BACKGROUND RESEARCH ASSIGNMENT

How to submit: Place a PDF document with the link/title "Background Research Assignment" in your group's resources folder and make a link to this on your group's Wiki page. Your page must be readable on a web browser. What to submit: Two-page paper per group, formatted as specified below.

You are to gather, summarize, and analyze existing work on your project topic. The purpose of this assignment is for your group to obtain a thorough picture of what design approach have been explored and the issues that have arisen. Your paper should reflect this understanding so that it can be a starting point for your own project design. Your paper should not be a series of summaries nor should it be a list of who has done what.

Your paper should include a list of references at the end for any material discussed in the paper. Your paper should be in 12 pt font, single-spaced, with 1" top, bottom, left, and right margins.

Created by:

Danny Duangphachanh
Leah Krynitsky
Brian Hilnbrand
Igor Janjic

September 11, 2014
[ECE 4534] Embedded Systems Design
Virginia Polytechnic Institute and State University

Background

Of the first autonomous robots, none are more famous than Grey Walter's tortoises Elmer and Elsie. These two robots were very simple analog circuits equipped with three wheels for locomotion and light sensors which they used to navigate their environment and move toward illuminated recharging stations when low on power [8]. Since then, the design of autonomous robots has shifted toward the processing of sensory data using algorithms based in terms of digital computation. The remaining paper focuses on this last view of autonomous robotics.

Robot Framework Considerations

The three primitives of robotics: plan, sense, act, as well as their relationships imply a robotic framework. Since the design specifications and hardware limitations are given, the robotics framework must then be determined by judging each framework's strengths and weaknesses as well as the compatibility between the framework and the choice of available sensors (described later). Thus, we are purposefully limiting our discussion to only those frameworks that coincide with usability of the various sensors available.

The framework must be able to handle quickly processing large amounts of complex information. In the 1960's, development of a roving vehicle for surface exploration of the planet Mars demonstrated the need to have the framework on the vehicle to help eliminate lag. This implementation proved useful when tested with terrain that was not smooth and level. [5]

Additionally, the framework must be able to handle the prioritization of tasks. Since the robot has multiple goals that it is trying to achieve, often at the same time and often contradictory to each other, the control system framework must be able to rank goals by priority and focus computational power to solving high-priority goals first while at the same time reserving enough computational power to service low-priority goals.

The more sensors the robot has, the more information it is able to gather and then process to help it achieve its goals. The framework must be able to handle error in the sensors, overlapping sensors, inconsistent or failed sensors, and the addition of more sensors,

Layered Control Paradigm [1]

This paradigm was designed by Brooks, R. at the MIT Artificial Intelligence Laboratory, which allowed for the design of various levels of intelligence in autonomous robots by building the control system in layers. Each layer is composed of simple computational modules communicating over low bandwidth channels asynchronously. The problem formulated in the language of control theory is as follows: a group of sensors receive information about the environment which the control system processes, turns into commands, and then sends to the motor controller which moves the robot accordingly.

Choice of Sensors

The sensors implemented in this design will be used mostly for obstacle detection and navigation. Some of the most commonly used sensors for these tasks include distance sensors and proximity sensors. These types of sensors are used to determine the relative distance between the sensor and objects in the environment and can be either passive or active.

The main type of passive sensor used in object detection is vision based, like a visible spectrum camera. The images captured by this type of system provide a large amount of information and are easy to interpret. Since passive sensors do not emit any signals, the environmental interference issues that can arise with active sensors are not present. Although passive sensors are advantageous in this regard, they perform poorly in low light environments since they must rely on natural energy in the environment. Alternatively, active sensors provide their own energy source for object detection and dramatically outperform passive systems in environments lacking illumination. Radar, ladar, and sonar are the three main types of active sensors that can be implemented in an object detection system, and all implement a system that emits a signal and interprets reflections to determine information about an object. [2]

Infrared sensors, available as passive or active sensors, are very widely used and offer many advantages over other types of sensors when performing navigation, object avoidance, and line following. Infrared sensors are also cheap in comparison to other sensor choices and are able to operate in real-time, making them a popular choice for designs like this one. [3]

