Background Research

Team 13: Thomas Lazor, Josh Hernandez, Tom Oestreich, Ben Johnson

Throughout the project there are countless design choices to be made which could drastically affect the difficulty of achieving the final product. Such choices could range from infrared sensors to adaptive movement algorithms, and for each decision it is helpful to have an idea of specific strengths and weaknesses. The purpose of this document is to provide a summary of those abilities for each choice as well as look at some suitable models.

In order to identify specific objects' locations, an RGB camera is most appropriate. One such option is the PIXY CMUcam5 which runs out to about $70. The PIXY differentiates objects, and then supplies users with specific coordinates by locating a specified color on screen and mapping it to a sort of grid. While this works well as a sort of system monitor, there is small 3cm error in some cases which would prevent trustworthy short range data. On the other hand, by increasing the size of the object it is possible to decrease the error and increase the effective range of the device. While accuracy usually depends on object size, it should be noted that poor lighting has the ability to render the camera useless. Still, given proper implementation this camera serves as a great locational tool.

For cases where a simple distance is needed there are two main options. The first is an ultrasonic sensor such as the Parallax PING (about $30). This specific device has a range of 2cm-3m, but data of the two extremes becomes unreliable. The risk here is that viewing objects at angles or in noisy areas may potentially skew results greatly. On the other hand, there is the Sharp series of IR sensors which runs around $20, and is not susceptible to ambient noise. Sunlight does, however, have a potential influence though. Still the device is more than adequate for indoor use. Its range is only 4cm-30cm, but this should work for our small scale purposes. It should be noted that both devices can be driven and measured on a single pin.

Finally, there is basic proximity sensing for simply indicating when objects are nearby. In this scenario a simple IR LED combined with receivers, such as the TSOP17 from sparkfun ($1 combined), would perform well enough. The range on this is only about 5cm, but that may be more than enough for jobs like preventing collisions. The downside here is that there is a large potential for ambient light interference, but this can be overcome if done correctly. Similarly, to the simple IR LED, an IR LED strip can be used to indicate when objects are nearby. An example of one of these strips is the SMD5050. Though significantly more expensive ($17), it doubles the detecting distance to 10cm at a 120-degree beam angle for all 10 LEDs. This option would provide a larger cushion to operate with while preventing collisions to maintain smooth maneuvers.

In order to have a fully functional autonomous vehicle it is likely that multiple sensors must coexist, each providing a specific function to contribute to the entire machine. Depending on the function, some sensors may be more appropriate to fulfill the necessary job than others. For example, a rover focused on driving between two walls may find the RGB camera to be more efficient, as opposed to the IR sensor. Yet, one following a line in order to determine direction would more often than not use the IR LED strip. As discussed above, there are many different uses for each sensor and in order to find out which sensor fits best for each job, one must first understand each sensors strengths and limitations.

As discussed in class, writing code that accommodates for error check is an important aspect of our project. Instead of designing one ourselves, we will be implementing a respected algorithm known as proportional integral derivative controller. According to research, this algorithm is used often in the industry for real world applications. Generally, a controller reads in values from the sensors and compares them to a reference value. The delta, or margin of error, is used to calculate a signal and convert it into a software command. These newly generated commands can be used to update our rover's speed and position. The error is usually mathematically manipulated in three ways, two of which are integration and derivation, then summed to generate a signal known as a gain. In some instances, a PID controller can be modified to only include the proportional and derivative gains to create a PD controller. Simplifying the controller allows for easier debugging and presentation. There are other variations as well. This flexibility grants our team the options to fine tune our controller as we deem necessary to complete our project.

Simultaneous Localization and Mapping (SLAM) is a common problem in robotics where a map of the environment is constructed while keeping track of the agent's location in that environment. One algorithm designed to solve SLAM is Monte Carlo localization (MCL). MCL requires a complete map of the environment beforehand, but then uses sensor data to compute probabilistic estimates of the agent's position. By acquiring more sensor data, MCL can increase the confidence in its estimate. Having to know the environment beforehand is a limitation of MCL, but it may be useable in some situations with the rovers. Another algorithm sometimes used to solve SLAM is linear quadratic estimation (or Kalman filtering). Kalman filtering assumes that sensor data will contain some amount of error and tries to combat this by computing probability distributions based on the accumulated sensor data. It has only two steps: predict the current sensor data and then update those predictions based on the next sensor data that has been collected. Kalman filters are sometimes used to get the best estimate of the sensor readings for PID controllers (mentioned above).

