Otto

Submitted: 2019-05-22

Ian Adams - ija2@case.edu
William Nourse - wrn13@case.edu
Chude (Frank) Qian - cxq41@case.edu

I hereby certify that the design and development of the vehicle Otto, described in this report is significant and equivalent to what might be awarded credit in a senior design course.
This is prepared by the student team under my guidance.
Dr. Roger D. Quinn, Department of Mechanical and Aerospace Engineering - rdq@case.edu

Introduction

Otto is a mobile robotic platform for testing sensors, high level autonomy algorithms, low level control strategies, electrical hardware, and mechanical solutions. The platform serves as an educational tool for undergraduates and graduate students alike. The current iteration of the robot has been updated to conform with the requirements of the IGVC: autonomous-mode warning blinkers and line detection camera subsystems have been added to the existing hardware and software systems, as well as a one-touch start button for all autonomy.

William Nourse graduated with a B.S. in Electrical Engineering from Case Western Reserve University in 2018. He is now an Electrical Engineering Ph.D student, focusing on robotic control and neuromorphic computing. Aside from robotics research, William also has personal projects in designing electronic musical instruments.

Ian Adams is a Mechanical Eng. PhD student focusing on intelligent robotics. He graduated from CWRU in 2017 with an undergraduate dual major in Mechanical and Aerospace Engineering. Ian is a full time student at CWRU and has several personal projects in motion control, including several other pet robots, and his home built CNC machine.

Chude (Frank) Qian is an Electrical Engineering BS/MS student focusing on autonomous vehicles. He is expected to graduate in 2020 for his BS and 2021 for his MS. Frank has several autonomous vehicle projects involving a low cost ackermann-steering autonomous vehicle platform, full sensor fusion, RGBD image processing and deep learning for object recognition. He is also the lead person for CWRU's duckietown initiative.

One exceptional part in the design of Otto is the array of sensors. We have a pair of optical flow odometry sensors which can measure change in position in x and y. These sensors do not slip and have been very reliable in other robotic applications.

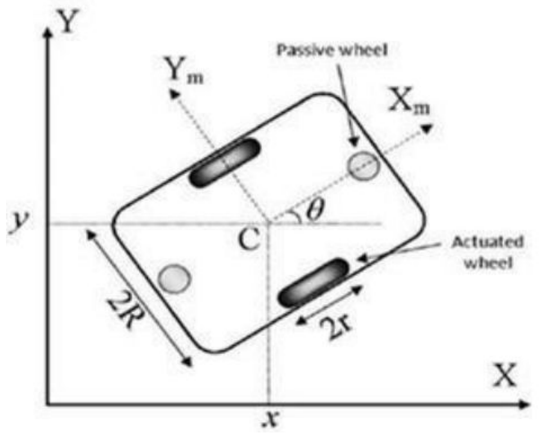
The optical detection of the painted white edges is an application of computer vision; by transforming the detected lines into obstacles in mapping, the robot can neatly recognize and respond to them. This is much more elegant than other line following algorithms we have employed in the past.

This year, we added an Intel RealSense D435 stereo-vision camera. Through some filtering algorithms, we have used these very cost effective sensors to act as both obstacle detectors, and in some cases, for visual odometry. By using this sensor for a wide range of functions, we hope to test its robustness for other applications.

Lastly, the data is processed with an Unscented Kalman filter, which allows us to get a robust estimate of position from noisy inputs. Each sensor contributes to the localization, and each sensor has a redundancy. This means no one sensor failing will drastically reduce the robot's effectiveness in operations. The Unscented Kalman filter is a change from last year's extended kalman filter. The UKF has the advantage of being simpler to design and calculate and more robust against nonlinear sensor readings such as those from the PX4flow and Rf2o odometry sources.

