PINGUINO INTELLIGENT ROBOTIC PLATFORM

Oakland University
Lucas Fuchs, Mackenzie Hill, Patrick Hines, Michael Lohrer, Dakota Perna, Tanner Snook, Daniel Vega
Ka C Cheok Ph.D. (cheok@oakland.edu)

ABSTRACT

This paper presents the Pinguino robotic platform developed to compete in IGVC for 2017. An octogonal aluminum frame houses a powerful powertrain capable of near zero-point turns and allows for off-road operation complete with a cantilever suspension system on the back caster. Inside the frame is a removable, custom-made, delrin plastic electronics box with a custom-made control board, motor controllers, power supply circuitry and LIDAR receiver. Although the LIDAR is attached to the front of the base, all other sensors including cameras and GPS antennas are mounted on a 71" sensor tower to increase field of view and location accuracy, respectively. To process sensor inputs, a-laptop is housed in the frame of the sensor tower to protect it from outdoor elements. The Robot Operating System (ROS) is used as a programming environment to take advantage of pre-existing, open-source packages such as path planning and image preprocessing. Innovations in the software system include multi-rate Kalman pose estimation, fusion of LIDAR with monocular cameras, line fitting with the RANSAC [2] algorithm, 3D map-based path planning, and the ability to create and load reusable maps of the environment.

INTRODUCTION

Oakland University is honored to enter Pinguino into the 25th annual Intelligent Ground Vehicle Competition (IGVC). Penguino's software systems include computer vision, pathfinding, LIDAR, and GPS which were all integrated using Robot Operating System (ROS). Penguino also features a custom built microcontroller designed specifically to meet IGVC requirements. All of these systems are housed on a two-wheeled platform utilizing differential drive steering.

Figure 1 - The Robotic Platform Pinguino

DESIGN PROCESS

A classic 'V-Model' design process was followed to develop Pinguino; see Figure 2. After defining the requirements of Pinguino, a design was formed using Solidworks and a detailed simulation environment was created to develop the navigation system; see Figure 3. After implementing the design and integrating the various components, a rigorous test cycle began where consistent failure points were identified and rectified through both minor adjustments and larger design changes as necessary.

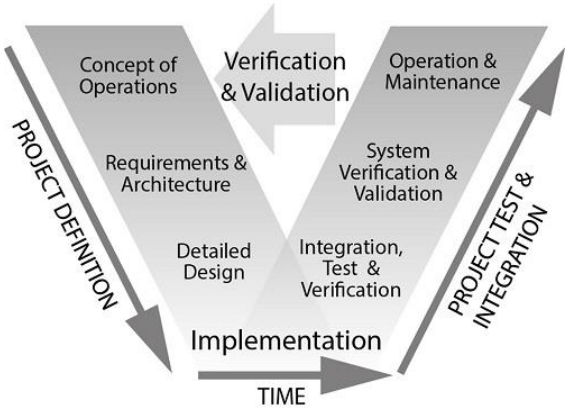

Figure 2 - The design process of the build

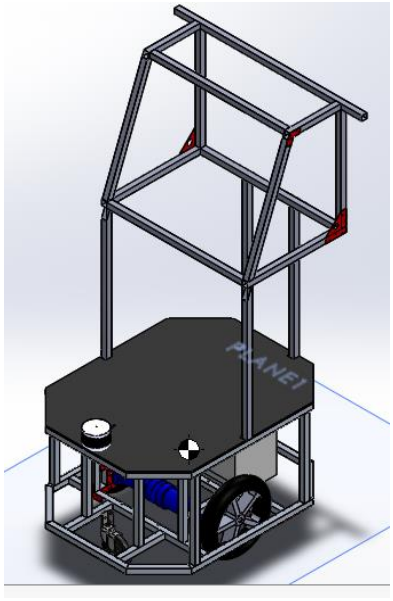

Figure 3 - CAD Rendering of Pinguino

MECHANICAL DESIGN

Before starting the mechanical design, it was necessary to account for each requirement of the competition including size and speed constraints, sensor positioning, and payload carrying capability. Although each of the requirements was adequately accounted for, size constraints and propulsion type were the main considerations while designing Pinguino. In order to satisfy the design constraints and ensure a sturdy and reliable platform, the team decided to improve on a previously attempted octagonal shape for the base. As in this previous design, Pinguino is equipped with differential drive to allow the robot to perform zero-point turns, but many improvements have been made pertaining to the structure of the base and sensor tower. Although it was unfortunate that our 2016 model lacked enough space in the base, was difficult to disassemble/reassemble, and lacked a suspension for off-road navigation, we have learned from the past and have kept these shortcomings in mind when designing this year's new and improved robot.

