current lack of hearing and understanding, in this bachelor thesis a speaker-dependent speech recognition system is developed. Using the Kaldi framework for the speech recognition and an own toolkit written in python to train different persons is supposed to give the robots a basic ability to hear and understand single commands. We will just work at the basics what means that every speaker can train an acoustic model without any other speakers needed but this is not replacing a real speaker recognition. It will just help to reduce the accepted commands by unknown speakers like surrounding spectators what is an intended result because they that could try to mislead the robots. The perfect result would be a recognition system that can distinguish between known human team members, team robots and spectators but that is something for future works.

1.1 The goals of the thesis

The goals of this thesis are:

1. Training and Testing of an acoustic model that recognizes speech commands under difficult circumstances.
2. Evaluation of the benefits of different modifications on the training data with respect to the speech recognition performance, specifically in challenging, i.e. noisy and reverberant scenarios.

3. Creation of a toolkit with python for the training of different team members of the RoboCup teams who don't have deeper knowledge of Kaldi.

1.1.1 Difficulties for humanoid robots in today's RoboCup

Speech recognition strongly depends on the circumstances and the training. While under good circumstances, with a good training and the language models of today's recognition tools they can understand most spoken words, the reliability strongly decreases with disturbing noises, like wind, noises of a crowd or echo, as they appear in big halls.

We have to consider that the robots will be playing in big halls with spectators and we have to adjust our training to the acoustic model.

1.1.2 Evaluating the benefits of the data augmentations

After having added all different modifications to the training we will evaluate the results with different test settings and compare the gains of every modification. The test settings are separated by adding the modifications one by one to the tests and run them on every training set. The results show which modification improves or reduces the recognition rate. For the evaluation we use the normal exports of Kaldi.

1.1.3 Python tool for usage in robots

The tool will be for training and creating acoustic models without having the knowledge of Kaldi to do so. A running Kaldi implementation is assumed and the path to it has been set. Then the tool offers to create new speakers and record the known commands. Every recorded command will automatically be modified by different room impulse responses, noises, stretching and compressing. The new audio files will be inserted and all necessary configuration files will be edited.

Outline

After the introduction we will start in Chapter 2 with a short overview of the used technologies and the environment, including the robot and its technology, and some needed background information about Kaldi, HMMs, Acoustic Models and Phones.

In Chapter 3 we will explain the different modifications that are made to the recorded commands to improve the acoustic model and explain what they should improve.

Chapter 4 contains the evaluation of our results of all test and training combinations and we will see, if they come up to our expectations.

In Chapter 5 the python toolkit will be explained as well as the way we hope to help all the RoboCup teams with the implementation of speech command recognition or speech recognition overall.

Finally, the outcome of the thesis is summarized and future directions for further improvements are pointed out in Chapter 6.

2 Fundamentals

2.1 Humanoid Robots

In the RoboCup we can find different leagues for different types of robots. Every type has its own advantages and disadvantages and is built for certain researches of possible tasks and abilities. Most Leagues are for humanoid robots and we are working in the RoboCup Humanoid Kid Size League. The humanoid robot has to become as close to the human as possible. That includes sensing, moving and interacting with the environment.

At the moment the senses of the used robots are one camera and two microphones combined in one web cam module [13]. The team of the university of Hamburg is constantly improving the robots and in the future there will be more of them equipped with human-like shaped eyes and ears replicates. That will additionally help to improve the visual and audio recognition. A deeper insight in the possibilities of using spoken language for the communication between robots can be found in the Bachelor Thesis written in German by Maike Paetzel [2].

Figure 2.1: Robot in the field [14]

2.1.1 Computing Power

Size and cooling are very important aspects in the construction of a humanoid robot. The robots get hardware upgrades whenever possible but their computing power is limited. At present the computing power is comparable to gaming tablets and low energy working laptops. To handle the necessary computing power of neural networks for the visual recognition more specialized APUs are needed. The robots of the team of the university of Hamburg got this upgrade around early 2018.

2.1.2 ROS - Robot Operating System

Practically all teams use the Robot Operating System (ROS) that was developed by Willow Garage in 2007 [11]. ROS is designed to work on different computing systems but officially supports only Debian and Ubuntu. It is open source and strongly relies on community work. It provides a big library of software modules that are needed for common robotic tasks.
A deeper insight can be found in the online documentation [12].

2.1.3 Microphones

The web cam module [13] is located on the head of the robot and can not move on its own (see Figure 2.1). The microphones are directed to the front, so we don't have an optimal equivalent for human hearing. Noises from the back or the side will not have the same quality as the noises from the front because of the surrounding case and the directed design of the web cam.
A deeper insight into the topic of the recognition of a robot in 360 degree around him can be found in the Bachelor Thesis by Robert Keßler [3].

2.2 Commands

We want to train nine commands that are helpful to control the actions of the robots in the field.

Schuss - Shot
Rückzug - Retreat
Angriff - Attack
Links - Left
Rechts - Right
Hinten - Behind
Vorne - Ahead
Verteidigung - Defense
Aufstellung - Lineup

2.3 Speech Recognition

The most common technologies for speech recognition are Deep Neural Networks (DNN) [4] and the Hidden Markov Model (HMM) [5]. In short, a neural network needs a huge amount of data for training and it is not expected that it will work well with only a few hundred of training recordings. Every trained speech recognition is working better with more data, to a certain point, but the HMM is able to achieve decent results with much less data than a neural network.

The reason why we are training just a few speakers is based on the necessity that other people's commands must be ignored. Our project focuses on recognizing and understanding a few, simple commands. With just a small vocabulary of commands we expect that the training of a HMM-based recognizer will result in a decent acoustic model.
Apart from the problem of getting enough open source data of the used commands for a DNN we would need to implement a speaker recognition. Without the speaker recognition it would be impossible to test the system in a real game. As soon as the surrounding people start yelling the commands, the robots would follow these orders. Because the HMM is able to be trained with much less data, we can focus on one or a few people, the team members, to train the recognition. We expect that this give us the opportunity to test the speech recognition in a real environment without a speaker recognition.

Besides, the functionality of the HMM is easier to understand and analyze. That helps to select the training data more specifically for our special field of application. On this basis, we opted for the HMM instead of the DNN.

2.4 Recognition Frameworks

In the beginning we compared the different recognition tools Kaldi, HTK, CMUSphinx and Julius. Our targeted criteria were an open source licence, support of the HMM, a slim and fast program core for the recognition, a good documentation and an active community. While HMM is supported by all of them, HTK has a proprietary licence. Julius is mainly developed in Japan and is fully supporting Japanese but is just in development for English. Finally we had to decide between PocketSphinx, the core part of CMUSphinx, or Kaldi. Both matched all needed criteria. We decided us for Kaldi.

2.4.1 Kaldi

Kaldi is in development since 2009 and started at the Johns Hopkins University. It uses the latest Apache licence [17] and is written in C and C++. It is possible to train an acoustic model with your own data with a training script of the community.