Mechanical Overview

Chassis Dimensions		
Overall Width	666	mm
Track Width	550	mm
Overall Length	990	mm
Wheelbase	690	mm
Height	720	mm
Drive Wheel Diameter	330	mm
Weight	68	Kg

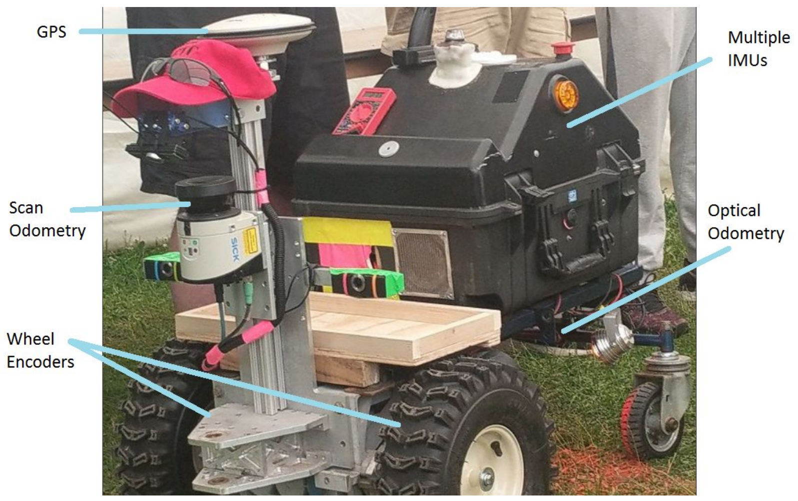


Otto is a uniaxial differential drive robot, meaning that its two collinear drive motors can operate independently to affect the linear and angular velocity of the system as a whole. The robot also has two passive rear casters. This geometry allows us to take advantage of the unicycle model of mobile robot. This model has well defined control characteristics, and easily defined kinematics.

Otto has a welded tube steel chassis which has a control box mounted to the top, and several modular attachments at the front. The robot has a cast aluminum battery bracket which also serves as the mounting points for the two motors. Cast aluminum or tubular steel mounting points on all four sides well as underneath the chassis allow flexible configurations of sensor instrumentation and task-specific implements.

The rear of the chassis has a rocker bar attached to the two passive casters, which allows the robot to travel smoothly over uneven terrain. In front of the motor mount, the chassis has an extruded aluminum Bosch rail for mounting the USB cameras, Novatel 702GG GPS antenna, and a Sick LMS111 270 degree lidar.

The control electronics are contained in a watertight Pelican Case 1560 with a custom vacuformed shroud. The E-stop and control switches are mounted on a laser-cut acrylic panel to the rear of the vacuform shroud. Weather proofing is achieved through silicone gasket material applied on all seams where components are not inherently sealed. With the exception of the cameras, the exterior sensors are weatherproofed enough to resist incidental water exposure. A plastic sealing wrap can be applied to the cameras if the weather turns south.



Electronics and Power Design

Otto is powered by two 12V Optima Yellowtop deep cycle AGM marine batteries, connected in series provide the vehicle with a 24V power system. This provides 38 amp-hours of power, allowing the robot to operate for roughly four hours between charging. Voltage regulation for many components is achieved through separate 12V and 5V switching regulators capable of 20A each, more than enough for our demands.

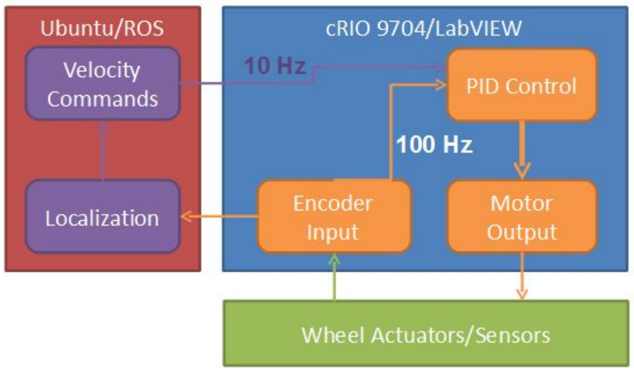
In the lowest level of our control systems, wheel velocity is controlled at 100 Hz. The onboard National Instruments cRIO FPGA counts encoder input from the motors, which runs at a 26,000:1 ticks per meter ratio (when we account for the diameter of the wheels). The FPGA then performs a PID control loop at 100 Hz to rotate the wheels at the angular rate described in the velocity command it receives over a serial link. The PID gains were hand-tuned through testing the closed loop system response to step inputs.

The next control layer provides velocity commands at 10 Hz. The 10 Hz velocity commands originate from the robot software guidance system which makes the robot converge on the given path. The higher level velocity commands are processed at 10 Hz in the onboard PC. Because the low level implements ten control cycles between each high level velocity command, we are confident the robot adequately executes the commanded velocity within the tolerances necessary to compete in the IGVC.

