Voice Commendable Robotic Plotter by Using Mapping Techniques

Aju Johnson*
Jasna V P.*
Muhammed Nadeem K K.*
Naseera C*
Nithin R.**

Abstract

The advent of new high-speed technology and the growing computer capacity provided realistic opportunity for new robot controls and realization of new methods of control theory. This technical improvement together with the need for high performance robots created faster, more accurate and more intelligent robots using new robots control devices, new drives and advanced control algorithms. This project is to wirelessly control the movements of a four-wheeled robot through voice, locate the exact position of the robot and identify the presence of living beings. Our project is designed in a way, that utilizing optimum man power. For that purpose there is software running in the PC that can recognize the voice commands by the user. The path taken by the robot is drawn in the computer monitor using mapping techniques. By analyzing the path we can determine the correct location of the robot. The presence of living body is identified by using PIR sensor that is attached with the robot. Another Feature is Obstacle Sensing, Direction Changing and Collision avoidance; this is also the very important and lively feature of the Robot. There are many locations where the map is not available as a priori. So when we make use of this robot firstly in such an environment, we can trace the path taken by the robot and can proceed reliably through that path.

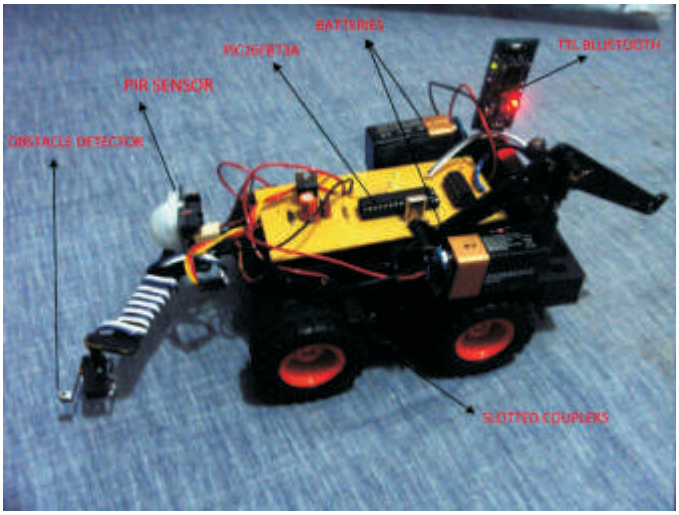
Keywords - Mapping, SAPI, Sensors, obstacle avoidance.

1. Introduction

One difficult problem in robotics is localization - the ability of a mobile robot to determine its position in the environment. Robotcists around the globe have been working on finding a solution to localization for more than 20 years, however, only in the past 4-5 years have we seen some promising results. But these solutions work in rather structured environments and tend to fail in realistic settings. Furthermore, systems using these solutions require expensive sensors, which make the solution infeasible for many consumer and commercial applications. In this work, we describe a technology for localization that requires only one optical wheel encoder. So this project aims at making a system that employs various techniques to collect environmental information such as root mapping, presence of living body and obstacles presence. Each of these readings has its own importance/applications under different situations, and hence the system to be developed here is expected to find diverse applications. The system is planned to be built around a trolley which will carry the transmitter (ROBOT) section consisting of sensors to remote places for data acquisition. It also contains the wireless module for wireless data communication. The transmitted signal is to be received at an operator section and displayed on PC. The control of robot is by using voice commands. The sensors as well as other modules shall be so

*Computer Science and Engineering, MEA Engineering College, Vengoor, Kerala, India

chosen/designed to withstand the atmospheric conditions of temperature, pressure and humidity which the system should face as a whole. The system has got two sections- TRANSMITTER (ROBOT) and RECEIVER (operator).The transmitter section runs over a trolley with inbuilt Bluetooth transceiver and hence is not a part of system design. The robot may use spectrum in the range 2.4~2.524 GHz and is never expected to make any interference with other radio devices included. A PIR sensor is provided at ROBOT. It will generate signals when any living body is detected. It can detect motion up to 6 meters. The robotic module also includes one optical wheel encoder and a switch. The wheel encoder is a slotted coupler and is used for measuring the number of wheel rotations. Switches are used for detecting obstacles in its path. The sensor data handled by the controller on ROBOT side is send to the operator side as serial data via Bluetooth module operating on both sides at 2.4~2.524 GHz. TTL Bluetooth module is expected not to face any interference from other wireless modules. On chip UART module handles serial transmission and reception of data via Bluetooth module within 10m.The entire system receives regulated power supply at 12V, 5V and 3.3V for each of the modules and sensors. The supply may have a battery as power source. The Visual Studio 2008 operator Interface running on PC provide operator, the interface for viewing the sensor data received as well as to control the motion of trolley with voice commands. The user can see the plotting of path in the same interface. An additional user interface is provided as user friendly option for modifying the voice commands that is already stored in the database. Robot is being controlled by the PIC controller at receiver section based on voice commands from PC. The system shall be encased in a protective case that can withstand the atmospheric conditions prevailing in the area of operation.



