Voice for ERTS

Shenshen Han
P545 School of Informatics and Computing
Fall 2011

1) Abstract

In this project, we will implement a simple but reliable voice navigation control scheme for ERTS.

2) Introduction

Our ultimate goal is to provide “voice control and navigation” facility for disabled people even the people with eye sight problems, so that they could be able to “drive” car to their desired destinations by themselves. Therefore, the reliability, interactivity, and user friendly are the main factors we want to design and improve.

3) Literatures

As it is quit difficult to find a native and powerful speech recognition engine, shortcomings of a chosen speech recognition such as Word Error Rate (WER) need to be considered and bypassed.

There are basically two general types of speech recognition engine. One is the native engine, such as the speech recognition functions provided by the Mac OS and Windows OS systems. Another type of speech recognition is actually carried out through the “cloud”, such as the “google voice search” and speech recognition on Android. Due to the field testing limitation, we are concentrating on native speech recognition engine.

As the ERTS itself is a quite autonomous model, so based on this, voice control is only good for administrative level of control, which means the vehicle should be able to drive by itself, and we just tell it where to go.

4) Objectives

4.1) Research and utilize a chosen voice recognition engine

We are dopting Sphinx-4 [1], an open source speech recognition engine mainly maintained by CMU.

4.2) Design and implement voice control primitives

Besides the well-acceptably recognized “words”, there are some common words that cannot be well distinguished by the speech recognizer due to the similar vowels and consonants. Such as “Left” and “Right.”

4.3) Design and implement voice navigation primitives

Some critical voice control primitives can also be used conjunctively with the voice path planning. Such as “Abort”, “Stop”, “Faster”.

4.4) Design and implement interactive voice control feedback

Any control command needs to have an appropriate feedback in order to provide reliable interactivity to the users.

4.5) Design and implement a simple traffic network navigator with GPS
We will use the basic control module we developed for ERTS in the group project.

5) System Requirements

5.1) Voice Control and Navigation
User says: "Take me to X"
Start a new journey.

User says: "Slower"
Reduce the throttle.

User says: "Faster"
Increase the speed.

User says: "Vehicle Stop"
Stop the vehicle (Pause the journey).

User says: "Go Ahead"
Start the vehicle (Resume the journey).

5.2) Voice Command, Query and Feedback
Computer says: "Command acknowledged, X"
Affirmatively acknowledge a command initiated by the user.

Computer says: "Sorry, X"
Negatively acknowledge a command initiated by the user.

Computer says: "We have arrived X, have a nice day!"
Report to the user for arrival of current destination.

Computer says: "We are through intersection"
Report to the user for vehicle going through the intersections.

5.3) Traffic Network Navigator
We will assume a simple traffic network graph (That is what normal commercial map data is like, such as the Tele Atlas [2] map data).

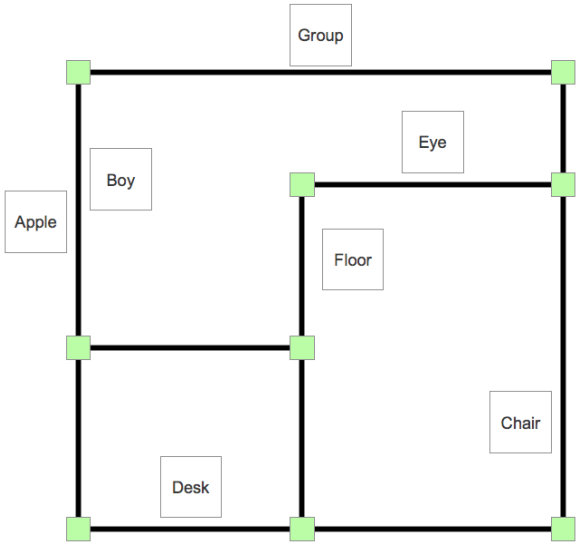

figure 1: a traffic network graph

The green squares represent the turning and junction points, the white squares represent the address block positions (such as 822 Queen Drive, Bloomington, IN). User can voice command the vehicle to go to any of the list addresses by saying their names. We will calculate a shortest route the vehicle to follow with considering the current position of the vehicle.

At this stage, we are assuming that the vehicle is meant to not go beyond this network. The black straight lines represent valid traffic routes. If we need to describe the real road conditions and shapes, we will need more “green squares” to “calibrate” the road shape.

Our navigation scheme can be re-used to work with any other “geocoding” [3] and “route service” [4] systems under the condition that we have Internet access.

6) Design and Implementation

6.1) Protocols

The system is mainly consisted of two components, the voice control module and the vehicle control module. The voice control module mainly deal with the voice recognition, voice synthesis, navigation control, and voice user interface. It is implemented under Java environment. The vehicle control module mainly deal with the ERTS vehicle model control. It is implemented under Python environment.

The communication channel between the voice control module and the vehicle control module is carried out using UDP [5] transport. Therefore, we designed simple “sequence number” scheme in order to cover the shortage of UDP, which is that UDP does not guarantee the certain delivery order as it is not a reliable transport protocol. However, UDP is faster in transmission than TCP [6] and we also require unreliable transport for real-time control. Because there is no needs to re-transmit an out-of-date packet.

