Meet the Meka
The Meka Robot

- The Meka is a humanoid robot designed for human interaction
- It has two stereo, directional mics mounted on the chest
- It has speakers on the base
- It did *not* come with any software for speech recognition
Motivation

For humans working with robots, speech has many advantages:

- A natural model for human-robot interactions
- Enables interaction away from a keyboard
- Faster than using the keyboard for many typists
- Who doesn’t want to talk to a cool robot?

However there are some challenges:

- More difficult for the programmer
- Hard to achieve high accuracy in the general case
- Potentially time consuming to customize per application
PILOT Plan

Add speech recognition to the Meka with the following goals:

- High-accuracy speech recognition
- Versatile/Flexible implementation
- Useful, simple-to-use APIs and tooling
- Stretch goal: provide sound localization
Intro Video

Intro video
Background

PILOT: Speech Processing for the Meka Robot
“ROS is an open-source, meta-operating system for your robot.”

ROS is the software stack used on the Meka

- Nodes (processes)
- Defined message exchange formats
- Synchronous style services
- Pub/Sub model on hierarchical topic tree
- Extremely modular system
CMU’s Sphinx library was used for the heavy lifting. It requires three main components:

1. **Acoustic model**
   - Breaks a stream of sounds into basic units (phones)

2. **Phonetic Dictionary**
   - Maps from phones to words

3. **Language model**
   - Restricts word search

![CMU Sphinx pipeline](attachment:image.png)
Solution Overview

System overview
Accomplishments

- Versatile, high accuracy, speech recognition system
- Simple to use APIs that integrate with ROS
- Scripts for creating services
- Scripts for testing accuracy
Approach
The solution needs to be accurate to be useful

- Quantified accuracy of interacting with the Meka in various scenarios
- Investigated variables influencing accuracy
  - Adapting acoustic models
  - Varying dictionaries
  - Varying language models
  - Audio filtering
  - Distance from mics
  - Headset vs On-board mics
Versatility

The solution needs flexibility for future use. It supports:

- Startup selection of Sphinx parameters (models/dictionary)
- Varying Sphinx parameters at runtime
- Multiple instances of Sphinx simultaneously
- Publishing to multiple topics/services at a time
Usability

The solution should provide a set of easy-to-use APIs, scripts for automation, and integrate seamlessly ROS

- Automation for creation of new application speech services
- Automation for recording samples for testing
- Fit into existing ROS ecosystem
- Provide API for common workflows
  - Query-and-Response API for workflow: Prompt a yes/no question and wait for response
An *acoustic model* contains acoustic properties for each phoneme:

- Breaks a stream of sounds into basic units
- Trained from many hours of transcripted recordings
- Currently leveraging the default US English model provided by CMU Sphinx
A *phonetic dictionary* contains a mapping from phones to words

- Rosie the robot
  - R OW1 Z IY0 . DH AH0 . R OW1 B AA2 T .
- The default provided by CMU Sphinx contains the 16k most common English words
A *language model* is used to restrict word search

- Defines which word could follow previously recognized words
- Used to strip words that are not probable
- Can be a grammar, statistical n-gram model, etc
- Ability to predict next word critical for performance
- The default provided by CMU Sphinx is trained on Guttenberg texts
Model Creation

- It is a *lot* of work to create models from scratch
  - Acoustic models require hours of recordings with transcripts
  - Language models can be very application specific
- It was a goal to avoid time consuming model creation for each new application
Experiments
## Experiment Overview

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adapting acoustic model</td>
<td>Accuracy</td>
</tr>
<tr>
<td>Audio filter</td>
<td>Accuracy</td>
</tr>
<tr>
<td>Varying language model</td>
<td>Accuracy</td>
</tr>
<tr>
<td>Varying dictionary</td>
<td>Accuracy</td>
</tr>
<tr>
<td>Accuracy vs Distance</td>
<td>Usability</td>
</tr>
<tr>
<td>Headset vs On-board mics</td>
<td>Usability</td>
</tr>
</tbody>
</table>

Table 1: Experiment impact
Acoustic model adaptation allows customizing an existing model with training.

- Run tests to measure baseline
- Record samples with transcripts
- Process samples and update acoustic model
  - Code provided by CMU Sphinx
- Re-run tests to measure improvement

CMU Sphinx pipeline
Audio Filtering

When recording with the chest mics on the Meka, the motors create background noise. *Incorrectly* believed that it affected accuracy.