1.1 Living Body Identification

This Passive Infrared Sensor (PIR) module is used for motion detection. It can be uses as living body detector on this robot. It can work from 5V to 9V DC and gives digital output. It requires 10-60 seconds of settling time before starting its operation. It consists of pyroelectric sensor that detects motion by measuring change in the infrared levels emitted by the objects. It can detect motion up to 2 meters. Pyroelectric devices, such as the PIR sensor, have elements made of a crystalline material that generates an electric charge when exposed to infrared radiation. The changes in the amount of infrared striking the element change the voltages generated, which are measured by an on-board amplifier. The device contains a special filter called a Fresnel lens, which focuses the infrared signals onto the element. As the ambient infrared signals change rapidly, the on-board amplifier trips the output to indicate motion.

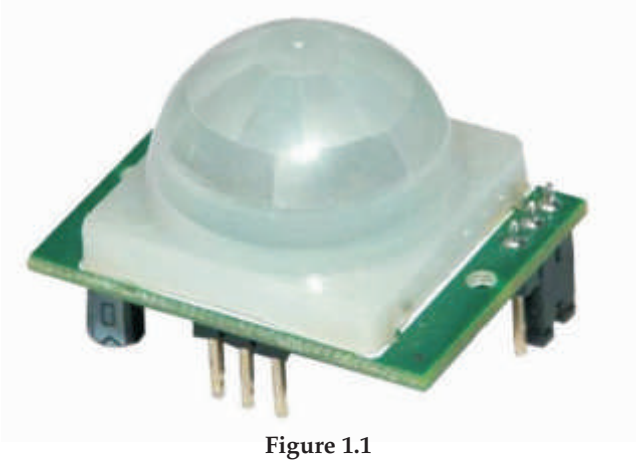


Figure 1.1

1.2 Obstacle Detection

For obstacle detection we are proposing Leaf Switches. Leaf switch is an electric switch with a button and two strips of metal that make contact when the button is pushed. The circuit is only closed when the button is held down. So these switches are best suited for generating short pulses that trigger an action in the circuit. In this project it is used for detecting obstacles in the robot's path. It is projected out from the robotic platform. So when an obstacle is encountered, the button will lock and generate signals. Obstacle avoidance is difficult in this project because it makes our map building quite complex.

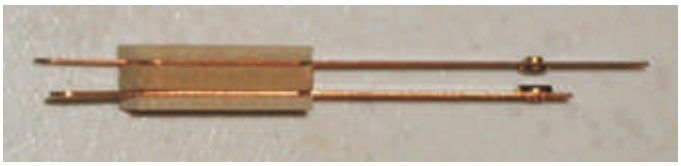


Figure 1.2

1.3 Calculation of Distance Covered by the Robot

Slotted couplers consist of an infrared emitting diode facing a photo detector in a molded plastic housing. A slot in the housing between the emitter and the detector provided a means of interrupting the signal. It will generate signals when wheels spoke pass through these slots. In this project it can be used as a counter for counting the total number of tyre rotations. By multiplying perimeter of tyre with this rotation value the total distance that travelled by the robot can be measured.

The picture of a slotted coupler is shown in figure 1.3. From that diagram it is clear that a slotted coupler mainly consists a IR emitter and collector. If that ray undergoes cutting it will results in some interrupts and can be used for incrementing a variable.