The training is based on HMM and in the process of creating the acoustic model a test run is implemented that gives out the information of Word-Error-Rate (WER), Insertions (INS), Deletions (DEL) and Substitutions (SUB). To have a straight value for the correctly recognized words without caring about the multiple insertions, we added 'Recognition Accuracy' (ACC). Kaldi provides a decoder that can analyze a Wave (WAV) file of any length but it doesn't provide the possibility to decode an audio stream. An audio stream will be necessary for an ongoing recognition of a match but we will ignore this problem in this thesis because it is not necessary for the basics we are working on here.
A deeper insight into Kaldi can be found in the documentation of Kaldi [7].

2.4.2 Values

To get a better overview what our later results contain and what we can learn from it, a short explanation follows.

DEL - Deletion: Whenever the decoding fails and can't detect the spoken word, we have a deletion.

SUB - Substitution: We get a substitution when we don't find the correct word in any of the recognized words.

INS - Insertion: When the decoder detects more words than there were spoken we have an insertion. Insertions are not limited and can affect the Word-Error-Rate a lot.

%ACC - Accuracy: Gives the percentage value of how many words got correctly recognized, regardless of the amount of insertions.

%WER - Word-Error-Rate: The Word-Error-Rate gives the ratio between the tested words and all summed deletions, insertions and substitutions. This value can get higher than 100% because we can have multiple insertions in one recognition test.

Mono - Is the abbreviation for the Monophone-based recognition in Kaldi

Tri1 - Is the abbreviation for the Triphone-based recognition in Kaldi

2.5 Phones

Phones are the essence of a distinct speech sound made [15]. So a specific sound will always have the same phone. Speech recognition uses phonetics to give a written word a spoken equivalent and is necessary for the acoustic model to match the recordings with the words.

The pronunciation of a word can be very different in different languages or dialects. That is the reason for the difficulties to develop an always fitting acoustic model. The better the phones for the phonetic spelling are chosen, the better is the recognition in

this distinct language. It is necessary for the model to know when a character is silent, or how it is pronounced at this position.

Kaldi uses monophones and triphones for the acoustic model.

2.5.1 Monophone and Triphone

Monophones are represented by adding all phones together to one word in phonetic spelling.

Triphones are represented by grouping the single phones as triplets. Every letter is combined with the letter before and after, so we get a chain of overlapping triplets 'L-X+R'. The first and the last letter naturally have just a duple because they miss one side.

Example [16]:
Mono: TRANSLATE [TRANSLATE] t r @ n s l e t
Tri1: TRANSLATE [TRANSLATE] t+r t-r+@ r-@+n @-n+s n-s+l s-l+e l-e+t e-t

2.6 Acoustic Model

The acoustic model is the dictionary between the raw impulses in an audio stream and the included words. It is necessary for the language recognition system to find the known words in the audio stream. Without it no recognition would be possible.

To train an acoustic model, a big amount of audio data is needed. As long as we don't want to train the acoustic model just for one person to have a decent speaker recognition it can be said the more words and the more speakers one has the better. For the training each audio file has to contain one to a few spoken words. Every spoken word has to be linked to a translation in files that contain the information which word is spoken and what phones are used in the word.
A deeper insight into the acoustic model can be found in the Master Thesis with the topic 'Automatic speech recognition using Kaldi' by Ondřej Plátek [9] and the documentation of Kaldi [8].

2.7 HMM - Hidden Markov Model

The Hidden Markov Model is based on the Markov model. The Markov model is a simple network of states. Every state represents a phoneme that is used at least once in any one of the trained words. The different states in the network are just connected with states that can follow them.
For the speech recognition this means that the trained system has a linked network of

states. Different combinations of these states represent different words. When a sound is recognized that fits to a known start of this network, the recognition starts. Now it will be checked if the following recognized sound is fitting to a linked state of this network. If it fits the next sound will be checked and so on. To finish the recognition of a word the network path has to match a known word pattern, get to an endpoint or discard the word because the current recognized sound won't fit to any of the paths the network could take.

Normally the network of Hidden Markov Model is weighted with a multivariate Gaussian distribution model [6]. The model just knows the current state and takes as next state the one with the highest probability.
A deeper insight into the HMM can be found in the Master Thesis with the topic 'Automatic speech recognition using Kaldi' by Ondřej Plátek [9] and the documentation of Kaldi [10].

3 Training Modifications and Expectations

To train an acoustic model with just one person to get some kind of voice recognition, the person needs to record a lot of repetitions of every word in order to have enough data for a decent training. To improve the quality of the trained model, we adjust the recorded audio files with different sound modifications. Following we will explain in detail what kind of modifications we used and how we expect them to improve the acoustic model.

3.1 Acceleration and Deceleration

To have a wider range of faster and slower speaking we modified automatically an acceleration and deceleration by 10% to each recording and added these files to the test or training.

Modification Details

Every clear record got accelerated and decelerated by 10%.

Expectations

We expect a slight improvement of solidness for the recognition because the speed modifier increases the amount of different training data.

3.2 RIR - Room Impulse Response

Especially in big halls, when the trainer shouts the commands the reverberation can be strong enough to disturb the recognition. To avoid repeated command recognition or wrong identifications, it is necessary to add the Room Impulse Response to the training. Since a clear echo sounds almost exactly like the speaker, the speech recognition system will probably recognize it as an order and act upon it.

The RIR is simulating echoes and reverberation for every spoken word and is practically a big set of modification data for the audio stream with which the stream can be converted

to an audio stream with reverberation effects. For this needed room impulse responses we use a tool named rir-generator [18].

To understand how it is created, one has to imagine a certain room. In this room exists a source and a receiver. To be able to calculate the expected reverberation effects, we remove all walls, the ceiling and the floor, so we have nothing but an endless space, the source, and the receiver. For our example we consider the process of calculating the reverberation just with the removed walls. We mirror the source on every wall we removed (see Figure 3.1). The distance is always the doubled distance the source has to the original wall. There are five noise sources now that will all make the same noise that the receiver will hear. Because of the greater distances, the new sources will be quieter, and the receiver will hear them later. Mirroring the new four noise sources in the same way will give us additional twelve noise sources instead of the expected sixteen sources because all of them will mirror once back to their original source. This could be increased infinitely but in addition to the diminishing effects, the complexity grows extremely fast.

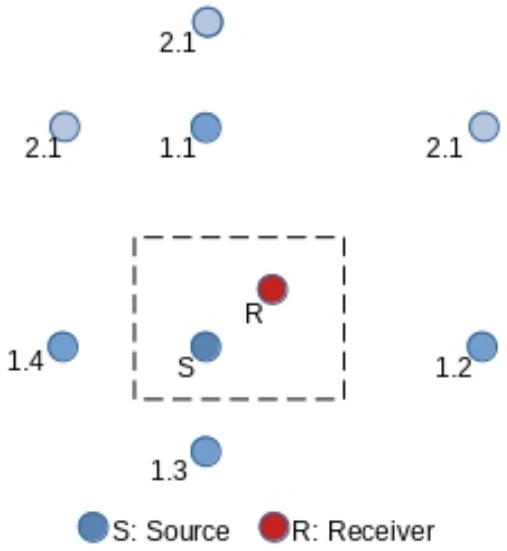

Figure 3.1: Room Impulse Response example with walls

After creating the RIR we save the information in a numpy (npy) file. It is not necessary to use numpy but it is a lot easier because the implemented functions are able to save and load the data structure without any changes. Those files have to be created just once and can be used for all future audio files.