In out defined protocol, voice control can initiate a “command” at any time, vehicle control can only acknowledge the command or report some updates. Once a command is initiated, voice control must wait for its acknowledgement before make another command. However, we will have a time-out checking in case that the “command” is lost during transmission.

The protocol is defined as following:
At time 1:
voice control -> vehicle control, [seq]:[command]:[command-parameters]
At time 2:
vehicle control -> voice control, [acked-seq]:[acked-command]:[acknowledgment]
At time 3:
voice control receives an valid ack, then voice control increases [seq]
otherwise
continues waiting for the valid ack
At any time:
vehicle control -> voice control, [update]:[update-parameters]
At command time-out:
voice control increases [seq], resets the command initiator so that it can allow user to make the next command

6.2) Traffic Network Map

We use a simple graph [7] to represent the assumed Traffic Network. Each vertex represents the intersection, each edge represents the road, the weight of the road is represented by meters.

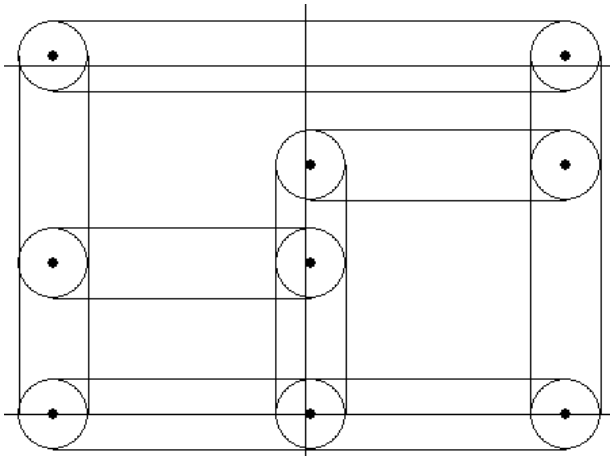

figure 2: built path network in ERTS

6.3) Route Search

We dynamically use a simple Depth-First Tree Search on the traffic network graph in order to find out the shortest route that with a given start point and a destination point. The start point is actually the current position of the vehicle reported by the vehicle control.

```
private void search(String start, String end, LinkedList<String> past) {
    past.add(start);
    String log = "past " + past + "\n";
    //logger.log(Level.INFO, log);

    for (PathEnd ends : net.get(start)) {

        if (!past.contains(ends.getEnd())) {
            if (ends.getEnd().equals(end)) {
                log = "get: " + end + "\n";
                //logger.log(Level.INFO, log);
                pathOutcome.add(past);
            } else {
                search(ends.getEnd(), end, new LinkedList<String>(past));
            }
        }
    }
}
```

figure 3: a code fragment of depth-first search by using recursion

6.4) Voice Conversation

As we promote using voice for controlling the vehicle, it is very sensible that the computer can give feedback to user in voice as well. Therefore, voice synthesis is required, and it is more preferred than using “pre-recorded speeches”. Due to resource and time limitations, we adopt FreeTTS[8] as our voice synthesis framework.

Then we implemented a simple synchronized conversation framework in order to make the voice control more user-friendly. The voice recognition module only listen to user’s speech once user clicks a “listen” button, until an utterance is recognized, the “listen” button function is disabled, which means we do not allow user to make any other speeches when process a current speech. When vehicle control responses, the voice synthesizer generates and speaks out a response speech. However, “listening to user” and “speaking out” are mutual exclusive, which means that we do not allow them to happen at the same time. The concurrency control is implemented using ReentrantLock of Java.

6.5) Enhancement of Voice Recognition

We have encountered a problem that sometimes the voice recognition cannot recognize well due to our pronunciation or noise in environment. We are not able to alleviate this problem by studying and tuning the voice recognition itself, however, we found another way to solve this problem. That is to increase the command complexity by increasing different words and number of words for commands. For example, firstly we set two speech commands as “go” and “stop”, but sometimes they cannot be recognized well. Then we use “go ahead” and “vehicle stop” instead, it turns out that the recognition correctness is better than previous approach.

```
#JSGF V1.0;

/**
 * JSGF Grammar for Hello World example
 */

grammar hello;

public <cmd> = take me to ( apple | boy | chair | desk | eye | floor | group ) | faster | slower | vehicle stop | go ahead;
```

figure 4: implemented voice recognition grammar

6.6) Vehicle Control

We mainly build our vehicle control module based on the ERTS control module implemented in the Lab Project [9]. Basically, we just added a few more functionalities.

6.6.1) Arriving Control

In the graph of our assumed traffic network, we could say that the edge between two vertices is the “street block”, and each address position belongs to one block. Once user commands vehicle to drive to an address, vehicle control will detect that when the vehicle enters the block that the destination address belongs to. Then the arriving control sequence is initiated: slow down the vehicle according to the current distance from the vehicle position to the destination position, if the distance to destination is larger than the distance calculated in the previous control-loop or the distance is 0 meters, then we say the vehicle is arrived.

