Group 44 - ARC

Tech Review

Senior Capstone Project

Oregon State University

Fall 2016

Tao Chen, Cierra Shawe, Daniel Stoyer

Abstract

Contents

1 Vision System Options - Cierra

For autonomous operation, vision systems are critical. The three main options include stereoscopic cameras, Infrared (IR) based systems such as Microsoft's Kinect, and Light Detection And Ranging (LiDAR) vision systems. With the exception of some forms of LiDar, all of these methods require what is called a disparity map, which creates a 3D image of the surface, that can be used for telling which objects in an image are closest or farthest away.

1.1 Stereo Vision

Stereo-vision uses two different cameras to create disparity maps in order to create a sense of depth. This is similar to how our eyes work. The biggest benefit to a stereoscopic camera system is the ability to detect objects outdoors, as the cameras are able to function with vast amounts of ultraviolet (UV) light. IR LEDs can also be used in order to illuminate an area at night, also allowing for night-time navigation. Another pro to stereoscopic vision, is the cost is relatively low, as cameras can be obtained for under $10 a piece. One of the challenges of stereo vision is the computational power required. Another is clarity of the disparity map without post processing of images, which makes real-time operation more difficult. [**?**] OpenCV [**?**] contains many examples of how to configure and process stereoscopic images, and is one of the largest vision resources. It can be used to synchronize and create the the disparity map, which can then be used by another part of our system for decision making.

1.2 IR Camera's such as Kinect and RealSense

Using an infrared point map, these cameras are able to tell the disparity between the points, which helps in creating disparity maps. The most popular example of an IR camera system, is the Microsoft Kinect. A big advantage to IR camera's, is the ability to function in low and non-natural lighting conditions, due to using the infrared spectrum, rather than only using the visible spectrum. The biggest problem with IR cameras, the functionality is greatly reduced outdoors, due to massive amount of infrared waves from the sun. IR cameras don't meet our requirement of being able to use the vision system reliably outdoors.

1.3 Lidar

LiDAR works by using laser pulses to detect range. By detecting different pulse signatures, it is able to take very precise distance measurements. LiDAR is a great way to form point clouds, however, the cost makes the product unreasonable for most people. A basic SICK LiDAR unite costs around $2,000 USD for a unit with 10m accuracy. citesick https://www.sick.com/us/en/product-portfolio/detection-and-ranging-solutions/2d-laser-scanners/tim3xx/c/g205751 A 2D RPLiDAR module is a lower cost alternative, at $449, however, the manufacture says that it will not preform well in direct sunlight, due to using IR lasers. http://www.robotshop.com/en/rplidar-a2-360-laser-scanner.html The RPLiDAR also only has accurate measurements up to 6 meters.

A third option for LiDAR, has yet to come to market. The Sweep LiDAR unit by Scanse will be released in January

of 2017. The Sweep unit claims to be a LiDAR unit available for all. Costing $349, the unit has the ability to scan 360 degrees, create points up to 40m away, and is able to function in ”noisy” environments, such as outdoors.
The Sweep’s distance capabilities and ability to be used almost any lighting condition, including outdoors, would make this a great candidate for our project. The caveat to the Sweep, is it will not be released until January.

The LiDAR systems that would be available to our group, would not meet the requirement of being able to function consistently in an outdoor environment, or do not fall under the category of commodity hardware.

1.4 Our choice

Due to the need to be able to navigate in outdoor environments, our team will start out attempting to use stereoscopic imaging as our primary vision system. We will do this using the OpenCV library to analyze the images, and create the disparity map that can be used for other purposes. If we have the computational power to post-process disparity maps in real time, we will attempt to do so.
If time allows, given we are able to obtain a unit, our team could investigate the use of the Sweep LiDAR system for navigation in place of stereoscopic imaging.

