CSC714 Project Final Report, Spring 2014

Android based NXT Steering

Lei Wu
lwu4@ncsu.edu

Project webpage: http://www4.ncsu.edu/hidden/csc714-project.html

Introduction

Due to the limited resources (i.e., sensors) provided with the NXT vehicle, the steering (i.e., including steering autonomously and avoiding obstacles when roaming around the floor of the lab, see the description of problem 2 in HW3 [1] for details) is not easy (if not impossible). Conversely, smartphone devices (e.g., phone or tablet) always provide plenty hardware (e.g., gyro, compass [2] and camera [3]) to support lots of functionalities.

In this project, we try to provide the NXT vehicle more information with Android device to assist the steering work. Specifically, an Android phone will be bound together with the NXT vehicle, and they will communicate with each other through the Bluetooth. When steering, the Android phone will detect the environment, and send the analyzed results to the NXT vehicle. The NXT vehicle then will make decision on roaming by considering both the received information and those collected by its own. We believe that the Android device is a good support to enhance the ability of the NXT vehicle.

We have designed and implemented a prototype to demonstrate the idea. Although the performance is not as good as our expectation (due to the unstable/unreliable detections and some other issues), we successfully collected and analyzed related information, and sent them to the NXT to assist the steering work. In the following, we will describe and discuss the design, implementation and challenges of our system respectively.

System Design

Figure 1 depicts the framework of the proposed system. Basically, the Android phone communicates with the NXT vehicle through Bluetooth. Thus we have a communication module in Android side as "master", while the NXT is treated as "slave" and we have a corresponding task named *BTTask* to receive messages from Android phone. The *master*, i.e., Android communication module, performs more work than the *slave*, including enable/disable Bluetooth, connect/disconnect with the NXT vehicle (whose address is hardcoded) and send out collected messages asynchronized (it is handled by a new thread). *BTTask* in NXT side interprets the received messages according to the predefined protocol and delivers them by using resource protection mechanism of NXT.

There are another two Android modules: sensor processing and image processing. The former catches detected data from *accelerometer* sensor and *magnetic field* sensor, and calculates the corresponding orientation information. The latter is able to analyze the image/video capture of the environment observed by the camera. Particularly, it tries to detect the relative position of the track (i.e., the two blue lines). All collected and analyzed results will be sent to the NXT through the communication module.

Beyond *BTTask*, we may have different tasks to serve our purpose. For example, we can have one task to detect the obstacles by using the touch sensor. In all, the NXT shall utilize all information received from the Android side and collected by its own to guide the steering work.