6.6.2) Through Intersection

Once vehicle enters an assument intersection, the vehicle control will report to the voice control.

6.6.3) Throttle Control at Turning

Although we allow user to directly control the throttle of the vehicle, this is not allowed during the vehicle is making a turning at the intersection. The throttle is then controlled by the autonomous model.

7) Test

I have no changes to make another fields testing and capture a video. However, I look forward to have one after the report deadline if it is desired.

A testing in simulator:
http://www.youtube.com/watch?v=HFhelt0O9Ms

I have tested my individual project on the field, the communication connections and basic voice control functions worked fine. However, the traffic map I set is a little bit out of range for the real field, so we had to manually avoid the parked cars, light poles, and cones. I will adjust the traffic map for next testing, and next testing will be video recored.

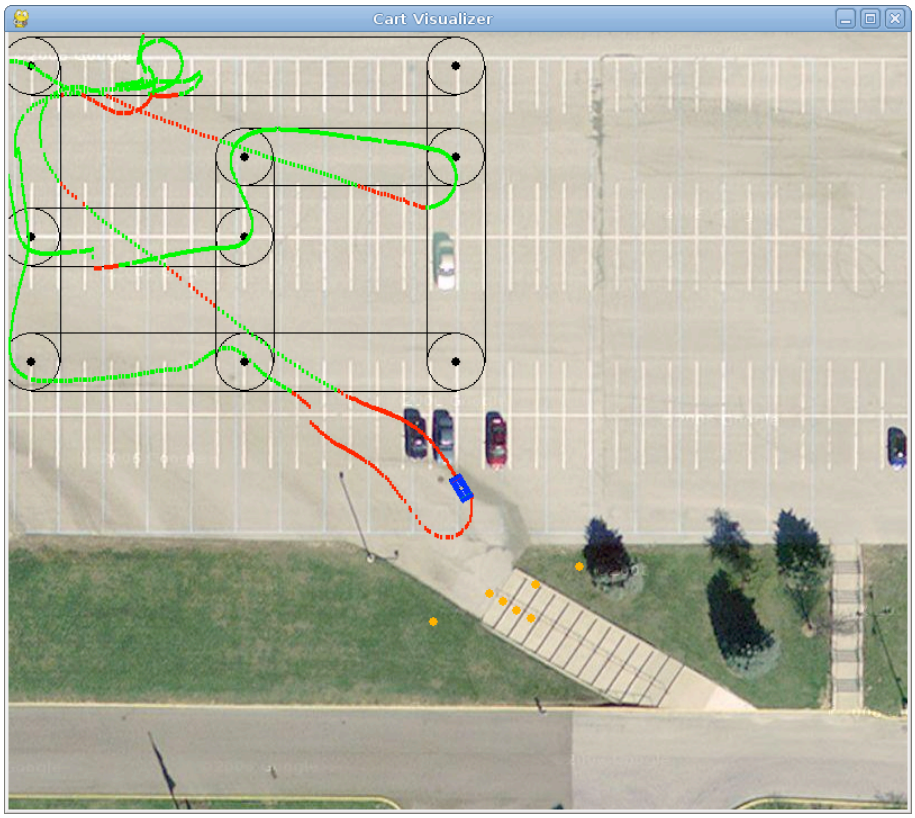

figure 5: a visualizer output of a field testing

8) Further Work

8.1)
More Voice Interactions and Information Feedback. Such as, user can ask "how long until we arrive destination?", "where are we right now?", and user can get real-time information feedback.

8.2)
More Voice Error Correction and Vocabulary.

8.3)

More ERTS Modeling for Real Navigation. A good starting point might be researching on "calibrate the real road shape", so that the vehicle can follow a course that has a more complicated shape (the real traffic road). And for the ability of the vehicle itself, how we make it to do a specific U-turn, how the vehicle do a parking action by itself, such functions might be desirable.

8.4)
More Safety and Reliability Enhancement based on ERTS Voice Control. Assume we can finally bring the vehicle to the real traffic road and use voice for navigation, there will be more concerns about how reliable and safe the voice control is. If I said something wrong, how can I quickly reverse it? Can ERTS itself validate my voice command if the command seems not sensible according to the real-time road condition?

9) Conclusion

In short, the technology platform for voice control is quite mature right now, and it is desirable for implementing more voice control functions for home-use vehicles.

10) References

[1] http://cmusphinx.sourceforge.net/
[2] http://www.geocode.com/, now owned by TomTom, http://corporate.tomtom.com/releasedetail.cfm?ReleaseID=319503
[3] http://code.google.com/apis/maps/documentation/geocoding/
[4] http://code.google.com/apis/maps/documentation/directions/
[5] http://tools.ietf.org/html/rfc768
[6] http://tools.ietf.org/html/rfc793
[7] http://en.wikipedia.org/wiki/Graph_(mathematics)
[8] http://freetts.sourceforge.net/docs/index.php
[9] https://www.cs.indiana.edu/svn/bhimebau/embedded_systems/class/fall11/shenhan/lab/5/