2 Sensors - Cierra

2.1 GPS - External

2.2 GPS - PXFmini

2.3 Internal Measurement Unit (IMU) - External

2.4 IMU - PXFmini

3 System Control and Synchronization - Cierra

IDK WHAT TO PUT IN THIS SYSTEM

3.1 Intel NUC

3.2 Intel UPboard

3.3 Raspberry Pi 3

4 Image Analysis Software - Dan

Image analysis, for the ARC project, is the processing of visual data received from cameras into deterministic information, such as pathfinding, or spacial awareness. This is the primary means for our autonomous vehicle to assess its surroundings and find its way to a given waypoint while avoiding obstacles. We require software that is freely available for use (via fairly liberal open source licensing), known to be correct (works well) with little modification needed, and has relatively easy to use API libraries.

4.1 DroneKit-Python

DroneKit-Python (http://python.dronekit.io/) is part of the DroneKit ecosystem (dronekit.io)

more to follow...

4.2 ArduPilot

ArduPilot (http://ardupilot.org/rover/index.html)

more to follow...

4.3 Image analysis choice

5 Telemetry Radio Communication - Dan

Telemetry is the transmition of measurement data (velocity, angle, rotation, etc.) by radio to some other place. This data is important for the human user to know the current state of the vehicle. This is especially important for autonomous operation, as the vehicle may not be operating within line of sight. Telemetry transmission is well-established, so we will not be comparing vastly different transmission technologies, such as long range (MHz radio frequencies) versus short-range (bluetooth) where the advantages of ranges of 2-15+ kilometers obviously outweigh ranges of 20-100 meters.

The main criteria for consideration are:

- Cost

  One of our main goals with ARC is to keep the costs low.
- Power consumption

  We have limited power available, therefore we need power consumption to be low.
- Ease of use

  The radio needs to be easily integrated into the autopilot system. This means it needs to have a developed API with little no modification required.
- Form factor

  The size and weight needs to be small and light. If it is too bulky, we might not have space on the vehicle. If it is too heavy, more power will be required to operate the drive system and will drain the battery faster.

5.1 3DR 915 MHz Transceiver

((https://store.3dr.com/products/915-mhz-telemetry-radio)

The 3DR 915 MHz telemetry radio has a cost of $39.99 USD for two radios. It is powered by the autopilot telemetry port (+5v) which means that has low power consumption. This radio transceiver has open source firmware, a robust API, and is fully compatible with PX4 Pro, DroneKit, and ArduPilot, using the MAVLink protocol, which will allow us to implement telemetry transmission with little to no modification of the API, should we use one of those autopilot systems. The form factor has dimensions of 25.5 x 53 x 11 mm (including case, but not antenna) at 11.5 grams (without antenna). The range of this transceiver is from 300 meters to several kilometers, depending on the antenna arrangement.

Biggest advantages: inexpensive, small form factor, low power consumption.

Biggest disavantages: range out of the box could be as low as 300 meters.

5.2 RFD900 Radio Modem

(http://ardupilot.org/copter/docs/common-rfd900.html)(http://store.jdrones.com/jD_RD900Plus_Telemetry_Bundle_p/rf900set02

The RFD900 Radio Modem has a cost of $259.99 USD for two radios. It requires seperate +5v power for operation which means that it has high power consumption. This radio has open source firmware, a robust API, and is fully compatible with PX4 Pro, DroneKit, and ArduPilot, using the MAVLink protocol, which will allow us to implement telemetry transmission with little to no modification of the API, should we use one of those autopilot systems.

The form factor has dimensions of 70 x 40 x 23mm (including case, but not antenna) at 14.5 grams (without antenna). The range of this transceiver is 25+ kilometers.

Biggest advantages: ultra long range.

Biggest disadvantages: expensive, large size.

5.3 Openpilot OPLink Mini Ground and Air Station 433 MHz

(https://hobbyking.com/en_us/openpilot-oplink-mini-ground-station-433-mhz.html) (http://www.banggood.com/Openpilot-OPLINK-Mini-Radio-Telemetry-AIR-And-Ground-For-MINI-CC3D-Revolution-p-1018904.html)

The OPLink Mini Ground Station has a cost of $26.59 USD for two radios. It requires input voltage of +5v and can be powered off the autopilot telemetry port which means that it has low power consumption. This radio has open source firmware but is only compatible with the LibrePilot RC control system. The form factor has dimensions of 38 x 23 x mm (including case, but not antenna) at 4 grams (without antenna). The range of this radio is not known, but based on the power requirements and frequency it likely has less range than the 3DR 915 MHz radio.