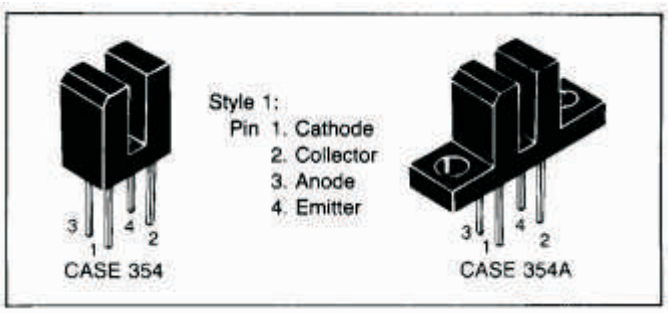


Figure 1.3

2. System Model

2.1 General Architecture of Mapping

In this architecture the highlighted path by red color is used for this project. In actual SLAM once the features of the environments are extracted (lines, corners or both) they are compared with the predicted features previously added to the SLAM system state. If there is a correct association between -at least- one feature of the SLAM system state with a recently extracted feature, then the SLAM system state and its covariance matrix are corrected according to the correction stage. If there is no appropriate association between the observed features and the predicted ones, the feature is added into the parallel map (if the feature is a line, then their beginning and ending points are incorporated into the parallel map; if it is a corner, then its Cartesian coordinates are added). When a correction stage is successfully executed, the values stored in the parallel map are also updated according to the changes of their corresponding features in the SLAM system state.

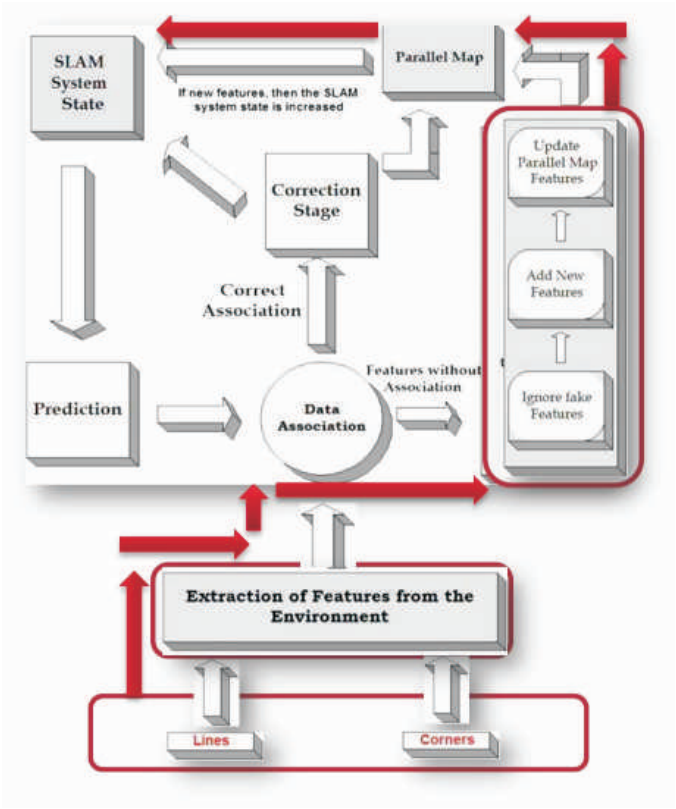


Figure 2.1

In this system we design a algorithm for mapping by following this architecture. According to that algorithm for mapping the total number of tyre rotations are counted by the mechanism implemented in the robot and will inputted to the algorithm. Next the algorithm will listen to any other commands from the computer. It may be any direction controlling commands. If such a command is received the robotic system will again inputted the next number of tyre rotations to the algorithm. If the number of rotations are received then the algorithm will invoke the drawing tool and draw the map using the current direction and multiplied value of tyre rotations with the perimeter of tyre. That will results in the mapping of path that followed by the robot.

2.2 Voice Recognition

We are not aiming to build software which can recognize a lot of words. Our basic idea is to develop some sort of menu driven control for our robot, where the menu is going to be voice driven. Recognition strength of a few words would do for such kind of jobs. A person interacting with such a system would not need to use his hands for routine jobs, which is what we wish to achieve. This leads us to our main task in the project. What we are aiming at is to control the robot using voice commands. So software which can recognize and distinguish the 5 commands from one another will do the job. So software needs to be developed which takes voice data as input & outputs the matched command. For voice recognition we are make use of the SAPI package from Microsoft. The recognizer uses the notion of a recognition context to identify when it will be active. An application may use one context for each form that will use SR, or several contexts for different application modes (Office XP has a dictation mode for adding text to a document and a control mode for executing menu commands). Recognition contexts enable you to start and stop recognition, set up the grammar and receive important recognition notifications. Part of the process of speech recognition involves deciding what words have actually been spoken. Recognizers use a grammar to decide what has been said, where possible. SAPI 5.x represents grammars in XML. In the case of dictation, a grammar can be used to indicate some words that are likely to be spoken. It is not feasible to try and represent the entire spoken English language as a grammar so the recognizer uses its own rules and context analysis, with any help from a grammar you might supply. With Command and Control, the words that are understood are limited to the supported commands defined in the grammar. The grammar defines various rules that dictate what will be said and this makes the recognizer's job much easier. Rather than trying to understand anything spoken, it only needs to recognize speech that follows the supplied rules. A simple grammar that recognizes three commands might look like this.

```
<GRAMMAR LANGID="809">
<!-- "Constant" definitions -->
 <DEFINE>
   <ID NAME="RID_start" VAL="1"/>
 </DEFINE>
<!-- Rule definitions -->
<RULE NAME="start" ID="RID_start" TOPLEVEL="ACTIVE">
 <L>
   <P>forward</P>
   <P>backward</P>
   <P>stop</P>
 </L>
```