A 95-watt-max picoPSU powers a 35W Intel Core i3 dual-core hyperthreading CPU, on an ASUS mini-itx motherboard with 8 gigabytes of DDR3 SDRAM, and a 128 gigabyte solid-state disk. This is the main processor for the robot's various algorithms providing high-level control. The GNU/Linux operating system runs Willow Garage's Robot Operating System (ROS), a comprehensive middleware framework focused on providing soft-real-time scheduling and message passing for an unlimited number of software nodes. These nodes operate independently and in parallel, constantly publishing new output messages for other nodes to use on the next timestep. Some ROS nodes are written at the driver level, interacting directly with hardware, and others are complex algorithms working only with data provided by drivers.

The robot has an internal remote E-stop as well as a physical hard estop in the form of a large red button on the top of the vacuformed shell. The remote E-stop uses a 555 timer to determine the presence of a signal from the remote, and will trigger E-stop if this signal is not present, or does not change within 20 milliseconds of its previous change. The safety chain requires both the remote E-stop as well as the physical E-stop to be enabled in order to operate. Triggering one Estop or the other will disable power to the motors, forcing the robot to halt operation. In the event of an emergency, the robot can be disabled by triggering the remote estop, or physically pressing the big red button. After triggering an E-stop, the robot can then be interacted with confidence knowing that it will not suddenly leap into motion.

Software Strategy and Mapping Techniques



Our vehicle's software systems are designed to achieve the long term goal of robust, low-maintenance operation. The navigation system consists of two primary parts:

1. Sensors: Various sensors on board the robot. Otto implements both relative sensors (which measure local velocity or acceleration) and absolute sensors (which return measurements in a global frame and do not drift over time).

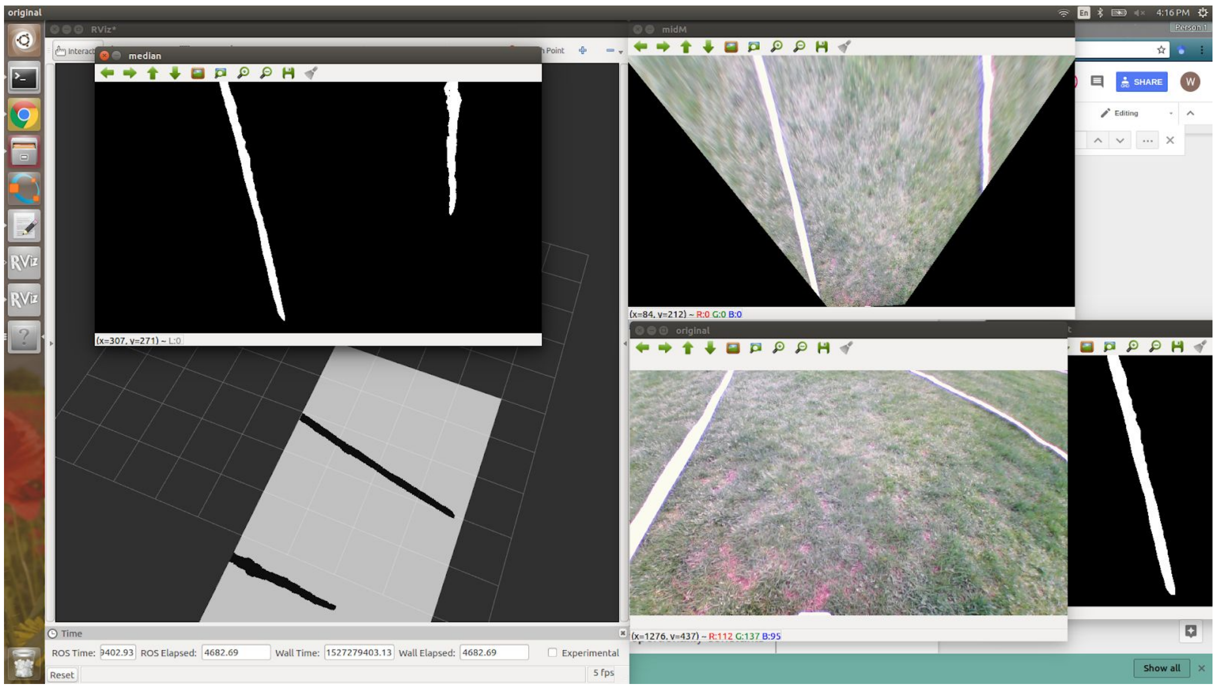
2. Algorithms: use UKF to combine data from multiple noisy sensors measuring different states of the robot.