Base

The mechanical base consists of the same octagonal frame, but its size has been increased to 36" x 28" x21" tall, which allows enough room for the electronics box as well as the payload to be stored inside the base itself. The batteries are positioned so that they offset the weight of the motors towards the front. The material used for the frame of the base is square extruded aluminum with 1/16" wall thickness and is held together using rivets and custom steel brackets created with our waterjet machine. This combination leads to a durable base that remains lightweight while still able to withstand motor torques and shocks from the suspension with ease.

Due to extensive CAD modeling in Solidworks beforehand, Pinguino's base is able to house all necessary components within the base in a compact manner. The components contained within the base include two lithium phosphate batteries, battery charger, payload, electronics box, and motor assemblies. To obtain a center of mass low to the ground, the motor assemblies were placed at the bottom of the base horizontally along with the two batteries. The design's ample space gives the option to include additional components and rearrange current components allowing for continuous structural optimization. In order to keep the inside components protected from weather, a large polycarbonate sheet was custom cut to the dimensions of the octagonal base and secured on top. This polycarbonate material also serves as the mounting surface for the LIDAR allowing it to attach at the front of the base where it is most beneficial.

Sensor Tower

Pinguino also features a sensor tower where a large part of processing and data collection takes place. This tower includes cameras used to identify lines and colors, GPS antennas to determine accurate location, and also houses the laptop in a weather-safe laptop tray for easy access on-the-fly. The sensor tower sits on top of the base, stands at 71" tall, and increases to a width of 36" to allow more spacing between the GPS antennas to therefore increase accuracy. Cameras are mounted at the top front of the tower as high as possible without passing the IGVC-specified height to allow for a wide field of view in front of the robot. The E-stops are mounted in in the sides of the sensor tower four feet off the ground to allow easy access in case of an emergency. A great improvement in the sensor tower from last year is the ability to remove it entirely from the base of the robot. This feature allows Pinguino to better fit through doors and be more easily transportable. All sides of the sensor tower are protected from the elements by sheets of custom cut polycarbonate just as it is done in the base.

Figure 4 - The Base for Pinguino

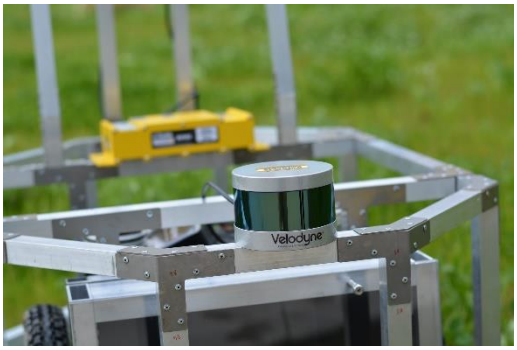

Figure 5 - Velodyne LIDAR

ELECTRONIC DESIGN

Pinguino's electrical layout is similar in structure to that of last year's entry, Schildkröte, albeit with an additional year of refinement. The majority of the electronic complexity of the robot lies in its custom-printed motor control board. This PCB contains the necessary regulators to power some of the external sensor as well as all of the processing power required to convert the usb commands sent by the computer, along with encoder information, into UART commands to control the H-bridge. As in previous years, our objective was to make the electronics modular for easy off-chassis integration, diagnostics, and testing. Pinguino's electronics, with the exception of some harnesses, are housed in a custom-made delrin plastic

box shown below in Figure 6. This allows the electronics of the robot to be removed entirely for ease of modification. The use of delrin plastic is an improvement from last year's aluminum electronics box since, by utilizing this material, the risk of shorting out components due to the housing has been eliminated.