Modification Details

For our training we decided to simulate an echo of three different sized halls or hangars.

The smallest one has the size of a sports hall typical for Germany. The second and third hall are not directly connected to any real hall except for a rough size ratio depending on different known halls (see Figure 3.2).

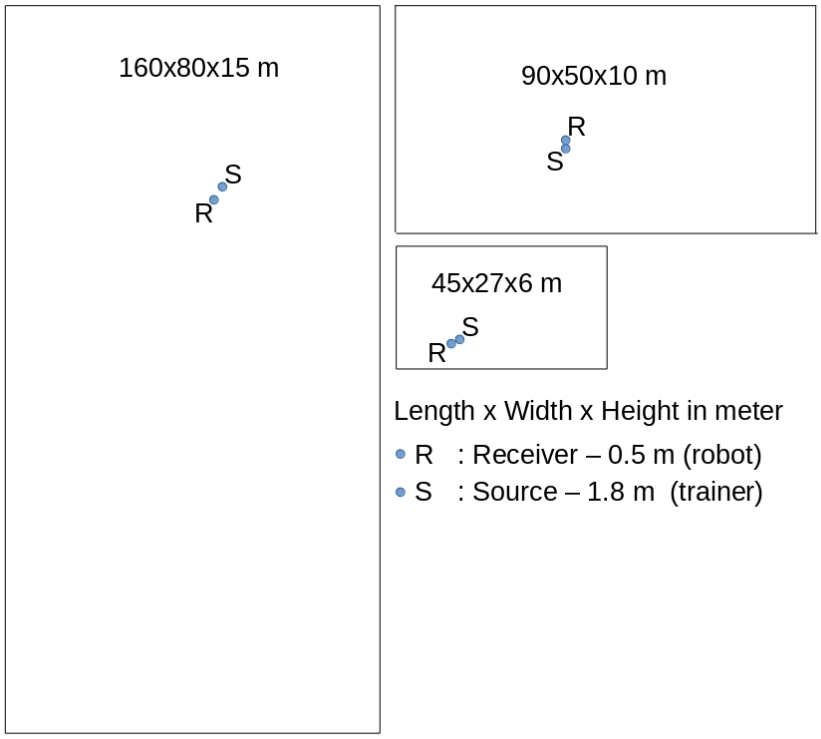

Figure 3.2: Halls used for the room impulse response

In every case we set the reverberation time value to 1.5 seconds. Depending on the hall size, we now have three different reverberation effects we can adjust our recordings to. When all our halls are present as RIR-data, we convolve them with our spoken word and create three audio files with different echos whenever we record a new command.

Expectations

We expect that the acoustic model will be more robust to handle reverberation effects to reduce the amount of insertions and substitutions (see Chapter 2.4.2).

3.3 SNR - Signal to Noise Ratio

With the Signal to Noise Ratio we want to simulate the commands spoken in a noisy environment like a spectating crowd or the sounds the body of a robot makes.
For the training and the test files we used four different sound snippets of 2 seconds each. Two for the tests and two for the training but this can be enlarged easily, if necessary.

The tests and the training have both one sound file with a the sounds of surrounding people, like laughter or loud speaking, and one with primarily the noise of the robots body. This is necessary because the movement produces mechanical sounds that will disturb the recognition in nearly every moment of the game, so it is very important to toughen the recognition against these noises.

Modification Details

In addition to simply adding the clear spoken word and the noise recordings, we modify the noise record with different ratios. This way we increase the amount of training data massively and can simulate more situations when the spectators become louder or less noisy.

For this attempt we have five additions for both noise records as follows:
clear spoken word + noise sound
clear spoken word + noise sound * ratio of -5db
clear spoken word + noise sound * ratio of 5db
clear spoken word + noise sound * ratio of 15db
clear spoken word + noise sound * ratio of 25db

The ratio is working in a negative way. The higher the ratio, the lower is the noise of the added record. This means, with 25db it is very hard to hear any noise anymore while the ratio of -5db is increasing it to such a level that a spoken word is barely understandable.

Expectations

We expect a highly improved reliability to noises by these modifications.

4 Evaluation of the results

4.1 Results

Kaldi provides a result table of every test run that is made after the training. We will analyze these results in detail to find out how the different training data modify the recognition.
Therefore we additionally selected every single recognition result of every run and transferred it into different excel sheets. Now we can sort and select according to our needs.

4.1.1 Data

The test data and the training data are randomly selected before the system starts working with them. All summed up, we produced nearly 11.000 audio files. The test data are 20 to 21% of all that data. So we have roughly 80% training data. It is elemental, that we never mix the training and the test data.

Since every training session always used all the test data, we can then evaluate each training session with adjusted test data. The results can be limited by filters to the desired tests in excel. In the following, we explain which training sets contain which modifications.
In simple terms, each modification is simply added to the previous training data.

Minimal setup: In this setup we just trained the clearly spoken words without any modifiers.

AD setup: The clearly spoken words now got accelerated and decelerated and added to the training. So we have three times the minimal setup with different speeds.

RIR setup: Now we add the RIR modified data to the training, so we have six times of every basic word. Now the half of our data have significant reverberation effects.

SNR setup: For this setup we add data to the training where we artificially added acoustic noise of people and robot mechanics at different SNRs ranging from -5 dB to 25 dB.

ALL setup: As the name suggests, we now use all our previous training data together.

4.2 Minimal Training

For the start we will analyze what what the system is able to recognize after the minimal training in the different test conditions and take a closer look at the impact of the different Signal-to-Noise-Ratios.

4.2.1 All Tests on the Minimal Setup

Table 4.1: Monophone - With the Minimal training the following results are given:

Training 1: Minimal	Words	DEL	INS	SUB	%ACC	%WER
Test 1: Minimal	111	0	0	1	99,10	0,90
Test 2: 1 + AD	333	0	0	3	99,10	0,90
Test 3: 2 + RIR	666	0	15	280	57,96	44,29
Test 4: 2 + SNR	1887	0	1177	465	75,36	87,02
Test 5: ALL	2220	0	1192	742	66,58	87,12

Table 4.2: Triphone - With the Minimal training the following results are given:

Training 1: Minimal	Words	DEL	INS	SUB	%ACC	%WER
Test 1: Minimal	111	0	0	1	99,10	0,90
Test 2: 1 + AD	333	0	1	3	99,10	1,20
Test 3: 2 + RIR	666	0	210	190	71,47	60,60
Test 4: 2 + SNR	1887	0	804	586	68,95	73,66
Test 5: ALL	2220	0	1013	773	65,18	80,45

As we can see, the minimal training shows no further problems with acceleration and deceleration but struggles with noises and reverberation effects.
A very remarkable point is visualized with this data (see Figure 4.1). The triphone phonetic has an accuracy of around 70% with the RIR and the SNR data. The difference is very small and just shows us that the reverberation effect and the noises produce distinct but similar difficulties for the recognition. The monophone phonetic on the other hand has a low accuracy with the RIR Test and a moderate accuracy with the SNR test. This suggest for minimal training, the monophone-based recognizer works better than the triphone-based recognizer in noise, but has bigger problems with reverberation effects.