When designing a sensor system, more than one type of sensor can be used when multiple sensors are to be implemented. As a result, many designs optimize performance by creating a system which includes both active and passive sensors, thus achieving the advantages of each and eliminating most of the disadvantages that come from using only one type of sensor. A design which implements a mix of active and passive sensors improves both system performance and object recognition. [2]

Comparison of Algorithms

The objective of navigating an unknown terrain can be accomplished by an assortment of navigation methods and their execution and completion times vary. These local path planning algorithms use sensors to realize the surrounding area, compute the fastest and/or shortest route, proceeds with the trajectory, and repeats the process until the autonomous robot is at its destination.

Normally for local path planning, a robot begins with a destination and is guided with a straight line and follows the line until it senses an obstacle. Once an obstacle is found, the robot performs obstacle avoidance by updating the route, the new distance, and continues running. This general approach is used in bug algorithms such as Bug1, Bug2, DistBug, VisBug, and TangentBug.

The variations of the mentioned bug algorithms differ in their importance over aspects to find a better route. Bug1 and Bug2 moves toward the target and circumnavigates the obstacle and continues. The DistBug algorithm also incorporates range sensors, but also makes use of odometry and destination. The VisBug is the first bug family algorithm that integrates a range sensors and calculates shortcuts relative to the destination. In addition to what the DistBug algorithm is capable of, the TangentBug uses range sensors with a 360-degree orientation resolution to completely see its surroundings. [6]

Communication Protocols

In general terms, a message intended for communication between two devices (the protocol) is comprised of two main parts: the header and the payload. The header consists of information about the message, including source or destination id, the nature of the message, and the size of the payload. The payload is the data which is being sent from one device to another. This payload can be formatted in any way, as long as it is consistent in both the source and destination.

Google Protocol Buffers [4] are used in the automotive industry to format and serialize both small and large packets of information, preparing the payload for transmission to any destination. Once received at the destination, the payload can be deserialized and treated as an object, pulling specific pieces of data out of the payload as necessary. A single .proto file is written to define any number of different payloads to send, which is compiled by the protocol buffer compiler to produce libraries to be used for serialization. This provides an easy way to create payloads with lots of information if necessary, which can be created and understood seamlessly by both the source device and the destination device.

In addition, a modified transmission protocol [7] should be used to define the order and confirmation of transmission. If the link between two devices is thought of as an open stream of communication, it is wise to set up a series of handshakes and data packets to ensure correct communication. For example, one may want to send the header first to alert the destination device of an incoming payload, what information it will contain, and how long the stream of data is. Once the destination device has processed this header and is ready to receive information from the source, such as a sensor data point, it will send a handshake, or similar confirmation. Since the destination device already knows the length of the incoming payload, there is no need to analyze it as it is coming in to catch a delimiter and end reception of a certain transmission.

On the other hand, delimiters are useful when sending data that will always be in the same format every time, but of varying length. This removes the need for certain information to be sent ahead of time, such as payload type and length, and any other overhead the payload may contain. This could reduce the size of the transmission, making for a faster exchange.

References

[1] Rodney A. Brooks. A robust layered control system for a mobile robot. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1985.

[2] Anca Discant, Alexandrina Rogozan, Corneliu Rusu, and Abdelaziz Bensrhair. Sensors for obstacle detection - a survey. *Electronics Technology*, pages 100–105, 2007.

[3] Calin Dragos. Types of sensors for target detection and tracking. *Intro Robotics*, 2013.

[4] Google. Protocol buffers, 2014.

[5] R. A. Hogle and P. P. Bonissone. A fuzzy algorithm for path selection in autonomous vehicle navigation. *The 23rd IEEE Conference on Decision and Control*, pages 898–900, 1984.

[6] Buniyamin N., W. A. J. Wan Ngah, N. Sariff, and Z. Mohamad. A simple local path planning algorithm for autonomous mobile robots. *International Journal of Systems Applications*, 2011.

[7] Inc. Network Sorcery. Tcp, transmission control protocol, 2012.

[8] W. Grey Walter. Imitation of life. *Scientific American*, 1950.