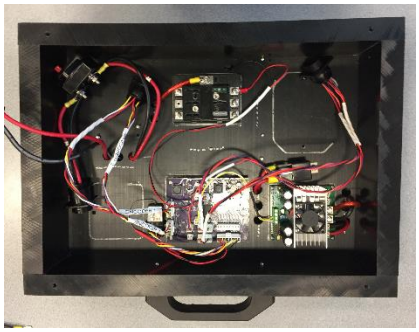

Figure 6 - Electronics Box

Figure 7 - Sabertooth 2x60 Dual Motor Driver

The motor controller utilized in Pinguino is a Sabertooth 2x60 dual-channel H-bridge shown in Figure 7. We have used this component in previous models and has proven to be robust and reliable and allows for high current draw for the motors as well as flexibility in software control. This H-bridge is capable of 60A continuous draw per channel which is much more than what Pinguino needs to optimally function. With a 6V to 30V input range, we can easily transition between using a 24V or 12V topology for our system. Sabertooth allows for packetized serial commands to be sent to control the speed and direction of our motors as well as a software E-Stop from our microcontroller. This will be the second year that we incorporate synchronous regenerative motor drivers, allowing the robot's batteries to be slightly recharged whenever the robot is decelerating.

Drive Control Board

Last year we introduced a custom drive control board built around the Programmable System-on-Chip (PSoC) 5LP from Cypress Semiconductor. While introducing this brand new hardware component to our system, we found many bugs with our hardware design, but have now made revisions to use a similar board as our drive controller. The chip contains 24 Universal Digital Blocks (UDBs) as well as an ARM processor, which allows it to make use of a synthesis of hardware and firmware.

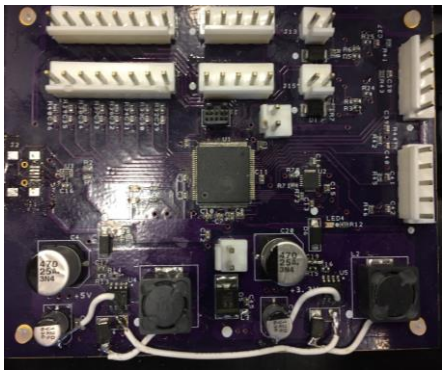

Figure 8 - Custom Controller Board

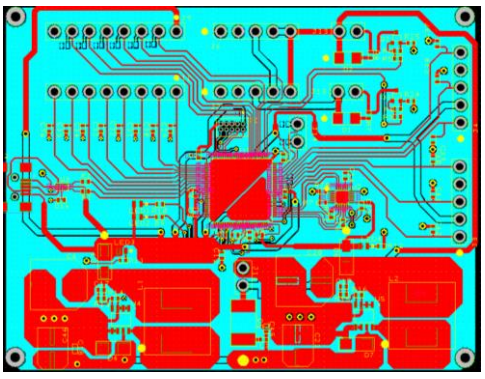

Figure 9 - PCB Layout of Controller Board

The use of a PSoC allows for the following features:

- Interface between motors and laptops using USB communication
- Allow Pinguino to be controlled with an RC radio transmitter
- Low level control of Sabertooth Motor Controller
- Read encoder signals from the motors
- Control the emergency shutoff relay for the motors
- Read motion data from an IMU
- Allow quick embedded development

This board is able to perform these function through the following methods:

- The PSoC reads the encoders using hardware, ensuring accurate readings
- Serial or PWM is used to control the Sabretooth H-bridge
- A mosfet is used to switch a 12V relay
- RC radio receiver is read using PSoC hardware, giving accurate readings

Figure 9 shows a block diagram of Pinguino's power distribution system. Pinguino's power comes from a 12V Lithium Ion battery. Due to the possibility of high amperage draw from the motors, both 8 gauge wire [3] and a 35A inline fuse are necessary. Using this size wire gauge helps to avoid heat build up in the connections around it during times of continuous high current draw. A circuit breaker is used to switch the power on and off to the power distribution system. The operator has the ability to power-up the embedded control board and sensors separate from the H-bridges for testing and safety purposes. The battery can be conveniently charged on-board, or quickly replaced by another set to achieve optimal runtime. Power and ground were distributed using two terminal rails; one for power and other was for ground. This allows an even distribution for both common ground and power among the elements of the electronic bay.