Some previous teams had rover environments that included a maze. Two maze solving algorithms that are suitable for rovers like ours are the Pledge algorithm and Tremaux's algorithm. The Pledge algorithm solves the maze using the Right hand rule (the agent keeps a wall to their immediate right the whole time) except when it encounters a disjoint obstacle. The Pledge algorithm detects a disjoint wall when it faces the starting direction and sum of the turns made since the start are zero. At this point, the agent then takes its "hand" off the wall and moves in a predetermined direction and continues with the right hand rule. Tremaux's algorithm draws a line to mark the agent's path through the maze. When the agent is faced with an intersection that it has not visited, it chooses a path randomly. If the agent has been there only once before, the agent turns around. If the intersection has been visited multiple times, the agent picks the least traveled turn and proceeds. Eventually, the agent will find the goal and the path with has been traveled once is the most direct path back to the start. Tremaux's algorithm works for almost all mazes.

Although there are still a variety of other decisions to be made along the way, the information above should provide enough technical background to create a solid foundation. As a specific project is chosen, and the team gains more experience issues such as message formatting will become more clear. Likewise, no immediate decisions are needed for the listed devices, but the knowledge of them should help better mold the idea of our intended project.

References

Brian Douglas. "PID Control – A brief introduction" *YouTube*. YouTube, 13 Dec 2012. Web. 06 Sept. 2016.

Burgin, Colin, Alex Makar, Jean-Phillipe Ouellet, and Conor Patrick. "ECE 4534 - Embedded - Team 13." *YouTube*. YouTube, 10 Dec. 2015. Web. 05 Sept. 2016.

CMUcam. "CMUcam5 Pixy." *Introduction and Background*. CMUcam, n.d. Web. 05 Sept. 2016.

"Kalman Filter." *Wikipedia*. Wikimedia Foundation, n.d. Web. 07 Sept. 2016.

Lao, Anna, Pooja Malhotra, Will Cowen, and John Kalin. "ECE 4534: Team 3 - Spring 2016." *YouTube*. YouTube, 05 May 2016. Web. 05 Sept. 2016.

Littley, Michael, Thomas Sickert, Daulet Talapkaliyev, and Renee Spangler. "ECE4534 - Spring 2016 Team 17." *YouTube*. YouTube, 01 May 2016. Web. 05 Sept. 2016.

Massa, Donald P. "Choosing an Ultrasonic Sensor for Proximity or Distance Measurement Part 1: Acoustic Considerations." *Sensors*. Sensor Mag, 1 Feb. 1999. Web. 05 Sept. 2016.

"Maze Solving Algorithm." *Wikipedia*. Wikimedia Foundation, n.d. Web. 07 Sept. 2016.

"Monte Carlo Localization." *Wikipedia*. Wikimedia Foundation, n.d. Web. 07 Sept. 2016.

National Instruments. "*PID Theory Explained*". 29 Mar 2011. Web. 06 Sept. 2016

"PING))) Ultrasonic Distance Sensor." *Parallax Inc*. Parallax Inc., n.d. Web. 05 Sept. 2016.

"Simultaneous Localization and Mapping." *Wikipedia*. Wikimedia Foundation, n.d. Web. 07 Sept. 2016.

Society of Robots, Eric,. "Infrared vs. Ultrasonic - What You Should Know."*Infrared vs. Ultrasonic - What You Should Know*. Society of Robots, n.d. Web. 05 Sept. 2016.

Zulu, Andrew, and Samuel John. "A Review of Control Algorithms for Autonomous Quadrotors." *OJAppS Open Journal of Applied Sciences* 04.14 (2014): 547-56. Web.