Voice Controlled Retrieval System

EECS 452: Senior Design Project

Matthew Edelman – Jeffrey Duperret – Ryan Mazur – Eric VanBuhler – Yan Qi

Introduction

This project is a proof of concept design for robotic aid system for the disabled and elderly. The concept is that the user will command the robot to retrieve an object via voice command, and the robot will autonomously navigate its current environment returning the desired object to the user. Our system includes:

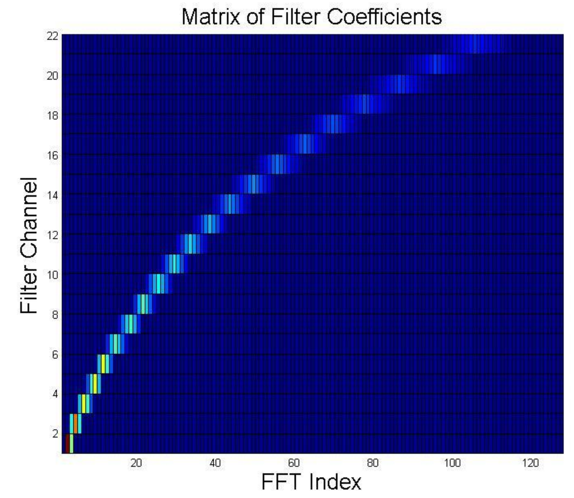
- Remote controlled tank as the robot
- Colored ping pong balls as the objects
- Noise cancelling microphone for voice input
- FPGA: color filter, visualization & robot control
- DSK: speech recognition & path planning

Speech Recognition

The Speech Recognition Engine has three main steps:

1. Record Sample

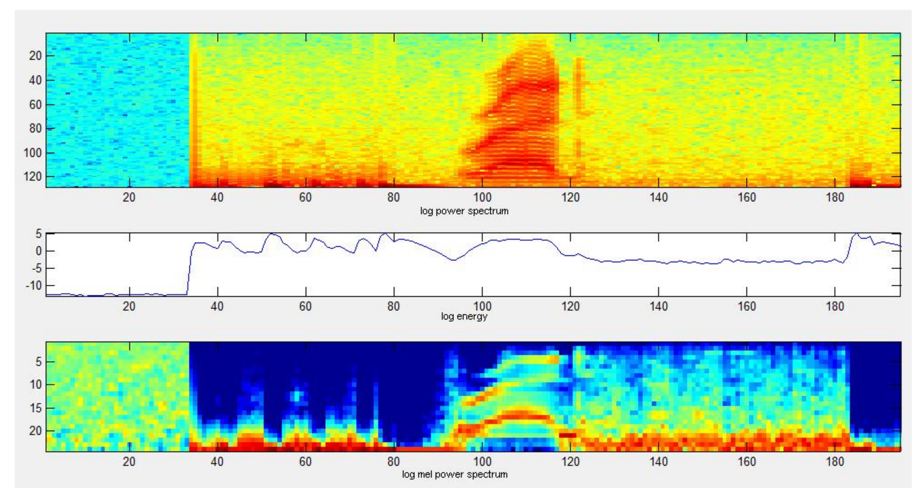
Following a short tone, a 2 second sample is recorded with an 8kHz sample rate. To reduce ambient noise, a noise cancelling headset is used. It uses two microphones: one pointed at the mouth and one pointed away. The ambient noise picked up from the microphone pointed away is removed from the voice signal.



2. Create Mel Coefficient Matrix

1. Take the FFT of 10ms sections of the recorded sample
2. Find log power spectrum and log energy for each 10ms section
3. Apply mel filter
4. Normalize with the energy and take log energy

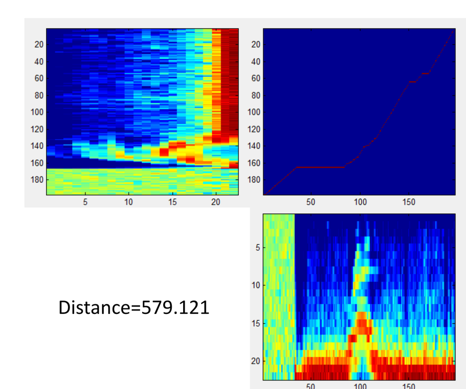
Visualizations of "red" spoken



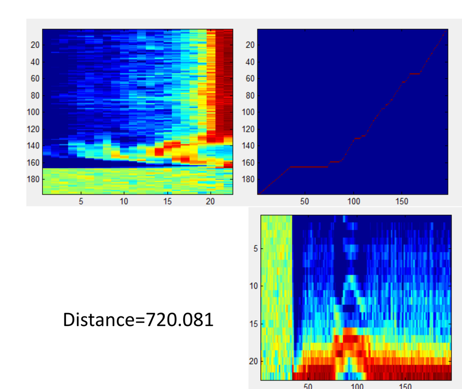
3. Compare Matrices

Compare the log mel power spectrum matrix to stored matrices by calculating distances. To calculate the distance, a dynamic programming technique is used to find the minimum distance in a group of points and use that to find the min distance in the next group of points. A path of the minimum distances is found and the value at the end of the path(upper right corner) is the minimum distance between the matrices.