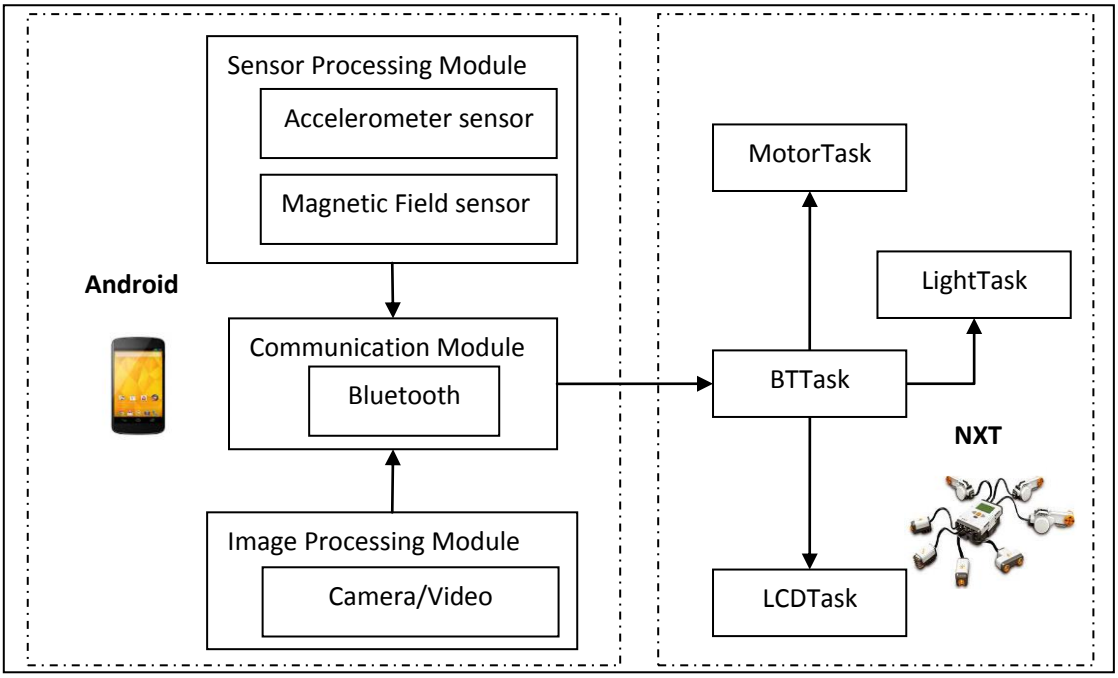

Figure 1: System Framework

Implementation and Challenges

We chose Nexus 4 (Android 4.4 with build number KRT16S) and the basic NXT components provided by the professor [4] as the experimental platforms. The whole system is Java and C code mixed (i.e., the Android side is all Java code, while the NXT side is C code). The system UI of the Android side is shown in Figure 2.

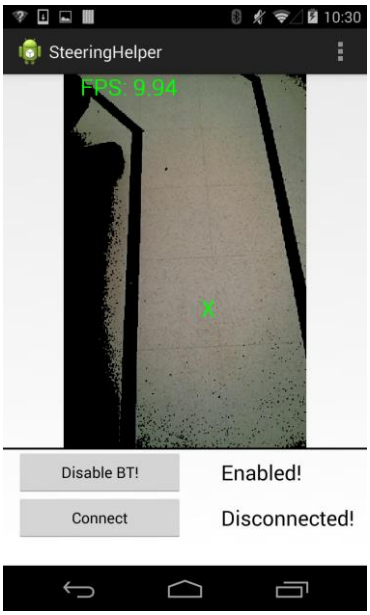

Figure 2: System UI of Android Side

We implemented the communication module and *BTTask* mainly based on [5]. However, some modifications are necessary to make them work properly. For the communication module, for example, the *createRfcommSocketToServiceRecord* API call is not useful in the recent Android devices (< API level 10), we shall use createInsecureRfcommSocketToServiceRecord instead (a more compatible way is to select them according to the Android version). Furthermore, the corresponding *send* method needs to be modified to support multiple bytes delivering. For the *BTTask*, sample code in [5] does not work (because we used C rather than C++). Fortunately we have figured out the right way for communication.

For the Android side, the sensor processing module somewhat is easy to implement, i.e., just implementing the interface *SensorEventListener* and record information once there is any updates. The key point is to calculate the orientation from the collected data, and [6] provides a good reference for the calculation. However, due to some unknown issue (maybe the unstable way to place the phone?), the two sensors always specify *SENSOR_STATUS_UNRELIABLE* warning on the accuracy (which has to be ignored otherwise no sensor data can be fetched).

The image processing module is the most complicated component in our implementation. The goal is to calculate the offset of the current position away from the middle point of track (i.e., the two blue lines), which at least can help the NXT to adjust the speed of its left and right motors. Initially we tried to perform the image recognition by ourselves, which based directly on the snapshots from the camera

and sample objects (i.e., the pictures of the two blue lines of the track). However, such approach has been proven infeasible because the pictures of the sample objects always change with the motion of the NXT vehicle. Thus we adopted an indirect way to solve this problem. It derives from the Erik Hellman's sample code [7] of camera frame by using OpenCV [8]. First of all, we can get camera frame data by using *VideoCapture* provided by OpenCV. Then we filter the other colors but the track (i.e., the two blue lines), and use the *FeatureDetector* functionality also provided by OpenCV to get the key points of the remainder color blobs. After that the average position can be calculated.