The obstacle detection job is done by exploiting the competition map. All the points shown on this occupancy map would be treated as obstacles. The robot would recursively update its own position and figure out the single closest obstacle to its current position, allowing itself to simultaneously travel to the next waypoint without touching any obstacle.

Long term path planning currently happens by creating several manual global waypoints. These waypoints exist at corners of the field, and have comparatively large acceptance thresholds. These waypoints exist to guide the robot in the general direction we would like it to go. The robot is attracted to the waypoint corresponding to the next direction of travel, and repelled by its current position more so the longer it occupies the same area. This means the robot will constantly seek new more optimal paths, and be biased against areas where it has been trapped previously. This is how we plan to avoid dead ends.

The competition field map is represented as an occupancy grid. This field map consists of information coming from both the camera and the lidar. For the camera data, the first step is to do image processing. We applied perspective transformation, HSV thresholding and median filter on it and later converted it into an occupancy grid, to be transformed and added to a global map of the entire competition field.

For the lidar data taken from our sick LMS111, we simply convert it into an occupancy map with a polar to cartesian transformation, limited to a 20m radius around the robot. ROS transformation tree data is used to easily add both of these two small and frequently updated maps to the competition map.



Data provided by the obstacle and edge detection nodes in the competition map frame is used to generates a costmap of the area around it. By solving for the optimal direction in the cost map, the robot can create local waypoints far from obstacles, which assist it in getting closer to the global waypoints.

Our sick LMS111 and RealSense camera can also be used for odometry. Visual Odometry is becoming more and more popular for its versatility and expandability, and is usually based on feature tracking or dense image alignment. To perform visual odometry, we're using the algorithm RF2O, proposed by the University of Malaga at ICRA 2016, which can achieve robust Planar Odometry using only Lidar information. The RF2O algorithm generally assumes the environment is static and that scans are continuous. According to the literature, RF2O outperforms computationally the traditional Point to Line ICP (PL-ICP) method and Polar Scan Matching (PSM). One big issue with relying on lidar information for visual odometry is that a typical Sick Lidar scan only provides a single strip of information. Therefore we are also using the pointcloud data from the RealSense camera for RF2O as well.

Failure modes, Failure points and Resolutions

A primary failure mode expected is for the robot to slip in wet grass or float on top of thick springy grass, it is essential that we have some means of recognizing when the robot is no longer able to achieve traction, and taking action accordingly. The robot uses optical odometry to determine whether or not it has moved and by how much, and in this way it can account for errors involving loss of traction.

In the event of low battery/unexpected battery drain, the robot may have operational issues. The robot is capable of controlled operation for most of the batteries' voltage availability. In the event of critical low battery, the robot will continue to operate, but we will trigger a stop to ensure the robot does not pose a threat in operation. To prevent this outcome, the batteries are routinely charged and checked for health so they will be ready for competition run.

In the event of sensor failure, the robot has a number of redundant sensors, including multiple cameras, multiple IMUs and multiple types of encoders. The UKF automatically distributes weightings to the sensor input and if a sensor cuts out or produces garbage data, the UKF will tune it out to a great extent. The wheel encoders, and optical flow odometry modules perform similar operations (i.e. how far did we move and in what direction) and so the failure of one will not significantly impact the result behavior of the robot. There are multiple IMUs which can cover yaw data in the event one of them starts to drift significantly or one of them cuts out. Lastly there are multiple cameras for detection, and the failure of a camera, while significant, would not completely halt the robots ability to navigate.

Some sensors do not have duplicates such as the Novatel GPS antenna, lidar, and the downward facing camera. The failure of these sensors pose critical problems to the successful operation of the robot. The robot can operate for a short amount of time without good GPS data, as GPS position data is highly imprecise to begin with. The immediate operations of the robot will not be suspended in the event of such an error. However, we have occasionally in testing experienced a complete crash in GPS localization. In the event of such a crash, the robot would continue to operate on lidar, IMU, and camera navigation, but would likely require E-stop as errors begin to accumulate.