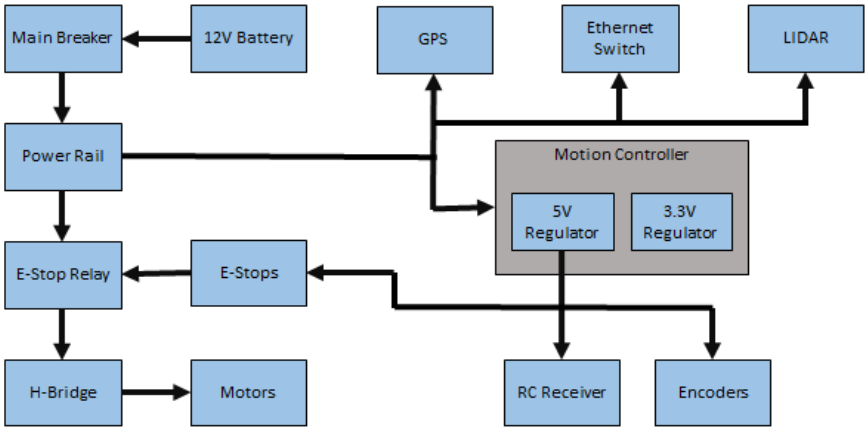

Figure 10 – Power Distribution System

Pinguino will be re-using the same GPS and Lidar used on previous year's robot as they allow for the most accurate obstacle detection and heading information given the hardware that we have available. The Trimble BX982 GNSS GPS Antenna and receiver shown below in Figure 10 as this allows for an increased true heading accuracy of less than a tenth of a degree based on the manufactured datasheets with a recommended 2 meter baseline. This is an improvement from previous IGVC competition robots that had a tolerance of 1.5m.

Figure 11 - Trimble GPS Antennas

Figure 12 - Trimble GPS Receiver

This year's model will consist of the following sensor array:

- Trimble BX982 GPS receiver ○ 20 Hz, with centimeter accuracy, shown in Figure 11.
- Velodyne VLP-16 LIDAR sensor ○ 100 meter range, 360° FOV and ±15° Vertical FOV, 20 Hz, shown in Figure 5.
- InvenSense MPU-6000 6-axis gyro and accelerometer, integrated shown in Figure 8.
- US Digital E3 Wheel 2500 CPR encoders
- DX6i wireless R/C aircraft joystick
- Embedded controller-based manual control and wireless E-stop

Safety Considerations

Many precautions were taken into account when designing Pinguino's emergency stop system. Both hardware and software based E-Stops were implemented. The E-Stop system disconnects power from the motors with a 12V 80A relay. This is accomplished by either triggering one of the side mounted, OMRON E-Stops or by turning on the software E-Stop remotely. A DX6i joystick is used for disabling the motor output wirelessly. The DX6i has a range of several hundred feet. To protect against a variety of failure conditions, the drive control system automatically turns off the motors if it fails to receive commands from the computer or joystick after 200 ms.

COMPUTING HARDWARE

Embedded Controller

The microcontroller module receives data from the encoders and imu, providing the lower level processing involved in directly controlling the motors based on computer inputs. Similarly, the embedded controller interfaces directly with the DX6I remote control module during manual control operation.

Laptop Computers

The computing platform we are using is a single Lenovo Thinkpad P50. The specifications of the laptop are a 2.8-3.7GHz Intel Xeon Quad Core Processor, 16GB of RAM, a 256GB SSD, and a NVIDIA Quadro M2000M GPU. These specifications were chosen due to the following requirements, a processor that allows for multiple concurrent threads, a plethora of RAM for complex calculations, a SSD to account for vibrations, and a mathematics driven graphics card for increased speed of complex calculations. For operation system we use Ubuntu 16.04 that runs ROS Kinetic. Finally, USB and Ethernet are used to communicate with various sensors and the micro controller.

ROS SOFTWARE PLATFORM AND SENSORS

Pinguino's software is implemented on middleware called Robot Operating System (ROS). ROS is an open-source development environment that runs on various Linux Distributions, such as Ubuntu. The community has created many resources for learning ROS and distributing packages for common robotic functionality. Popular packages include general-purpose mapping and planning packages and drivers for sensors, such as LIDAR, Cameras, and GPS Units. These packages allow for faster development and implementation of sophisticated navigation algorithms.