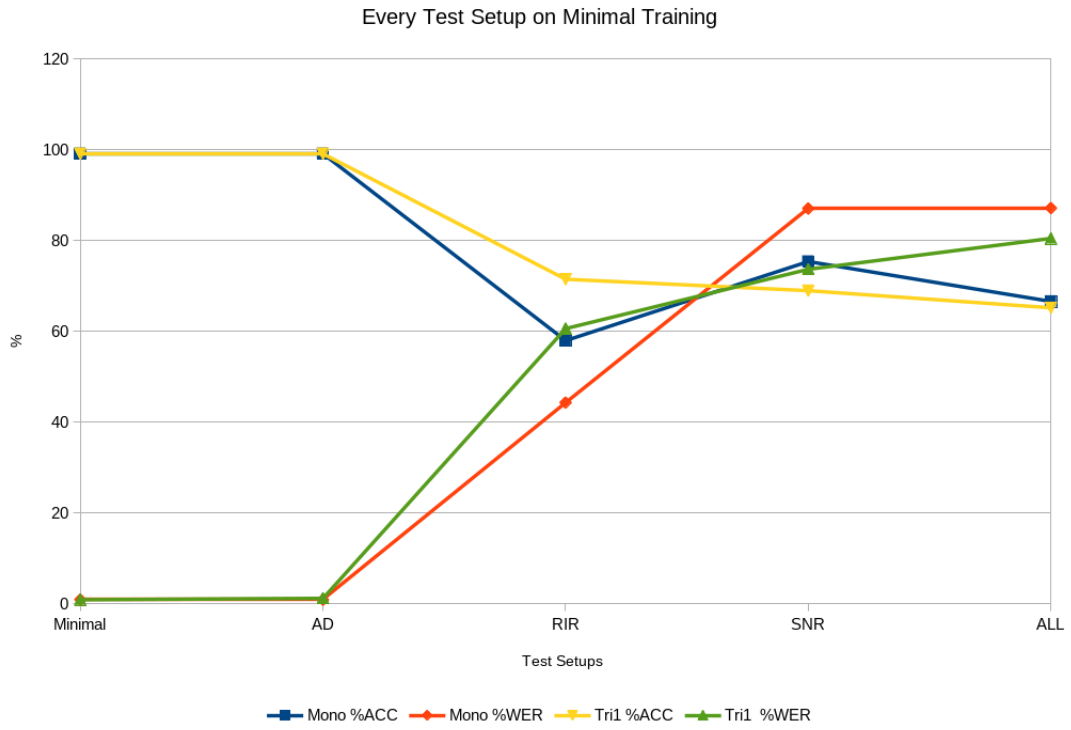

Figure 4.1: Compare of Mono and Tri1 with minimal training over all test conditions. (See Tables 4.1 and 4.2)

4.2.2 Detailed view on the SNR recognition

To get a deeper understanding of how the monophone- and triphone-based recognizer are working, we take a detailed look at the single SNR values and compare their results.

Table 4.3: Tri1 - Minimal Setup with detailed SNR

Words: 222	DEL	INS	SUB	%ACC	%WER
SNR -5	0	105	186	16,22	131,08
SNR 0	0	128	128	42,34	115,32
SNR 5	0	150	109	50,90	116,67
SNR 15	0	147	22	90,09	76,13
SNR 25	0	93	3	98,65	43,24

Table 4.4: Mono - Minimal Setup with detailed SNR

Words: 222	DEL	INS	SUB	%ACC	%WER
SNR -5	0	44	156	29,73	90,09
SNR 0	0	188	84	62,16	122,52
SNR 5	0	203	76	65,77	125,68
SNR 15	0	212	25	88,74	106,76
SNR 25	0	119	3	98,65	54,95

Here we can see the difference in the functionality of monophone and triphone (see Figure 4.2). While the triphone recognition system has a very bad %WER and %ACC with the worst SNR, the monophone recognition system starts much better. That's because the monophone is not detectable in that strong noise, so we mostly have substitutions for a wrong recognition but in the noise beside it rarely finds anything.
With the next steps, when the noises get reduced, the decoder more and more often finds words where shouldn't be any. So while the triphone decoder finds less and less insertions with lowered noises, the monophone decoder finds more. Only in the last SNR the noises become so low that the recognition is improving.
The reason why the accuracy of triphone is lacking at low SNRs is the same why the Word Error Rate starts this bad. The functionality of triphone recognition system to split the word in smaller parts for a more flexible recognition shows more problems with strong noises.

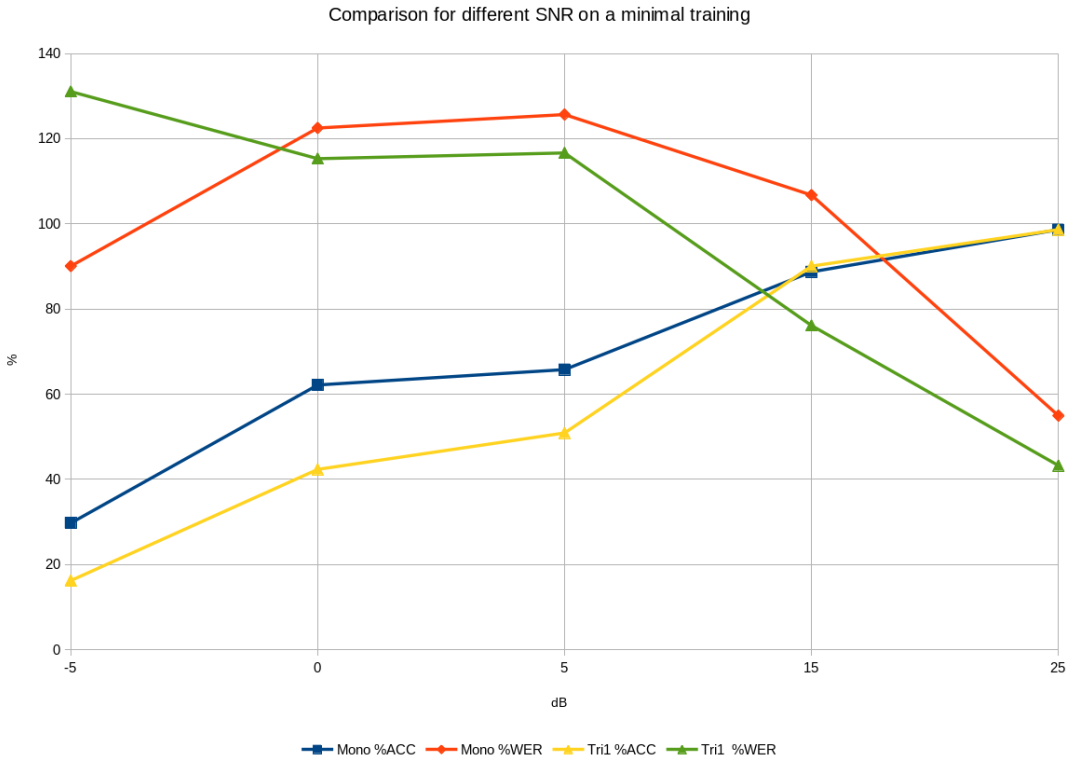

Figure 4.2: Compare Mono and Tri1 in the Minmal Setup. (See Tables 4.4 and 4.3)

4.3 Considering Noise and Reverberation in the training

Until now we just took a deeper look in the results of the minimal training. More interesting are the RIR and SNR trainings we evaluate in this chapter. First we look at the results of each training for itself. Later we compare the results to get a better understanding for the differences.