In the event of an unrecoverable error, we will trigger a remote E-stop, and then trigger the physical E-stop , and lastly service the device. This is a last option failure recovery method and we will only resort to it when the robot can no longer sustain safe operation.

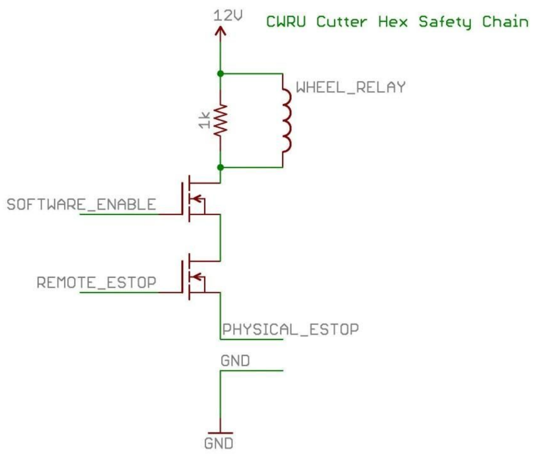
In order to prevent these failure modes, we have a regimen of safety checks. Starting by checking battery voltages, and ensuring the robot has enough power to sustain operation for the required course. Once this is determined to be acceptable, we will launch robot sensors to determine that they are in good working order and they are receiving good data. If sensors are active and the robot successfully initialized, we will have mitigated most of the risks.

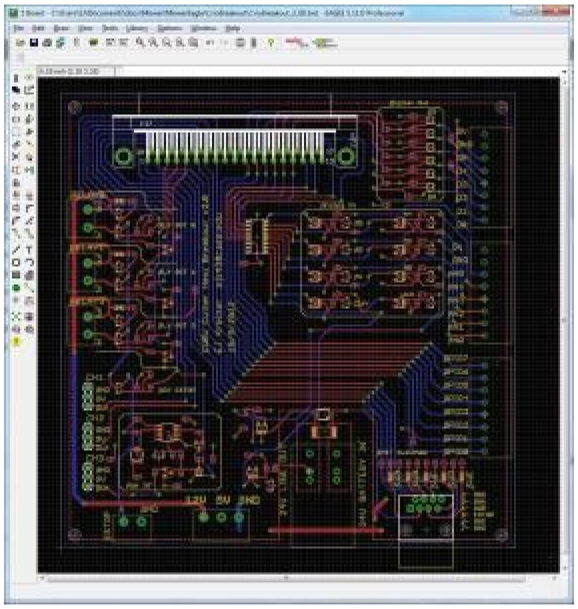
Vehicle Safety Design Concepts

The Otto safety system provides a triple-redundant safety check for the robot. The three components are:

1. Physical Emergency Stop: Red button mounted on the back of the robot.
2. Remote Emergency Stop: Built into the Futaba remote control with a range over 100 meters. If the remote control is out of range or is turned off, the remote e-stop triggers and the robot ceases operation.
3. Software Enable: The software includes a watchdog "heartbeat" measuring onboard processing. The robot is only enabled when the heartbeat is active.

The safety chain is implemented in hardware on a custom PCB. As shown in the circuit diagram of the wheel-relay safety chain, all three conditions (physical e-stop, remote e-stop, and software enable) must be true to provide power to the wheels. When any condition is false, the robot motors no longer receiver power.





Performance Testing to Date

UKF Testing:

In order to design an unscented kalman filter for Otto, we first needed to characterize the various sensors which would be incorporated. To do this we placed Otto in 2 controlled environments, a long hallway in Glennan building and a densely packed section of the Glennan parking lot. We ran Otto in a straight line at constant speed for 10 meters in the hallway, and around a square with a perimeter of 40 meters in the parking lot, with multiple trials of each action. From these experiments, we were able to record the various sensor outputs using rosbags and compare them with the known ground truth movements.