Efficient Node Communication

A ROS system consists of a network of individual software modules called "nodes". Each node is written in C++ or Python and runs independently of other nodes. Nodes can communicate between each other on "topics" via "messages". A topic is the name given to a group that a node can "subscribe" to. To send messages to the group, a node "publishes" a message to the topic and all subscribers receive that message. A major upside to this system is that it can be kept locally on the computer or messages can be sent over a network via ethernet. The modularity of this system allows easy development of reusable code and shortens implementation time of new code.

Debugging Capabilities

One of the most powerful features of ROS is its debugging capabilities. Messages can be recorded in "bag" files or displayed in real time on a GUI. Bag files can be played back with modifiers such as a different speed, between selected times, or only specific topics. Bag files also record a timestamp with every message so that playback can be recreated as if it were being played in real time. These bag files allow for testing software without needing physical access to sensors, allowing for development and testing to occur at any time. The debugging capabilities are especially helpful for testing mapping and vision algorithms to help identify and reproduce failure cases.

Another capability of ROS is a reconfiguration GUI that allows users to change parameters on the fly. This tool is invaluable since robotic vehicles often require precise adjustment of several parameters; therefore, being able to change these parameters while the algorithms are running proves to be very beneficial.

Simulation

Gazebo is an open source simulation environment with a convenient interface to ROS. Imitation IGVC courses were constructed to test the artificial intelligence of Pinguino. These courses contain models of objects encountered where Pinguino is used, i.e. the courses are grassy with lines, barrels, sawhorses, and fences to emulate Pinguino in the IGVC course as real as possible. Furthermore, simulation specific data required for Gazebo to create correct results from the simulation, such as but not solely including the moments of inertia and mass of specific parts on the robot and the physical body of the robot. This simulation is useful for development as it not only allows us to test how the robot acts but also collect data on what the sensors see inside the simulation.

Line Vision

Since IGVC is held outdoors, noisy images are a constant concern when attempting to use a lane detection algorithm. Stagnant noise such as dead grass and clover as well as exposure variances due to clouds each affect images in their own way and make it nearly impossible to design a robust algorithm to accurately detect the white lines. For instance, Figure 13 is a good example of why detecting these lines

with traditional image processing is complicated. Due to this complexity, an intelligent, self-learning algorithm was utilized to detect white lines with less noise sensitivity using Artificial Neural Networks (ANNs). ANNs have been used to train Pinguino to extract white lines and discard the majority of noise in the image.

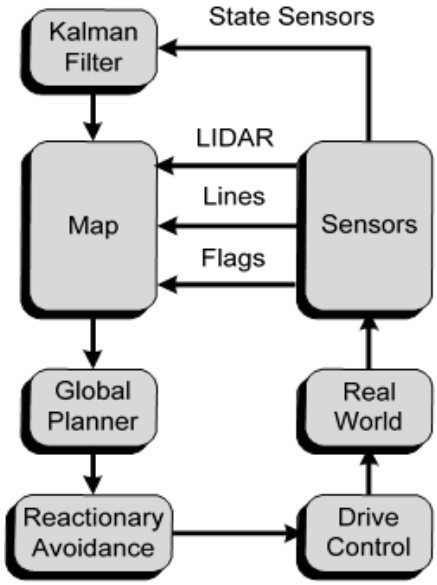

Figure 13 - Diagram of the Navigation System.

LIDAR System

Due to the way stereo vision operates, the mesh acquired is often imperfect and distorted. The Velodyne Puck on the other hand, produces a large very accurate mesh with few imperfections and minimal distortion. The Velodyne Puck uses a 16 laser array with +- 15 deg vertical viewing angle, and 360 deg horizontal. While the max range of the Puck is 100m, the easiest and most accurate distance to find barrels is within the first 10m. Due to the low speed requirements for the competition a larger radius is not needed as the objects will not be approaching faster than 5mph allowing for more than enough time to react.

Barrel Detection

After each scan from the LIDAR, the pointcloud is simplified and cropped to a 6m cube centered around the LIDAR. A Progressive Morphological Filter is then applied to the dataset. This separates the data into two distinct groups, one which is the ground and another which is classified as “non-ground” or objects. The objects point cloud is then filtered through a clustering algorithm to find each individual object. Once individual objects are separated, a cylinder fitter is used to check for the approximate size of a barrel. Once the barrel is identified and verified it is added to the costmap.