Table 4.5: Average values of the monophone recognition system on the noisy test files

Words: 777	DEL	INS	SUB	%ACC	%WER
RIR-Training	256	128,5	46	61,13	55,41
SNR-Training	0	347	163	79,02	65,64
ALL-Training	0	392,5	117	84,95	65,57
Better	SNR	RIR	RIR	SNR	RIR
Improvements in ALL	NO	NO	NO	YES	NO

The training sessions obviously have different strengths and a combination should im-

prove the results. For whatever reason, that is not the case. Just the accuracy is increasing slightly, and the SNR training can neutralize the deletions of the RIR training.

Table 4.6: Average values of the monophone recognition system on reverberant test files

Words: 111	DEL	INS	SUB	%ACC	%WER
RIR-Training	0	162	7,67	93,09	152,85
SNR-Training	0	34,67	10,33	90,69	40,54
ALL-Training	0	2	0	100	1,8
Better	-	SNR	RIR	RIR	SNR
Improvements in ALL	-	YES	YES	YES	YES

The insertions of the RIR training with the RIR test files are remarkable, especially because the SNR training is performing much better. While Accuracy and Substitutions are nearly equal there are big differences in Insertions and WER between them. A combination improves the results and comes near to perfection.

Table 4.7: Average values of the triphone recognition system on SNR test files

Words: 777	DEL	INS	SUB	%ACC	%WER
RIR-Training	20,5	464,5	265	63,26	96,53
SNR-Training	0	133,5	230,5	70,34	46,85
ALL-Training	1	59,5	113,5	85,27	22,39
Better	SNR	SNR	SNR	SNR	SNR
Improvements in ALL	NO	YES	YES	YES	YES

As expected, the results of the SNR test files on the RIR training are not good but remarkable is, that the results on the SNR training are quite bad too. There we expected a much higher accuracy. In the combined training we can see a huge improvement in nearly all aspects.

Table 4.8: Average values of the triphone recognition system with RIR test files

Words: 111	DEL	INS	SUB	%ACC	%WER
RIR-Training	0	128,67	4,67	95,80	120,12
SNR-Training	0	56	22,67	79,58	70,87
ALL-Training	0	0	0	100	0
Better	-	SNR	RIR	RIR	SNR
Improvements in ALL	-	YES	YES	YES	YES

As before with the monophone recognition system the insertions with the RIR training

are exceptionally high but again the combination of RIR and SNR training presents us a perfect result.

If we look at these data we cannot find any other explanation than the RIR training lacks in strengthening the recognition against insertions. It even seems ist can worsen the quality. On the other hand, the substitutions are obviously better than in the SNR training sessions, and the results of the full training are much better than the results of the interim training sessions suggest. The two training sessions seem to complement each other and look like a very good combination.

Another remarkable point is what the graphs (see Figures 4.3, 4.4 and 4.5) and tables (see Table 4.5) are showing. It is something we could recognize in other results too. The monophone recognition system doesn't seem to be good for the recognition in surroundings with loud noises. While the triphone recognition system is improving in every aspect with the full training, the monophone system is only improving in accuracy and even worsening in insertions.

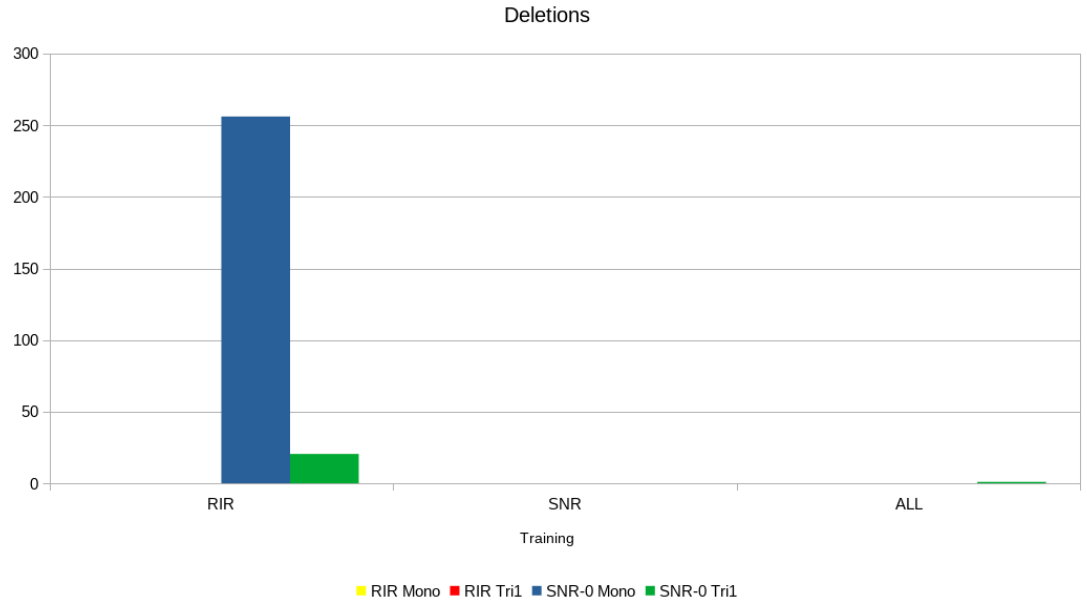

Figure 4.3: Compare deletions in different RIR, SNR and ALL Training. (See Tables 4.5, 4.6, 4.7 and 4.8)

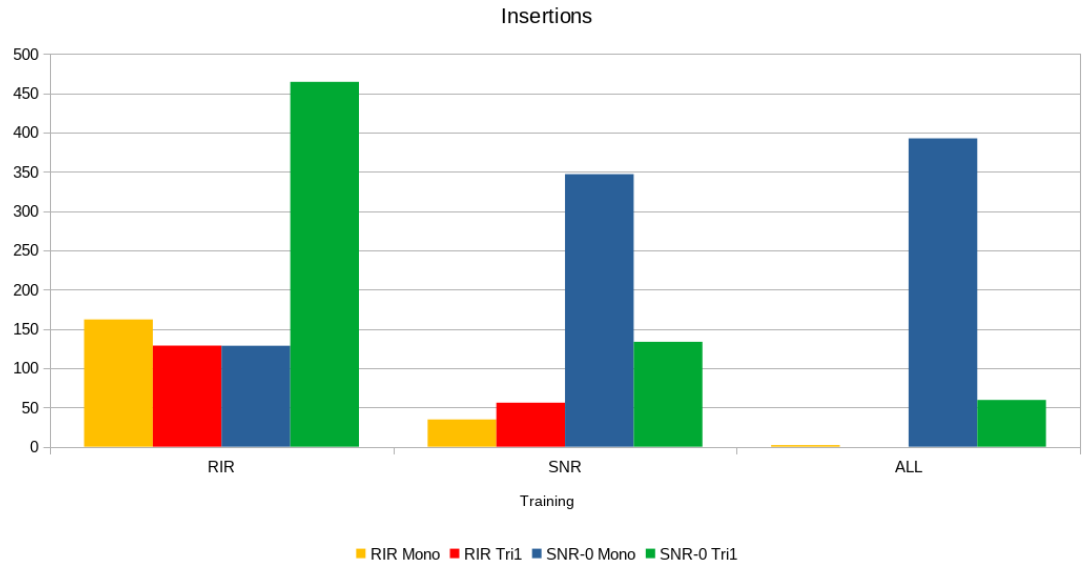

Figure 4.4: Compare insertions in different RIR, SNR and ALL Training. (See Tables 4.5, 4.6, 4.7 and 4.8)