"red" compared to "red"



"red" compared to "blue"

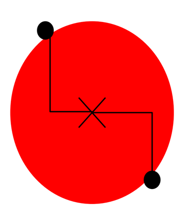


Color Filter

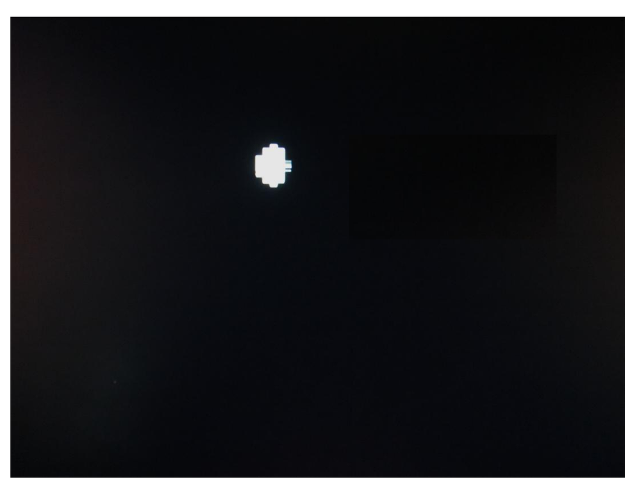
The color filter implemented on the FPGA analyzes the image captured from the camera to get the central position of objects in different colors. The position information is constantly updated and is used for Robot Control and Visualization.

Process:

1. Scans through the whole frame by row
2. Stores the calibration value of chosen color
3. Compares the calibration value with the scanned pixel quad
4. Averages the first and last pixel positions to locate the center
5. Returns the location information, encodes the data and sends it to DSK



Color filter output



Our algorithm continuously scans the pixels in the image one at a time. If a pixel's 8 bit RGB representation matches the pixel template for a specific object, then a white pixel is outputted to the VGA display. If not, a black pixel is outputted. This gives the user the same information that our object detection algorithm is given; a binary representation of if the camera sees an object or not.

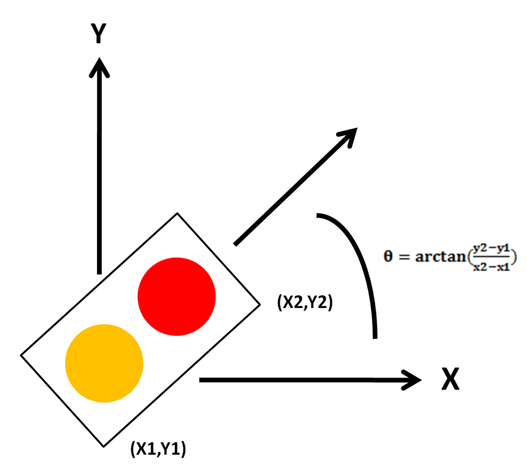
Robot Control

Overview: A small, radio-controlled toy is used as the robot. It has simple digital controls and is able to turn left or right in place, and can move forwards or backwards.

Control: The remote control has been modified to be connected directly to the FPGA. Commands are sent from the DSK to the FPGA.

Coordinates: Two sets of coordinates are used for the robot; a coordinate for the front and rear. From this we calculate the orientation of the robot in 2-D space using these coordinates. These values are calculated on the FPGA and sent to the DSK in real time.

Path Planning: The key component behind path planning and robot control is the overhead camera. The final output of the image processing algorithms is a set of X and Y coordinates. These are transferred from the FPGA to the DSK as 16-bit words. These coordinates are then used by the software running on the DSK to control the robot. The control algorithm implements a simple state machine.



Visualization

Significance: Visualization is a critical aspect of our project, as it gives a method of understanding what the algorithm is doing. This is both necessary in debugging, and it gives the user a sense of how our system works.

Output object tracking: To see how well our method track objects, we place a white square over them as they move. The FPGA is given the location of the objects from the color filter algorithm, and if the current pixel being read in is located on a specified square surrounding the object then a white pixel is outputted, if not then the original pixel is outputted.

Assist with calibration: The camera distinguishes objects according to object color. We assign these colors by placing the objects in known positions on the scene that the camera receives, extracting the pixels at the known location, and assigning those pixel colors to each object. To accomplish this, the user must know where to place the objects when calibrating their color values. Our algorithm outputs white pixels in those locations over the image to the user to facilitate this process.

As shown in the figure to the right, our algorithm outputs object detection and tracking. The algorithm is also used in specifying object color to detect objects.