</RULE>
</GRAMMAR>

The GRAMMAR root node defines the language as British English ($809, American English is $409). In similar way we can make a grammar for controlling the robot. After the correct command recognition the designed software in c# will look for any corresponding action. The actions are stored in a database which has two attributes. The corresponding action for each command will send into the robot via serial port. The Speech Application Programming Interface or SAPI is an API developed by Microsoft to allow the use of speech recognition and speech synthesis within Windows applications. To date, a number of versions of the API have been released, which have shipped either as part of a Speech SDK, or as part of the Windows OS itself. Applications that use SAPI include Microsoft Office, Microsoft Agent and Microsoft Speech Server. In general all versions of the API have been designed such that a software developer can write an application to perform speech recognition and synthesis by using a standard set of interfaces, accessible from a variety of programming languages. In addition, it is possible for a 3rd-party company to produce their own Speech Recognition and Text-To-Speech engines or adapt existing engines to work with SAPI. In principle, as long as these engines conform to the defined interfaces they can be used instead of the Microsoft- supplied engines. In general the Speech API is a freely-redistributable component which can be shipped with any Windows application that wishes to use speech technology. Many versions (although not all) of the speech recognition and synthesis engines are also freely redistributable. Broadly the Speech API can be viewed as an interface or piece of middleware which sits between applications and speech engines (recognition and synthesis). In SAPI versions 1 to 4, applications could directly communicate with engines. The API included an abstract interface definition which applications and engines conformed to. Applications could also use simplified higher-level objects rather than directly call methods on the engines. In SAPI 5 however, applications and engines do not directly communicate with each other. Instead each talk to a runtime component (sapi.dll). There is an API implemented by this component which applications use, and another set of interfaces for engines. Typically in SAPI 5 applications issue calls through the API (for example to load a recognition grammar; start recognition; or provide text to be synthesized). The sapi.dll runtime component interprets these commands and processes them, where necessary calling on the engine through the engine interfaces (for example, the loading of a grammar from a file is done in the runtime, but then the grammar data is passed to the recognition engine to actually use in recognition). The recognition and synthesis engines also generate events while processing (for example, to indicate an utterance has been recognized or to indicate word boundaries in the synthesized speech). These pass in the reverse direction, from the engines, through the runtime dll, and on to an event sink in the application. In addition to the actual API definition and runtime dll, other components are shipped with all versions of SAPI to make a complete Speech Software Development Kit. The following components are among those included in most versions of the Speech SDK:

a) API definition files - in MIDL and as C or C++ header files.
b) Runtime components - e.g. sapi.dll.
c) Control Panel applet - to select and configure default speech recognizer and synthesizer.
d) Text-To-Speech engines in multiple languages.
e) Speech Recognition engines in multiple languages.
f) Redistributable components to allow developers to package the engines and runtime with their application code to produce a single installable application.
g) Sample application code.
h) Sample engines - implementations of the necessary engine interfaces but with no true speech processing which could be used as a sample for those porting an engine to SAPI.
i) Documentation.
j) Shared Recognizer. For desktop speech recognition applications, a recognizer object can be used that runs in a separate process (sapisvr.exe). All applications using the shared recognizer communicate with this single instance. This allows sharing of resources removes contention for the microphone and allows for a global UI for control of all speech applications.
k) In-proc recognizer. For applications that require explicit control of the recognition process the in-proc recognizer object can be used instead of the shared one.
l) Grammar objects. Speech grammars are used to specify the words that the recognizer is listening for.