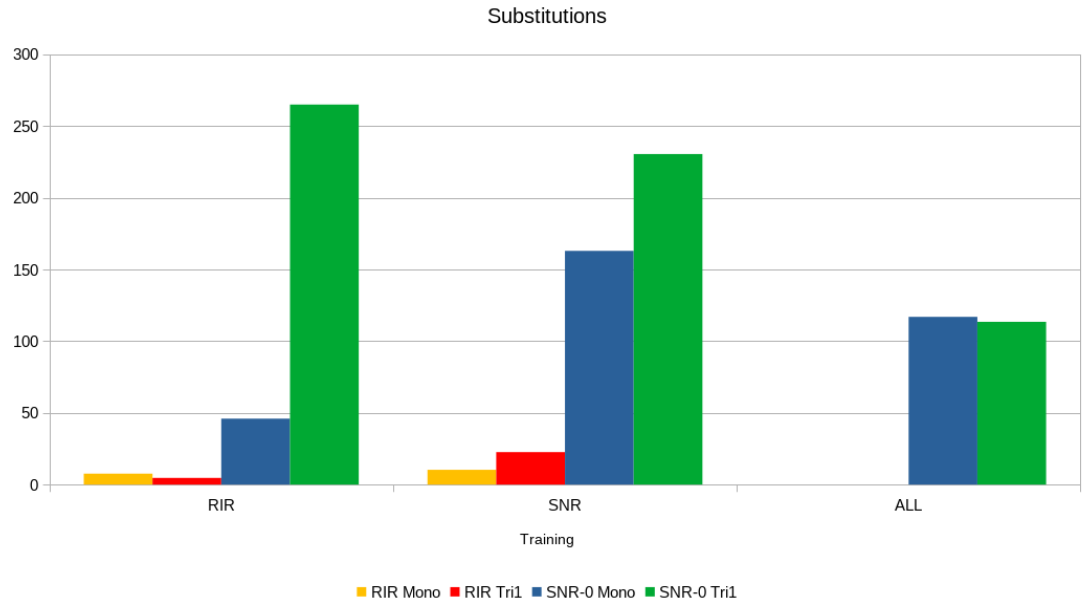

Figure 4.5: Compare substitutions in different RIR, SNR and ALL Training. (See Tables 4.5, 4.6, 4.7 and 4.8)

4.4 Outcome of All Training

Apart from the details of partial training sessions, we now want to get to know how the full training is performing with the more difficult tasks and in general terms.

4.4.1 SNR Comparison on a full training

To be able to estimate the impact of the noise at different SNRs and to get detailed information about the recognition differences of the monophone and triphone recognition system, we compare the test results at different SNRs separately SNR-Level on a training with all data included.

Table 4.9: Mono - SNR Comparison on a full training

Words: 222	DEL	INS	SUB	%ACC	%WER
SNR -5	0	75	161	27,48	106,31
SNR 0	0	257	32	85,59	130,18
SNR 5	0	227	24	89,19	113,06
SNR 15	0	126	0	100	56,76
SNR 25	0	25	1	99,55	11,71

Table 4.10: Tri1 - SNR Comparison on a full training

Words: 222	DEL	INS	SUB	%ACC	%WER
SNR -5	0	20	155	30,18	78,83
SNR 0	1	22	24	88,74	21,17
SNR 5	1	17	27	87,39	20,27
SNR 15	0	10	1	99,55	4,95
SNR 25	0	5	0	100	2,25

On the strongest level of Signal-to-Noise-Ratio the triphone phonetic has many substitutions but just very few insertions. It was predictable that on this level the recognition would fail but it is impressive that still 30% of the words are identified correctly (see Figure 4.6).
On the other side the monophone system has more insertions but can hold nearly the same accuracy. With the neutral or medium SNR-levels, the difference is much more remarkable. The monophone recognition has more insertions than tests cases were executed. It is obviously not able to handle the background sounds that were added. On the lowest SNR-Levels both recognition systems can identify nearly every word correctly but the monophone still struggles with insertions.

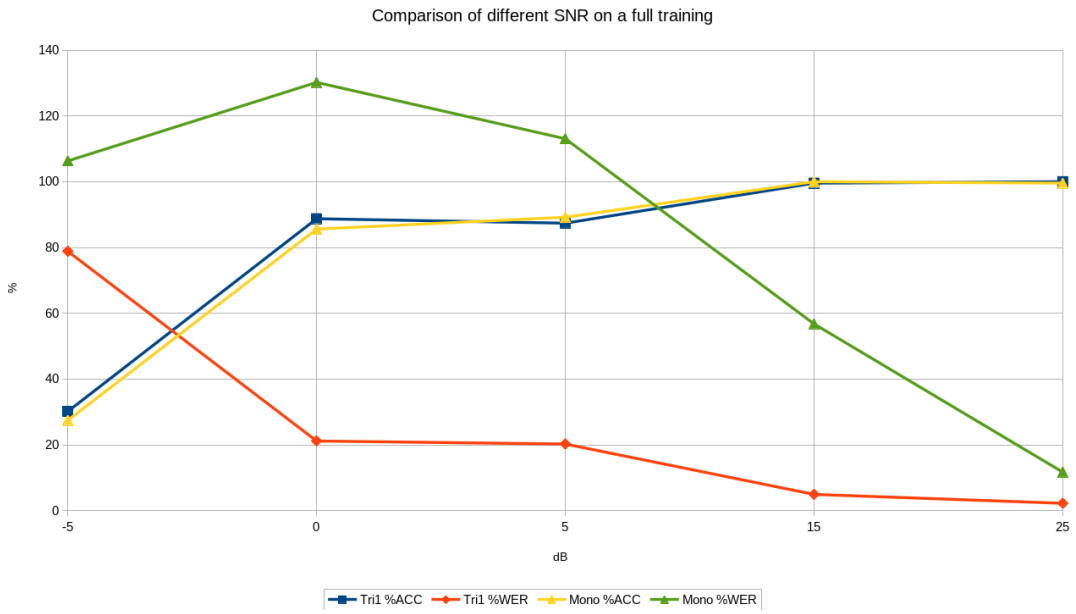

Figure 4.6: Comparison of the different SNRs ranging from -5 dB to 25 dB on a full training (See Tables 4.9, 4.10)

4.4.2 Final Results

Finally we look at the results of a full training with all tests included. The following tables show a very satisfying picture.