NAVIGATION SYSTEM

Kalman Filtering

The Kalman Filter provides the ability to predict the position and orientation of the robot, deriving it from the State Sensors’ input as shown in Figure 13. The different inputs are added together and updated with the fastest sensor to available accurate determination of position and orientation in between slow but precise GPS updates. The orientation is represented using quaternions, due to the discontinuity of Euler angles. For adding the measurement of the orientation data the quaternion gets normalized and transformed into an angle. The yaw angle can be represented by a 2D vector. The sensors providing data to the Kalman

Filter include the GPS, which provides orientation and position, the IMU, which provides acceleration and orientation, and the wheel encoders, which provide velocity.

Mapping

Information gets placed on the map using Pinguino's mapping algorithm. The map is represented in 3D as 5 planar layers. All of the estimated position and orientation data is time-stamped and uses the ground plane as a reference. Sensors can mark and clear space on the map. A sensor is clears space by projecting rays through the 3D space of the map starting at the sensor. As the rays move through spaces not occupied by objects the sensor clears those spaces on the map. Two instances of the mapping algorithm are used to do this, the local and global. The local map is a 15 meter square with a 5 cm resolution while the global map is 100 meters and has a 10 cm resolution. The global map is initialized beforehand if there is already data that can be used to create one.

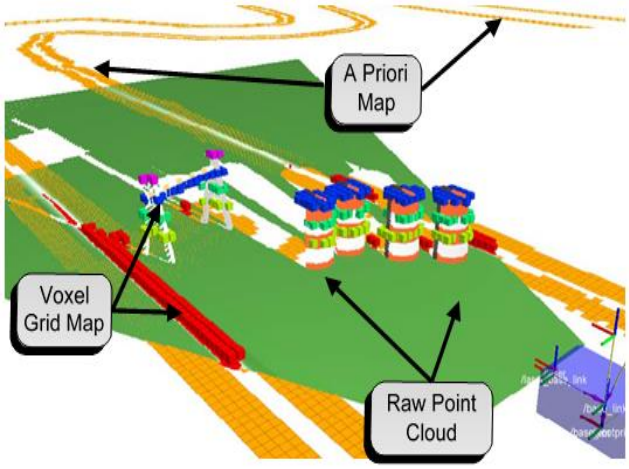

Figure 14 - Example of Mapping Procedure

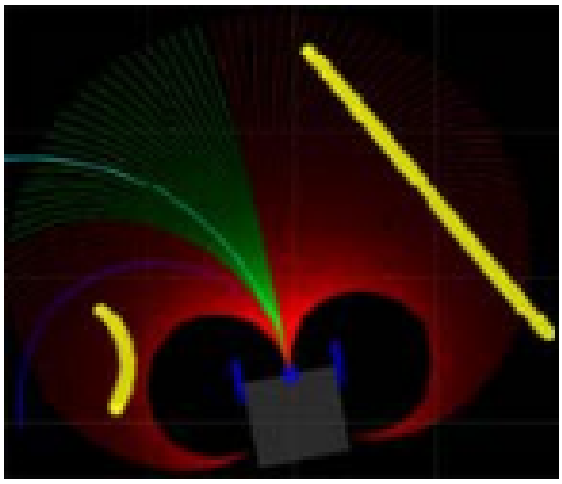

Figure 15 - Reactionary Avoidance

Global Path Planning

The global path planner uses a global map create a path to the goal with the least cost. Any occupied squares on the map are inflated to account for the robot's width, these squares are known as having lethal cost, and then they are further inflated using an exponential cost function to account discourage the pathplanner from getting to close to objects. The path planner is not allowed to plan through squares that have lethal cost however they are allowed to plan through the other inflated square if necessary. The global planner makes use of Dijkstra's algorithm, this algorithm guarantees that the planner will find the most optimal path to the end goal given the restrictions of the costmap.

Reactionary Avoidance

The last stage in the path planning is reactionary avoidance. This makes use of a local map which is a smaller version of global map that follows the robot and updates representing any new obstacles as the robot moves. If a new object is detected by the sensors the local path planner must update the global path to avoid the object. The local planner does this by projecting out different trajectories for the robot to travel. These trajectories all start from the center of the robot and extend out in all directions. The Reactionary

avoidance then finds what trajectory best fits the global plan without hitting the object. Some additional space is left for safety to ensure that the robot does not drive too close to the object.