Biggest advantages: smallest form factor (only 4 grams), lowest cost ($26.59 USD)

Biggest disavantages: Only works with the LibrePilot control system.

5.4 Telemetry radio choice

The 3DR 915 MHz Transceiver is our selection for the telemetry radio. While the OPLink Mini Ground Station was significantly smaller, lighter, and cheaper than the other two, its implementation being tied solely to LibrePilot was a deal breaker (more information on LibrePilot can be found in the User Interface evaluation). The RF900 Radio Modem would have been a good choice, it has fantastic range and all the API options we were looking for. But it had a significantly larger form factor, required a separate power supply, and was quite expensive at $259.99. Put together, these facts eliminated the RF900 as a viable option. The 3DR 915 MHz Transceiver is a good balance of cost, performance, and size. The cost of $39.99 for two radios, the ability to power the autopilot off the telemetry port, and the portability of its APIs and their ease of use, puts the 3DR 915 MHz at the top of our list and the clear choice for the telemetry radio going forward.

6 User Interface - Dan

6.1 QGroundControl

(http://qgroundcontrol.com/downloads/) more to follow...

6.2 DroneKit-Android

(http://android.dronekit.io/) more to follow...

6.3 LibrePilot

(https://www.librepilot.org/site/index.html) more to follow...

6.4 User interface choice

QgroundControl, for many very good reasons (no, this isn’t the actual reasoning). more to follow...

7 Control System - Tao

A good control system can improve the accuracy of the estimated current location of the vehicle. If we choose unreliable control scheme and motion model, it will produce a lot of overhead for our main computer and the results won’t be as precise. Power consumption may increase due to the excessive amount of calculation done by the computer. In conclusion, motor control, servo control, and motion model are methods to minimax the computation the main computer has to undergo.

7.1 Motor Control

There are two types of motor control schemes that are practical for our project. I would not say that they are technologies. They are just two ways to decide how the vehicle should move forward and backward:

1) Time Critical: This motor control scheme tells the motor to spin forward/backward for a certain amount of time at a certain speed.
2) Distance Critical: This motor control scheme tells the vehicle to go forward/backward for a certain distance.

When an iteration is finished under time critical, the system will decide whether to keep the speed for another time period or slow down or accelerate. In order to reduce jerkiness, the rate of acceleration and deceleration will be low.

When an iteration is finished under distance critical, the system will decide what to in the next cycle. It can either switch to time critical or move for another distance.

In the final implementation, we may switch between the two methods based on real-time conditions. When the vehicle is operating at high speed in an open environment, such as a parking lot, with GPS cooperating, time critical better suits my purpose. When operating indoor without a pre-defined map, in other words, the vehicle is exploring the environment, distance critical works better. When the system detects an approaching obstacle, whether it is operating indoor or outdoor, the system should always switch to distance critical.

7.2 Servo Control

Servos control the steering of the vehicle. Servo control is similar to motor control. There are also two scheme under which the system operates the servo:

1) Time critical: The system tells the servo to keep a certain angel for a certain amount of time.
2) Angel critical: The system tells the vehicle to steer left/right for a certain angel.

Time critical makes drifting possible, which it is one of the goals of this project. When the car is operating at high speed, time critical will be easier to harness because oversteering will happen, and angel critical will cause unexpected maneuver when oversteering happens.

Angel critical may do a better job in obstacle avoidance. The angle of turning is defined as the angel between the driving direction of vehicle before and after the turn.

Again, in the final implementation, both schemes will be implemented and the system will switch between them based on real-time situation.