To characterize the performance of each sensor, we took the recorded data from each trial and isolated the sections which corresponded to known behavior, so either driving straight for 10 meters or turning clockwise by 90 degrees. Once these sections were isolated, we used a simple calculation to determine the predicted displacement per time recorded timestep,

$$x_{dt} = x_{total}/t_{total} * dt$$

where x_{dt} is the predicted displacement per recorded timestep, x_{total} and t_{total} are the known total distance and time period, and dt is the timestep (0.1 seconds on Otto). Knowing the predicted displacement per step, we can compare each timestep with the recorded displacements using the following relations, based on Otto's non-holonomic constraints:

$$x_{error}(t) = (x(t) - x(t-1)) * dt - x_{dt}$$

$$y_{error}(t) = 0 - y(t)$$

$$\theta_{error}(t) = 0 - \theta(t)\,.$$

Where x_{error}, y_{error}, θ_{error} are the errors of each quantity at a given timestep, and x(t), y(t), and Θ(t) are the sensor values at said timesteps. These relations are for Otto traveling in a straight line, similar equations can be constructed for rotational movement. For the RF2O data, all of these equations are used because it returns an estimated global position in all coordinates. Due to previous experience with the wheel encoders and IMUs onboard Otto, we concluded that their behavior was satisfactorily linear or gaussian and did not need to be characterized.

After the above sensor measurements were evaluated, code for an unscented kalman filter was developed. When setting up the filter, each input sensor contributes its measurements to the filter. Based on our characterizations, we adjusted each sensor for which kinds of data it was most predictable in providing. IMU’s contribute directly to rotational, and translational velocities, while the wheel encoders, GPS, PX4flow, and RF2O contribute to x and y coordinates, rather than differential values. We focused just on data directly output from the sensors, in the future we can also incorporate differential information so that all sensors can contribute data for most of the different robot states.

RealSense Testing:
We observed that the RF2O algorithm designed for lidar works very well for lidar based odometry. However, we did observe some issues with lidar only representing a thin strip of information. It cannot provide the whole picture. This raises issues of detection robustness as if there are obstacles at the edge of being detected. with roughness of the road, we might observe a drastically drifted odom or lost odom. One solution for this is to utilize a multi-line lidar. However, the issue with multi-line lidar is that there are really few options on the market and they are inherently expensive. Therefore we started to look for alternative solutions.

Intel RealSense provides a good RGBD image within 10 meters both indoor and outdoor. Effectively, we can utilize the RGBD image and extract information and consider them as some sort of lidar scan. We then pipe the “fake” lidar scan back to the RF2O algorithm and hence effectively, we can obtain several different lidar odometries using different features on the scene. Afterwards, we input these multiple odometries into the unscented kalman filter to get a more robust filtered visual odometry.

Initial Performance Assessments

While the unscented kalman filter is able to perform well in feature rich environments, more tuning and testing is necessary. With proper tuning we hope to overcome the performance issues seen in our testing, including drift in areas with few features and the covariance failure observed in some of our trials. Due to lots of inclement weather, our initial filter testing was mostly performed indoors on level ground. At the IGVC, Otto will have to navigate outside between GPS waypoints and on grassy terrain. Further tuning and testing in these environments will be necessary to guarantee satisfactory performance of the filter.

Although the performance of the proposed RealSense visual odometry is not as accurate as we expected, the loop closure test shows there are still serious drift in affect, the result gives us some confidence in this theory. We only experimented with one RealSense camera at the resolution of 1280x720. In theory two RealSense’s depth map can be fused together and provide a laser scan consisting 2560 scans and a wider field of view. At that point, the scan result will more look like an industrial multi-line lidar. However, we didn’t proceed with this option because fusing two depth map together is very resource intensive and defies our purpose of minimal computational power use.

In addition, all testing of this project is conducted indoors. Though indoor is generally considered a more contained environment, but due to the building's special reflective coating in the hallway as well as glass panel on the wall, it actually introduced more error and noise to the system than the outdoor tests.

Furthermore, the odometry data is mostly self-contained, without any global reference such as AMCL or SLAM. There is no other sensor information for sensor fusion such as IMU or wheel encoder. Although RF2O was designed for robust odometry, considering we are not providing it sufficient data, it is reasonable to have error on the result.

Finally, the hallway scenario is actually a tricky setting as it has few unique features required for robust odometry. Based on our expected environments, the theory of using RealSense's depth map for RF2O processing is workable.