- Sometimes no results at all when motors active
- Audible in recording as annoying hum
- Created a simple audio pipeline to remove noise
  - Expand
  - Compress
  - Amplify
  - Limit
Audio Filtering

- Expand
  - This increases the dynamic range, with a cut-off threshold
- Compress
  - This decreases the dynamic range
- Amplify
  - Adjust the remaining signal
- Limiter
  - Helps prevent distortion/clipping
Varying Language Model and Dictionary

- When using n-gram language model, model and dictionary are linked
- One method for creating models and dictionaries is to create a paired dictionary/language model based on a text corpus
  1. Create a representative text corpus for the intended application
  2. Create a dictionary entry for each word
  3. Create an n-gram language model based on sentences in corpus
- Leveraged existing software provided by CMU Sphinx for this approach
  - Simpler than creating grammar for small sets
  - Less manual work, more automatable

CMU Sphinx pipeline
Accuracy vs Distance

Speaking more softly or from a greater distance will change the quality of the audio received

- Assumed a normal speaking volume
- Created recording samples from various distances
  - With human trying to speak at the same volume
  - With a recording played back at the same volume each time
- A distance of at least 4 feet is required to be out of arms reach of the Meka
- You might not be comfortable within arms reach all the time
Qualitative Results
Accuracy vs Distance

Accuracy %

Distance in inches

Recording
Real human
Physical Distance

- Quality is inversely proportional to distance
- High accuracy requires being with 12 inches of the Meka
Acoustic Model Adaptation Results

- Acoustic model was not improved using any of the recorded samples
- Possible reasons why it didn’t improve accuracy
  - Baseline acoustic model matches our environment well
  - Accuracy was already quite high
  - Training data are fairly small
- Ran sanity check by training with incorrect data
  - 20 bad sample sentences lowered accuracy by 25%

![CMU Sphinx pipeline diagram](image-url)
Audio Filtering Results

Removing background noise from motors

- Mixed results
  - Allowed recognition to occur, but didn’t improve accuracy
  - Sometimes removed speaker due to signal being too similar to noise
  - Brittle solution

- Real problem was actually an issue of detecting silence
  - Stream of sounds coming in.
  - When do you stop and process?
  - When the user stops speaking!

- Motor noise caused system to never detect silence
- Simpler solution: measure motor noise baseline and use that as silence threshold
Varying Language Model and Dictionary Results

- Created a custom dictionary and language model
  - Built dictionary for each word in the text corpus
  - Built n-gram language model based on corpus
- Reduced size of dictionary
- Reduced size word search space for language model
- Most of the improvement in accuracy came from custom dictionary and language model

![CMU Sphinx pipeline](audio -> acoustic -> dictionary -> language -> text)
Benefits of Using a Microphone

A solution to the distance issue was to use a headset microphone

- Using a wireless, headset microphone allowed consistent accuracy and mobility
- Tested using a Bluetooth headset
  - There are other high quality UHF/VHF microphones that could be used
- The accuracy was equivalent to, or better than, being within 6-12 inches
- Also eliminated the need to remove background noise from motors
- Makes interaction much better
Video of “Simon Says”

Video of the Meka playing “Simon Says.”
Quantitative Results
Quality Measurements “Simon Says”

Measured the accuracy of 10 samples from a 20 sentence text corpus

- Samples from 7 unique individuals
  - 3 native English speakers
  - 4 non-native English speakers
- Corpus for playing a game of “Simon Says”
- Samples recorded using Meka chest mics without motors running
  - From 6-12 inches
- Corpus contains 30 unique words
“Simon Says” Corpus

Simon says:

touch your head
wave your left hand
wave your right hand
turn around
rub your belly
flap your arms
thumbs up
move your right arm like this
move your left arm like this
move both arms like this

step forward
step backward
look left
look right
look up
look down
imitate me
step left
step right
raise your arms
"Simon" Accuracy

"Simon Says" test results

- Substitutions
- Deletions
- Insertions
- Correct

% of total

Default

Custom
Results for “Simon Says”

- Default dictionary/language model accuracy = 17.7%
- Custom dictionary/language model accuracy = 90.4%
  - Native English speakers accuracy = 97.4%
  - Non-native English speakers accuracy = 82.8%
Quality Measurements “Random”