7.3 Probabilistic Analysis for Motion

1) Bayes Filter
2) (Extended) Kalman Filter
3) Particle Filter

7.4 Motion Model

In this context, the motion model states the behavior of the vehicle under different combinations of speed and steering. A concrete and precise motion model is important to the control system. The control system operates the vehicle based on this pre-defined motion model. For example, if the system needs to turn the vehicle 90 degrees right, it will have to output a sequence of actions by applying the motion model to the status of the vehicle, such as speed and center of gravity.

Different vehicle will have different configurations of motion model. When our vehicle is fully loaded with hardware, it will also have a different configuration of motion model than when it is unloaded. Thus, we will have to conduct multiple experiments to create the motion model of our vehicle. Alternatively, ROS provides a very nice simulation environment, where we can have a script that describes the configuration of the vehicle and have the simulator run the vehicle.

Potentially, as the development progresses, we may find the motion model to be obsolete, because I have a feeling that we can get around without a pre-defined motion model. But for now, a large part of me still believes it is necessary.

8 PATH PLANNING - TAO

8.1 Global Path Planning

Global in the perspective of our vehicle will be an area about the same size of a basketball court. We will pre-define maps with multiple way points for the system to navigate itself through.

- Breath-first Search:
  This algorithm promises to output the optimal path for start to destination in terms of nodes visited. In reality, the cost of going from one node to another should be considered when planning, so this algorithm will not be considered.
- Depth-first Search:
  Similar to breath-first search, this algorithm guarantees to find the optimal path for start to destination in terms of nodes visited. In reality, the cost of going from one node to another should be considered when planning, so this algorithm will not be considered.
- Dijkstra's algorithm:
  This is algorithm and breath-first search are very much alike. This algorithm takes into account the costs between nodes and promises to find the shortest path.
- A* Heuristic search:
  This algorithm uses an extra variable, the heuristic, to improve the performance of the Dijkstra's algorithm.

In conclusion, we will be using the A* Heuristic search algorithm as our global path planning algorithm. It produces reliable results and is the fastest algorithm among all the above-mentioned algorithms. Our computer should handle the computation no problem.

8.2 Local Path Planing

local in the perspective of our vehicle will be an area about the same size of a classroom. The vehicle may have to build the map itself by exploring the space or be provided with a pre-defined map.

- Rapidly-Exploring Random Trees (RRTs):
  This algorithm guarantees to find a path from start to goal as the number of sample points goes to infinity.
- RRT*:
  This algorithm is an extension of RRTs. It promises to find the optimal path from start to goal as the number of sample points goes to infinity.

These two algorithms are both feasible given the map is small, because they require a large amount of data to be stored in memory. This is why we did not consider the two to be our global path planning algorithms.

Depending on the computer memory and time constraints, we will use any one of them that suits our purpose. If the system is asked to output a valid path as fast as possible, RRTs will be used. If we were to ask the system to output the shortest path it can generate given all the computational resources, RRT* will be used.

9 Other Algorithms - Tao

9.1 Obstacle Avoidance Algorithm

In the global and local path planning algorithms sections, the underlying assumption is that the world is static (the map does not change). However, our team wants the vehicle to respond properly when unexpected objects are blocking the planned path.

There are two ways to accomplish this. We can:

1) Diverge from the original path as little as possible and converge back to the pre-planned path as soon as possible.
2) Reconstruct an entirely new path.

While re-planning may be more intelligent overall, it is infeasible in a global setup due to the large amount of computation the system has to redo. Rarely the construction of a path will take less than noticeable period time. Our team's vision of the system is to enable the vehicle to drive itself as if a rational human is driving it. New plan construction that causes suspension of movements is unacceptable. Thus, we will go for the first option and will not consider reconstructing new paths.

9.2 Parallel Parking Algorithm

By going through a research project carried out by students at University of Minnesota, parallel parking requires the system to perform following tasks:

1) Parking Spot Detection
2) Parking
3) Exiting the spot

For simplicity, we don't need to perform parking spot detection if the system is given a map. The algorithm will output a trajectory to the control system on every iteration. The system will be using multiple sensors, such as IR/sonar sensors, to scan surroundings, adjusting the trajectory accordingly. The control system must make sure the vehicle follows the trajectory precisely.