PERFORMANCE ANALYSIS

Maximum Speed

Pinguino's motors spin at approximately 130 RPM at nominal load. Using this speed with 12.5 inch diameter wheels, the calculated maximum speed is 4.8 mph. This speed is within the rules of IGVC.

Ramp Climbing Ability

At nominal load, the drive motors provide 230 in-lbs of torque. Assuming a realistic vehicle weight of 115 lbs, this corresponds to a max slope of 19 degrees. However, experiments have shown that Pinguino can handle much steeper slopes, up to approximately 30 degrees, although the motors will perform outside of the nominal operating envelope.

Reaction Time

The computer systems are made to respond to sensor data at the rates they are provided. This is convenient, but also means that decisions can only be made at the slowest rate of sensor data, 20 Hz as shown in Table 1.

Battery Life

At peak draw for the two batteries, 360 W could be delivered at any given time. This would go through the combined battery charge completely in 30 minutes. However, this will likely not be the case, as the motors are not expected to require peak draw very often if at all. At the same time, a convenient battery charging circuit allows for the robot to charge from any standard AC wall plug without the need for detaching the batteries from the chassis. Thus, even if the robot can only sustain a 30 minute lifespan, the batteries would be easily recharged.

Obstacle Detection

The Velodyne LIDAR has a range of about 100 meters, and has 16 360-degree scan lines at angles between -15 and +15 degrees. The cameras are oriented to see 4 meters away from the vehicle.

GPS Accuracy

The Trimble BX982 GPS receiver is accurate to within 1 centimeter using the RTK technique. It also utilizes a second antenna to provide heading information. Position updates are sent at 20 Hz, but the Kalman filter algorithm fuses the GPS readings with the rest of the sensors to provide faster position updates.

VEHICLE EQUIPMENT COST

A breakdown of the cost of the components on Pinguino is shown in Table 2.

CONCLUSION

Pinguino has proven to be a rugged, efficient, reliable vehicle that performs well while driving on any kind of terrain. The new robot design shows promising results, and the Oakland University team has great confidence going into this year's competition.

Table 1. Sensor Data Rates

Sensor	Data Type	Frequency
Kalman Filter	Position and Orientation	200 Hz
Velodyne LIDAR	Obstacles	20 Hz
uEye Cameras	Lane Obstacles	20 Hz

Table 2. Cost Breakdown of Vehicle

Item	**Cost**
Trimble BX982	$5000
Thinkpad Laptop	$1500
Velodyne VLP-16 Lidar	$8000
uEye Camera x2	$917
Camera Lenses x2	$150
12V Battery x2	$420
Motors, Wheels, Speed Reducers	$1397
Frame Material	$650
Sabertooth Dual 60A Motor Driver	$190
NETGEAR Gigabit Router	$73
Total	$18297

ACKNOWLEDGMENTS

The Oakland Robotics Association would like to thank several people and organizations. The team first of all thanks their advisor, Dr. Ka C. Cheok, for his invaluable advice and guidance. Thank you also to the School of Engineering and Computer Science (SECS) and its dean, Dr. Louay Chamra, for generous funding and lab space to work in. Immense thanks to Peter Taylor, machinist in the SECS for his help and guidance in the fabrication of Pinguino. Additionally, without guidance from our past teammates we would not have the solid foundation to necessary to build this complex system. They are many, but we would like to especially thank Kevin Hallenbeck, Micho Radovnikovich Ph.D., Steve Grzebyk, and Lincoln Lorenz.

REFERENCES

1. IGVC Rules Committee, "IGVC Rules 2017," http://www.igvc.org/rules.htm (Accessed May 15, 2017).
2. Non-linear RANSAC method and its utilization. Radek Beneš, Martin Hasmanda, Kamil Říha. December 2011, Electrotechnics magazine ISSN 1213-1539, Vol. 2. 4.
3. Dimension Engineering, "Sabertooth 2x60 User's Guide," https://www.dimensionengineering.com/- datasheets/Sabertooth2x60.pdf (September 2011)