Measured the accuracy of 3 samples for a 10 sentence text corpus of random words

- Sanity check for evaluating accuracy
- All 66 words are used only once
  - “Simon Says” corpus had only 30 unique words
- Allows good restriction of search space
  - Word ordering is unique
- Doesn’t follow normal English language patterns
  - No bias in corpus
- Recorded using chest mics without motors running
  - From 6-12 inches
“Random” Corpus

Random sentences built from dictionary:

stringer dandruff throes clash fought soirées disturb
stews avenue bipolar orators bushman inquest chantey enthused
thudding outflank grassed eastward grimaced bailouts
slim mocks mariachi artworks slogged floating cider quailed
neuritis headier pullback manages urban enema
foxy civic sunset eyelash shaykhs apprises
blossom glided started lofted tractor defense locates contrary
boogies confront garlands unsure finis
armadas bused measles iguana garrotte timidity janitors
info dowelled athlete rind defter
“Random” Accuracy

"Random" test results

- Substitutions
- Deletions
- Insertions
- Correct

% of total

Default
Custom
Results for “Random”

- Default dictionary/language model accuracy = 14.4%
- Custom dictionary/language model accuracy = 93.9%
Discussion
Useful System

Goal to be able to go from idea to service with minimal effort

1. Generate list of sentences for application (text corpus)
2. Quickly build and deploy a service for recognition
Steps required to add speech recognition to an application

1. Create corpus file
2. Invoke utility script to create a paired language model and dictionary
3. Copy template launch file and configure service name, models, dictionary, and corpus
4. Launch new service

Recognized text is published to /servicename/rawoutput and /servicename/corpusoutput topics
System Ease of Use

Topic /rawoutput vs /corpusoutput

- Applications can get all audio converted to text from rawoutput
- Applications requiring strict matches can use corpusoutput
  - Text only published when it exists in corpus
- Using both allows detecting things that are heard but not understood
Integration

ROS provides a concept of a parameter service

- Solution provides a method to register parameters for a node
  - Integrates using the standard ROS naming convention/methods
- Enables using common core service, with parameters controlled by specific application
  - Callbacks for acoustic model, dictionary, language models, and corpus
- Can provide parameters at launch or runtime
  - Launch must files contain parameters
  - The core listen service provides callbacks for providing acoustic model, dictionary, and language models
Create a paired language model and dictionary

- usage: "/utils/build_lang_pkg.sh <corpus>
- Creates tar file with paired dictionary and n-gram language model
Provided Software

Record a corpus sample for accuracy testing

- usage: `./utils/record_corpus_sample.sh <corpus> <outputdir>`
- Prompts user to say each sentence in corpus
- Creates a recording for each sentence in corpus
- Creates a map from recording to transcript
Test accuracy of recorded sample

- usage: `./utils/run_test.sh <recording-dir>`
- Output
  - Words: 99 Correct: 94 Errors: 5
  - Percent correct = 94.95% Error = 5.05% Accuracy = 94.95%
  - Insertions: 0 Deletions: 3 Substitutions: 2
Provided Software

API to prompt human subject and wait for response (Q/A)

```python
from meka_ros_listen.srv import *
...
QandA = rospy.ServiceProxy('yesnoapi', YesNoAPI)
anst = QandA("Is it cold out?")
```
Git Statistics

- **Age**
  - 116 days

- **Total Files**
  - 758

- **Total Lines of Code**
  - 1089

- **Total Commits**
  - 66
Future Work

- Sound localization
  - Explored existing libraries and ROS modules that could be leveraged for the Meka
  - Experimented with them to determine possible value/difficulty of use
  - After initial work, this was dropped due to time constraints in order to prioritize accuracy and ease-of-use for speech recognition
  - HARK
  - http://wiki.ros.org/hark
  - ManyEars
  - https://github.com/introlab/introlab-ros-pkg/tree/master/manyears_ros
Future Work

- Custom, per-user acoustic models
- ESL acoustic models
- Confidence scoring for rejecting an unlikely hypothesis
- Other methods of creating language model
- Customizable phonetic dictionaries
  - Not as much discussion on this topic
- More service APIs like query-and-response
Thanks for coming!
“Simon Says” Corpus

Simon says:

raise your arms
wave your left hand
wave your right hand
turn around
rub your belly
flap your arms
thumbs up
step forward
step backward
look left
look right
look up
look down
step left
step right