Auto-patrol robot car project proposal

Chunshu Wu happycwu@bu.edu
Leyang Yu yly@bu.edu
Shidong Sun Shidongs@bu.edu

March 12, 2019

1 Introduction

With the help of machine learning applications like facial recognition and all kinds of identification methods, cameras can be made very smart nowadays. However, as a kind of security device, the searching mode of the cameras are somewhat passive. So our project is to take one step forward and make the device mobile, by installing on a robot car which follows a scheduled patrol path. A picture will be taken and transferred to a remote computer once the target is found.

Audio recognition will also be applied. Not only to be fancier, but also allows the user to give easier and shorter instructions. The car should follow the user's voice instructions like "find me a happy man" or "start running" and give the correct feedback.

Last but not least, information transfer. IoT devices are supposed to be secure and we intend to make our device so. The information goes both ways, from the control unit to the car and vise versa. A secure information transfer channel will be established using the cryptography methods taught in class.

2 Resources

This project is not a sequel of our previous projects, but we are not building everything from the ground. Listed below are all the modules we need to build such a robot car, both hardware and software.

Building a robot car out of nothing is extremely hard in such limited

Figure 1: Car kit picture

time. So we decide to buy a robot car kit directly which has the motor driver board and sensors ready for us. Here "UCTRONICS Robot Car Kit for Raspberry Pi model K0073" is chosen because it has all the peripherals together with a stand for pi (see Figure 1). It has its own software and app, but we are not going to use them. (otherwise we miss all the fun!)

Camera selection. Let's make an assumption here. The line to be followed by the car should be placed on flat ground, and shock absorption matters are not taken into account. Therefore, we just choose one of the regular raspberry pi cameras (5 mega pixels, 1080p or 720p for videos, see Figure 2). The good news is, the car kit includes the exact camera we desire.

Figure 2: A simple raspberry pi camera

A remote computer. The user uses the computer to monitor the car status and receives the information of whether an object is found. The user also uses voice recognition to speak to the computer in order to send instructions to the car. In return, the car sends back a picture upon finding the target.

Board. There are a few choices of boards, including Arduino, Raspberry pi, and Beaglebone. The three boards are compared generally and we finally decide to choose Raspberry pi. Arduino is a perfect choice if there aren't many complicated operations, which is not very true in our case (We can use raspberry pi to control Arduino but that could be complicated and power consuming). Beagle bone is good and it has changeable output voltage, but it seems that beagle bone's specs are not as good as raspberry pi 3 b+, and the adjustable output voltage is not really necessary. Thus, raspberry pi 3 b+, as an all-purpose board, is preferable.

Software-wise, a machine learning platform can be utilized since there's not much computing ability in the car itself. Amazon Rekognition Video uses Amazon Kinesis Video Streams to receive and process a video stream, which could be used as our cloud computing platform.

Line following. With the sensors installed, Q-learning will be used to decide which direction the car is going as the car is running in real time.

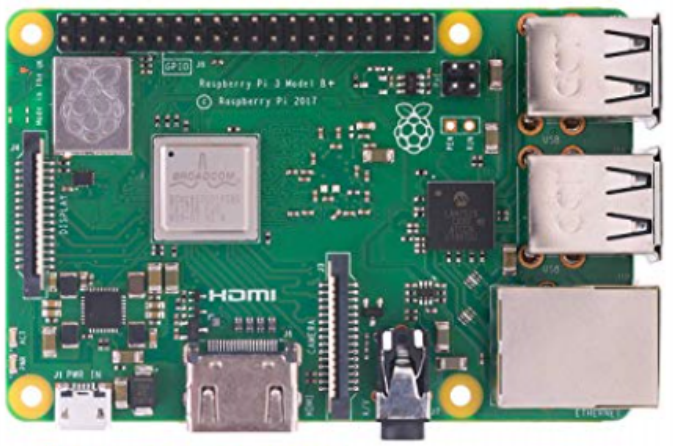

Figure 3: raspberry pi 3 b+

Voice recognition API. Since AWS is one of our project's essential component, we just as well use Amazon Transcribe for voice recognition so we have fewer accounts to manage.

Module coordinator. ROS is designed for this. It can interrupt other modules if there's message coming in. ROS can help connect the voice recognition, video recognition, and the car-driving modules so they can communicate to each other. Miscellaneous. OS, network, etc. For the most of the time we will develop using Linux 18.04 LTS. We may also test our code on Mac and Windows. Wi-fi will be used in our project since it's wireless and cloud services are needed. For better performance, symmetric keys (AES, for example) will be used to establish a secure channel. As for programming language, python and c++/c (g++6 compiler) are expected to be used.

3 Possible risks and risk management

AWS services may have steep learning curves. The learning process may slow us down. But there should be enough tutorial resources online so we can dig into them.

What if any of the parts are broken (especially the car)? This problem can be our tragedy. I'm pretty sure there are rooms with workbenches, electron irons, materials in BU somewhere. If this really happens, those rooms are the places we'll be in.

What if we need more computational power than we have? There shouldn't be very heavy workload on the pi itself, but it's still possible that it's super slow and the car not being able to follow the line precisely. If it's running python code, I believe there are many ways to have a better performance. Otherwise we may just need to slow the car down manually to meet the performance.

Coordinate the modules. ROS is designed for this, but it's also known for its complication. We anticipate that the modules are not correlated or intertwined so a small part of ROS will be used. Still, this may be hard. We will assign someone to work hard on this part and share the knowledge to the team.