   SAPI 5 defines an XML markup for specifying a grammar, as well as mechanisms to create them dynamically in code. Methods also exist for instructing the recognizer to load a built-in dictation language model.

m) Voice object. This performs speech synthesis, producing an audio stream from text. A markup language (similar to XML, but not strictly XML) can be used for controlling the synthesis process.
n) Audio interfaces. The runtime includes objects for performing speech input from the microphone or speech output to speakers (or any sound device); as well as to and from wave files. It is also possible to write a custom audio object to stream audio to or from a non-standard location.
o) User lexicon object. This allows custom words and pronunciations to be added by a user or application. These are added to the recognition or synthesis engine's built-in lexicons.
p) Object tokens. This is a concept allowing recognition and TTS engines, audio objects, lexicons and other categories of object to be registered, enumerated and instantiated in a common way.

3. System Design

The total system is represented through two designs. The first one is map building. Second one is working environment with obstacles and living body

3.1 Map Building

For map building we are make use of slotted coupler's output that is the total number of tyre rotation. By multiplying this value with the perimeter of tyre we can calculate the total distance covered. By the help of command, that is inputted through the voice and distance the map can be drawn.

3.2 Working Environment with Obstacles

The robot is capable of identifying obstacles that is present in its

path. It is also capable of differentiate a living body from other non-living body. If an obstacle is identified automatically the system will terminate, that is stop its movement. And also the presence of obstacles will plotted in the map.

4. Experimentation and Results

4.1 Map Building

The map building is start with the reference point according to the algorithm that is, initialize the starting from an arbitrary initial point, the mobile robot should be able to explore the environment with its on-board sensors. Knowledge about it is gained. The scene is interpreted. Build an appropriate map is build. The output is shown in figure 4.1

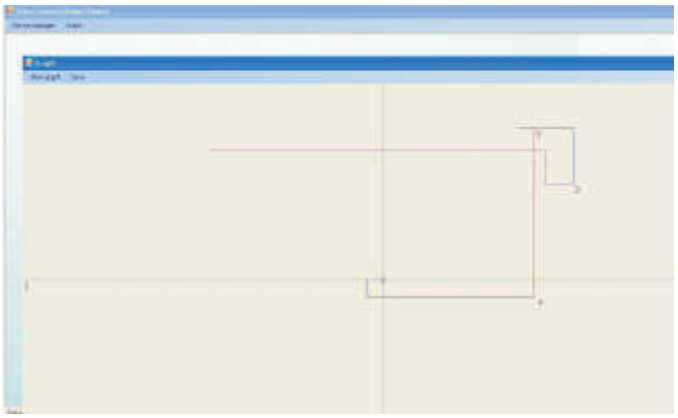


Figure 4.1

4.2 Obstacle Detection

Obstacle detection is one of the basic tasks required of most mobile robots. Range-based sensors provide effective means for identifying most types of obstacles facing a mobile robot. In this case we propose switches for this purpose. It will simplify both obstacle detection and map building but has so many disadvantages. The Robot has come across an obstacle and would stop its further movements automatically and will inform its presence through the output map. It is shown in figure 4.2

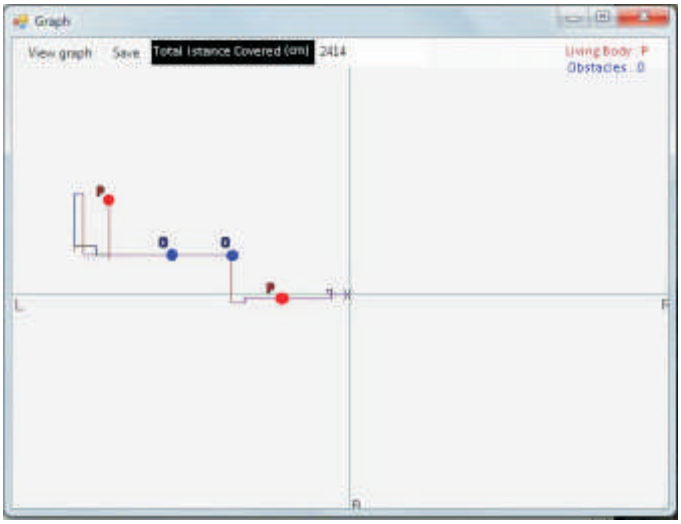


Figure 4.2

5. Conclusion

The indoor experiments were performed in an office environment. A training set is obtained by manually operating the Robot through the environment. A test dataset is generated through this operation. Localizing in this manner provides an accurate representation of the office environment by map building using the very basic requirements for map building.

6. References

1. *Paolo Pirjanian, Niklas Karlsson, Luis Goncalves "Low-Cost, Visual Localization and Mapping for consumer robotics".*
2. *Thrun, S. "Mapping: A Survey". Technical Report, CMU- CS-02-111, Carnegie Mellon University, Pittsburgh, PA,USA, February, 2000.*