Figure 3 just gives the effects of part of the image processing. The left screenshot is the one before the filtering, while the right is the one after the filtering. The green *X* indicates the calculated average position. Note that in these two screens the average positions are almost the same, however, if there exist many noises (e.g., obstacles), the average position before filtering is quite bad.

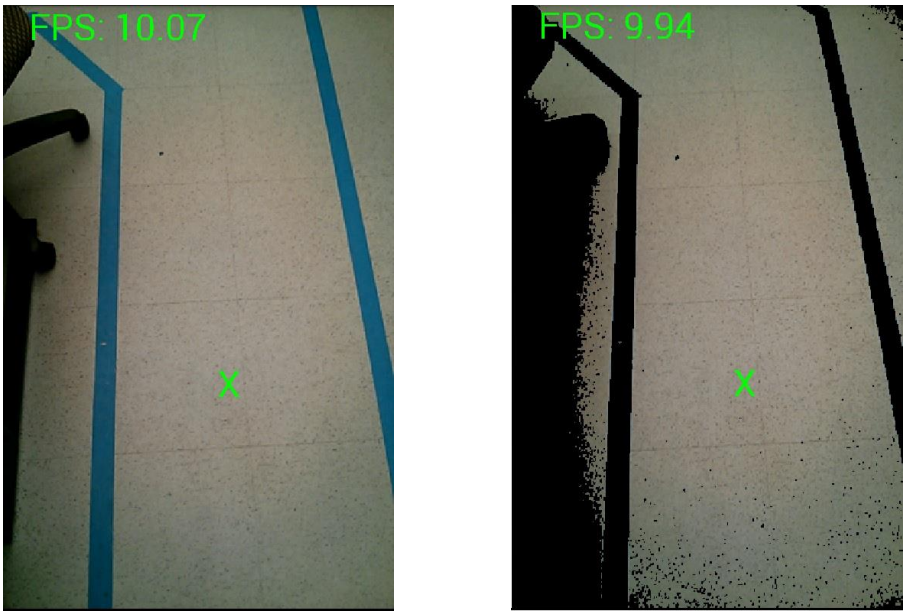

Figure 3: Screenshots before and after filtering

For the NXT side, we have implemented three tasks in our current prototype besides the *BTTask*. As shown in Figure 1, the *MotorTask* is used to update/set motor speed, the *LightTask* is used to detect track by using the light sensor and the *LCDTask* is used to display information on the LCD. We have implemented these tasks based on a very simple strategy, but the performance is not quite good as what we expected. Thus its improvement is the most important and challenged work in the future.

Discussions and Limitations

As mentioned before, we are able to analyze the collected data and provide potential useful information for the NXT from the Android device. However, there are still some challenges need to be considered. For example, the two sensors and camera preview are quite sensitive, and can be affected by even very small motions. In the NXT side, we have not found a solution/algorithm which is able to effectively and efficiently guide the steering. Our current prototype just implemented a straightforward strategy which definitely does not take full advantage of the collected information.

Reference

[1]http://courses.ncsu.edu/csc714/lec/001/hw/hw3/hw3.html
[2]http://developer.android.com/guide/topics/sensors/index.html
[3]http://developer.android.com/guide/topics/media/index.html
[4] http://courses.ncsu.edu/csc714/lec/001/
[5]http://stackoverflow.com/questions/4969053/bluetooth-connection-between-android-and-lego-mindstorm-nxt
[6] http://stackoverflow.com/questions/14433182/direction-using-sensor
[7] http://developer.sonymobile.com/knowledge-base/tutorials/android_tutorial/get-started-with-opencv-on-android/
[8] http://opencv.org/platforms/android.html