Table 4.11: With using all trainings together the following results are given:

Mono Training 5: ALL	Words	DEL	INS	SUB	%ACC	%WER
Test 1: Minimal	111	0	2	1	99,10	2,70
Test 2: 1 + AD	333	0	6	3	99,10	2,70
Test 3: 2 + RIR	666	0	12	3	99,55	2,25
Test 4: 2 + SNR	1887	2	791	237	87,44	54,48
Test 5: ALL	2220	2	797	237	89,32	46,58

Table 4.12: With using all trainings together the following results are given:

Tri1 Training 5: ALL	Words	DEL	INS	SUB	%ACC	%WER
Test 1: Minimal	111	0	0	0	100	0
Test 2: 1 + AD	333	0	0	0	100	0
Test 3: 2 + RIR	666	0	0	0	100	0
Test 4: 2 + SNR	1887	2	121	225	87,97	18,44
Test 5: ALL	2220	2	121	225	89,77	15,68

As we can see, monophone and triphone are acting quite similar. We can monitor excellent or even perfect results for the three first test sessions (see Figure 4.7) and can see the first real problems in the recognition of the last and most difficult test section, the noises.
But even here we have excellent results, if we take into account that one of the SNR values is -5 and produces so heavy noises that even a human would have problems to understand the records. If we take the results of section 4.4.1, we can see (see Table 4.10) that we achieve much better results with ignoring the worst files.
This won't count for the monophone recognition but anyway here we can strongly increase the accuracy by ignoring them. Only the insertions stay a problem, as we discussed earlier.

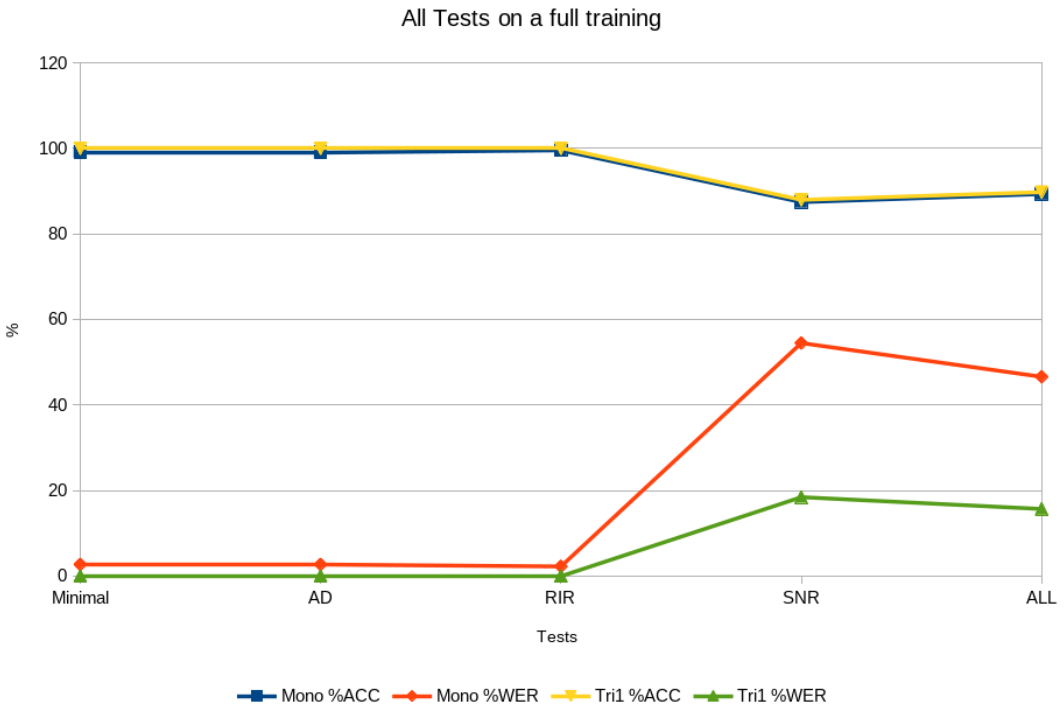

Figure 4.7: Compare Mono and Tri1 with a full training (See Tables 4.5 and 4.6)

5 Python toolkit

From the start it was our goal to produce a usable tool for the teams in the RoboCup to make the first steps to speech recognition for the robots. The purpose of the tool was always to automatically do all the configurations and modifications that Kaldi needs to work and to produce an acoustic model to give every member of the team the ability to train his own acoustic model.
It became obvious after starting the research, that we would need this intended tool for our research as well. The configuration of Kaldi, labeling every audio file, make the modifications and add every single path to all needed files in Kaldi means to create and modify hundreds to thousands of audio files and configuration files. The tool is a simple python program that can be navigated with the console and should work on every system, that can run Kaldi and python.

5.1 Overview

The tool is able to support the user with different tasks. The most important part is the recording and the modifying of the training and test data. It provides the ability to select a distinct speaker and a command to be trained. With a selected command word the tool is able to start and stop a recording. If the user is satisfied with the recording, the tool will start creating modified duplicates of the record automatically. The file will be accelerated, decelerated, overlaid with noises and reverberation effects. Afterwards the resulting 11 audio files will be added to the Kaldi configuration files for the individual speaker and moved to the speakers folder in the Kaldi data. In addition the tool will make a randomized choice if the recording is used as a test file or a training file.
Afterwards the user can proceed with the training of the same word or go back and choose another one.

5.2 Features

The tasks the user can perform with the toolkit are limited, but very helpful. In this chapter every tasks is explained in more detail. Especially the whole audio file creation and the following steps are very important to work, because if there is a failure in the configuration files, it can be very difficult to find the failure, even with the tools Kaldi

provides for searching through them. Also if it is found it could be, that the effort to fix it, is greater than to repeat the recording.

5.2.1 Speaker Addition

As long as we don't have a speaker recognition we want to train just one speaker at a time but when the requirements are given more speaker would produce better training results because it would strengthen the recognition for different voices, genders and pronunciations. Creating more speakers gives the possibilities to switch the trainer if needed at the moment and in the future to easily train a new team member.

Therefore we implemented a simple option to add a new speaker. The script will asks for the gender and add all necessary folders and configuration entries in the Kaldi structure.

5.2.2 Audio File Creation

For the recording we used pyaudio and wave. After the user started the recording he has two seconds to speak. After that time, the recording is stopped and the audio file is saved in a folder temporarily, until the user decides, if he keeps or drops it.
The file name is created with the information about the chosen command word, the active speakers name and the number of unique audio files in the folder of the speaker. If the user decides to keep the file, the modification process is started.

5.2.3 Audio File Modification

AD - For the simple acceleration and deceleration we use the package ffmpy, because the implementation of wave would change the duration of the audio stream, what would change the frequency. That is a problem, because Kaldi requires all audio files to have the same frequency.