Network problem. It's well known that BU 802.1x Wi-fi cannot be connected using raspberry pi. Indeed, we can use BU guest Wi-fi instead, but the speed may not be as fast. We may need to meet at one of ours' apartment and this may slow things down. Or we can find some place in BU that has less hard encrypted Wi-fi.

4 Technical Approach

The project can be divided into 4 major modules: Real-time video analysis, car automation and line following, voice recognition, and communication.

Real-time video analysis. Our video stream will be uploaded to AWS kinesis and processed using AWS Rekognition. This module is relatively independent from the car. We just need to go through the documents and tutorial videos and set things up. The objects found by rekognition are classified using labels. We can search the labels of our interest to determine if the object is found.

The car is expected to be a tricky module. First, we assemble the car and test its basic functions. Then we use the sensors to get data and use q-learning to steer the car. Finally, install the camera on the car and test the performance (if the video is blurry, we may need to reduce the car speed)

Voice recognition. This part is done on a remote computer. So the interface between the module and the other modules is the problem we need to focus on. Our idea is that we use voice recognition API to translate the voice signal into keywords, and send the keywords to the video recognition module. For example, "find, happy, man". "find" triggers the searching function, and "happy, man" are the keywords to be searched for.

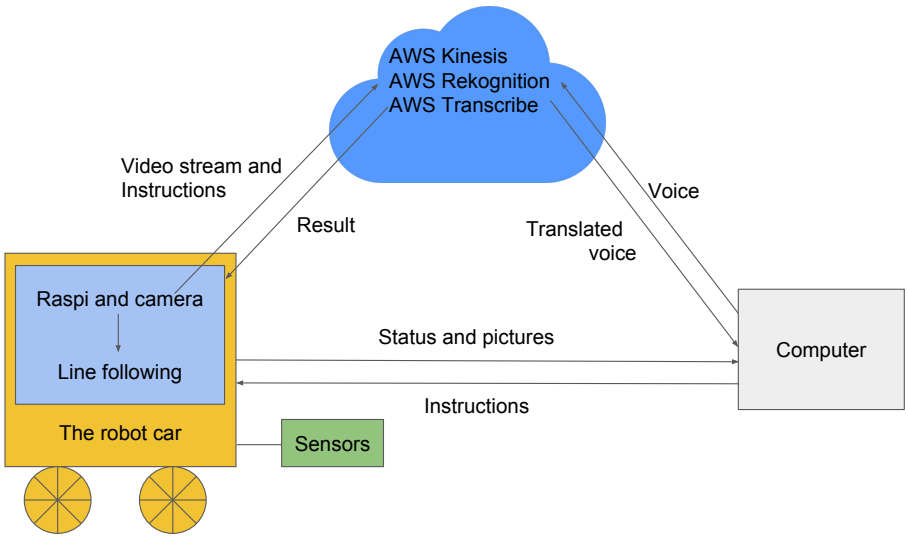

Figure 4: System Diagram

Communication. The modules talk to each other using ROS. The video recognition module and the car automation module listen to the voice recognition module. The communication channel between the pi and AWS is secure, so is the channel between the remote computer and AWS. However, we still wish to monitor the car status and receive the target picture directly with our computer. Therefore our own security channel shall be built (Seems ROS does not have a fully-developed security system, SROS is under development though).

As for responsibilities, Shidong has more experiences with robots and hardware, therefore configure the car and the sensors. Leyang is familiar with network, so he is in charge of the problems related to communication. Chunshu is more software oriented, and he will be leading the coding part. We'll make sure everyone participates in everything so everyone learns about everything.

If our project turns out to be easy, we will add more parts to it. The system diagram is shown in Figure 4.

5 milestones

The first stage is to make every module work independently.

Week 1: While we are waiting for all the purchased materials to arrive, we can manage to learn AWS video recognition services. For a start, we need to be familiar with the system and try their examples with our recorded videos using our raspberry pi's, considering that the expenses increase with the data size. After we pass all the tests with recorded video, the real-time video recognition service will be tested.

Week 2: All the stuff we need should be at hand now. Our team will work in parallel, finishing the car installation and learning how to use AWS voice recognition service. All of the car's basic functions shall be fully tested without a line to follow. For voice recognition, we also use recorded audio first to ensure that the module is robust, then use real-time audio in order to control the expenses.

Week 3: According to the system diagram, only line following and secure communication channel between the computer and the raspberry pi need to be worked on. We'll come up with the Q-learning code in C/C++ and use it to determine the car's next move on the line. In the meantime, The security channel should be established so the computer can exchange data with pi. The encryption details and methods are to be determined later.

The next stage is to test things together and debug.

Week 4: Now each of the modules can work independently, so the next step is to put things together and test. We start by testing if the video recognition works on a running car, then with line following included. There could be many unexpected problems so we plan to take our time on this step. Afterwards, add voice command to the system and test.

Week 5: Continue testing if not enough. Secure information transferring will be added to the system in this week. So far everything should be included. We run several more general tests to make sure the entire system is robust and intact.

The final stage, paper work.

Week 6 and 7: Now to do the paper work. A detailed manual shall be uploaded to github, with all the procedures and steps to install our software. We will prepare for the final presentation (if any) and write the final report. This part is not as hardcore as before, but time consuming. Therefore the last 2 weeks are planned as such.