RIR - For the reverberation effect we load the previously created room impulse response information with numpy.load, and the audio stream with pysf. Then we merge them with numpy.convolve and save the new modified audio stream with pysf.

SNR - After loading both audio streams with pysf, we modify the stream of the noises with a formula and functions of numpy. The modified noise stream is added to the audio stream of the spoken word and saved with pysf.

5.2.4 Configuration Files & Modifications

Our tool provides all necessary configuration files to create own acoustic models with Kaldi and will edit all information about new data to the needed configuration files, after the audio file creation.

Provided scripts

- **local/score.sh** - Copied script from 'kaldi-trunk/egs/voxforge/s5/local' that is needed to run the tool.
- **cmd.sh** - Provided script of the Kaldi Tutorial [19] to set local system jobs.
- **path.sh** - Provided script of the Kaldi Tutorial [19] to set paths to the needed files and directories.
- **run.sh** - Provided script of the Kaldi Tutorial [19] to run the training and testing with console output.

Provided configuration files

- **conf/decode.config** - Rarely modified configuration file for the different beam values.
- **conf/mfcc.conf** - Rarely modified configuration file for different options, like the sample frequency.
- **data/local/corpus.txt** - Lists all known words of the tool.
- **data/local/dict/lexicon.txt** - Holds a phone transcription for every word in the corpus.
- **data/local/dict/nonsilence_phones.txt** - Lists all phones that are audible.
- **data/local/dict/optional_silence.txt** - Lists all optional phones that are not audible.
- **data/local/dict/silence_phones.txt** - Lists all phones that are not audible.

To add a new word to the tool it is necessary to add it to corpus.txt and lexicon.txt. The better the phone transcription in the lexicon.txt, the better recognition results can be achieved. If the new word includes phones that are not already known in the nonsilences or silence phone files it is necessary to add them.

6 Conclusion

6.1 Conclusion

In this thesis the Kaldi framework was chosen to implement a speech command recognition for robots. For this implementation python comes to use because its good compatibility on different operation systems and simple linking to the Kaldi framework. The goal of this tool is to get closer to a speech command recognition for humanoid robots used in the RoboCup.
With this speech command recognition the trainer has the ability to influence the game and the behaviour of the robots from the side lane. Also is this speech recognition the basic for future communication between the robots itself.

The tool provides a simple console interface to create and train different speaker for Kaldi for the creation of a acoustic model. Also it provides different modifications of the training data to improve the robustness of the produced acoustic model. This are the first steps to get an acoustic model that is able to understand the commands even in a very noisy surrounding like the crowd in a football match.

This tool was tested by one speaker with in total nine different words. Every word was recorded around fifty times. Including the modifications the amount of words is around 9.500. 80% were used for training and 20% were used for testing the training.
The results of this training get analyzed step wise to get to know what kind of training has what kind of impact to the recognition.
All modifications lead to greater robustness of the acoustic model. An interesting point is that the appropriate training for a modification is not enough to get a good speech recognition like we can see in the results of the RIR training, where the recognition results for the RIR tests is not good. Other training data like that from SNR are necessary to get a very good speech recognition. At least for the now trained words the speech recognition is good enough to perform tests in a real environment.

6.2 Future Work

For the future we have a lot different options to improve the tool or work on the greater topic.

6.2.1 Robustness

The next step for better robustness of the speech recognition could be to add more modifications to the training set. This way more environments could be reproduced.

For the same goal it is possible to use the two microphones provided by the humanoid build of the robots to improve the overall audio quality of the recordings. For example the usage of a beam former that can increase the sound level of the commands of the trainer.

To improve the goal that only the crew of the team is able to command the robots, the addition of a distinct speaker recognition would be very helpful. In addition if the results of this speaker recognition are good enough, that would establish the possibility to use more than one speaker to train the speech recognition.

6.2.2 Practical Use

The first thing to do for practical use would be the implementation of 'real time' or 'online' recognition that is not provided by the Kaldi framework.

Beside different additional commands the robots will have to be able to recognize the whistles of the referee, so this would be an important future work.

Bibliography

[1] RoboCup - Official site, http://www.robocup.org/

[2] Maike Paetzel (2013). Spielerkoordination in RoboCup-Fußballspielen mittels gesprochener Sprache. Bachelor Thesis. Hamburg, Germany. https://robocup.informatik.uni-hamburg.de/wp-content/uploads/2015/02/BachelorThesis_MaikePaetzel.pdf

[3] Robert Keßler (2012). Geräuschquellenlokalisierung mit einem humanoidem Roboter. Bachelor Thesis. Hamburg, Germany. ,https://robocup.informatik.uni-hamburg.de/wp-content/uploads/2012/02/BA-Geraeuschqullenlokalisation.pdf

[4] Michael Nielsen (2017). Neural Networks and Deep Learning, http://neuralnetworksanddeeplearning.com/

[5] Gales, M. and Young S. (2008). The Application of Hidden Markov Models in Speech Recognition, volume 2, page 200, https://mi.eng.cam.ac.uk/~mjfg/mjfg_NOW.pdf

[6] Reynolds, D.A.; Rose, R.C. (January 1995). "Robust text-independent speaker identification using Gaussian mixture speaker models". IEEE Transactions on Speech and Audio Processing.

[7] Kaldi - Official site, http://kaldi-asr.org/

[8] Kaldi - Acoustic Model, http://kaldi-asr.org/doc/model.html

[9] Ondřej Plátek (2014). Automatic speech recognition using Kaldi. Master Thesis. Prague, Hungary, https://github.com/oplatek/kaldi-thesis/blob/master/text/main.pdf

[10] Kaldi - Hidden Markov Model, http://kaldi-asr.org/doc/hmm.html

[11] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. (2009). ROS: an open-source Robot Operating System. In ICRA workshop on open source software, volume 3, page 5. Kobe, Japan

[12] Wiki of ROS, wiki.ros.org

[13] HD Pro Webcam C910 - Logitech, https://support.logitech.com/de_ch/product/hd-pro-webcam-c910/specs

[14] Source of the robot picture, https://robocup.informatik.uni-hamburg.de/photos/?occur=1&lang=de&slide&album=115&photo=1677

[15] Phone explanation, http://www.voxforge.org/home/docs/faq/faq/what-is-the-difference-between-a-phone-and-a-phoneme

[16] Monophone and Triphone example, http://www.voxforge.org/home/docs/faq/faq/what-is-the-different-between-a-monophone-and-a-triphone

[17] APACHE LICENSE, VERSION 2.0, https://www.apache.org/licenses/

[18] RIR-Generator, https://github.com/Marvin182/rir-generator/blob/master/python/pyrirgen.pyx

[19] Kaldi tutorial for Dummies, http://kaldi-asr.org/doc/kaldi_for_dummies.html

Eidesstattliche Erklärung

Ich versichere, dass ich die vorstehende Arbeit selbstständig und ohne fremde Hilfe angefertigt und mich keiner anderen als der im beigefügten Verzeichnis angegebenen Hilfsmittel bedient habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht.

Ich bin mit einer Einstellung in den Bestand der Bibliothek des Fachbereiches einverstanden.

Hamburg, den _____ Unterschrift: _____