System Architecture and Implementation of a Hybrid Powered Omni-Directional Autonomous Vehicle

Kyu Hyong You, Owen Allen, Jonathon Jeske, Ryan Chilton, Shannon Ridgeway, and Carl Crane
Center for Intelligent Machines and Robotics
University of Florida, Gainesville, Florida
352-392-1932
kyuhyong@gmail.com

ABSTRACT
An omni-directional vehicle is described. The system architecture is detailed with consideration of the power system, drive system, and control system. The vehicle’s design is being implemented and current progress is discussed.

The power system consists of a modified off the shelf generator-IC engine pair coupled with lead acid batteries. The generator-IC engine pair has been repackaged and electric start added. The generator produces alternating current that is rectified and conditioned to supply a DC buss. This buss is backed up by lead acid batteries that have approximately 1 KW/hr of storage. The DC buss provides power to the drive motors, steering motors, cooling system, and control system.

The drive system consists of three drive wheels. The drive wheels are of a custom hub driven type and were fabricated for this application. They are propelled by a frameless brushless dc motor that drives the wheel rim via an epicyclic gear train. The drive motors are cooled by a closed coolant loop cooling system that includes a radiator mounted on the vehicle chassis. The drive wheels are mounted on a spindle that is actuated by a steering motor, allowing a minimum of 400 degrees of steering angle for each wheel. The steering motors are sourced from Animatics and include a dedicated controller and amplifier. The steering/drive train is mounted on a sprung swing arm to the vehicle’s chassis.

The control system is composed of a hardware control component and an Autonomy component. Two COTS Micro-ATX motherboards and an embedded microcontroller are utilized to implement the control system. The motherboards host a quad core and dual core processor and are used to run the higher level hardware control and the autonomy. The embedded micro-controller manages the drive motors and hardware system control. The autonomous control of the vehicle is implemented utilizing the JAUS 3.3 standard. The architecture is a derivative of past CIMAR work utilized in the DARPA Grand and Urban Challenges and is discussed.

Keywords
Omi-directional, autonomous ground vehicle

1. INTRODUCTION
The Center for Intelligent Machines and Robotics (CIMAR) at the University of Florida is developing an omni-directional ground vehicle. The vehicle will utilize CIMAR’s autonomy that has been developed over the course of the three DARPA Ground Vehicle Challenges. The size and speed capability of the vehicle are tailored to allow competition in the Association for Unmanned Vehicle System International’s (AUVSI’s) intelligent ground vehicle competition.

2. Background
Many conventional wheeled type vehicles have fixed drive wheels and change the direction of the vehicle by orienting the steering wheels. The fixed configuration of the drive wheel constraints the mobility of the vehicle to one as it typically follows Ackerman steering. The omni-directional vehicle platform presented in this paper has three steerable drive wheels allowing the vehicle to maneuver in any direction instantaneously. The steerable wheel mechanism used in this platform is considered as having non-holonomic mobility [7] as it requires a finite amount of time before the steering mechanism can reorient the wheel to the next projected curve. Some special wheel designs such as Mecanum wheels have been introduced [5] to tackle this problem by adding rollers on the wheels. However the rolling mechanism often increases power loss due to conflicts among other actuators [8] and are hard to build for bigger applications, thus we have chosen steerable wheel mechanism for the omni-directional mobility.

3. Platform
3.1 Drive
CIMAR undertook the design and development of a drive wheel for omni-directional use. Fulmer [2] successfully demonstrated the performance of an hub drive wheel for the omni-directional vehicle in his paper. The hub drive wheel developed at CIMAR consists of a brushless motor embedded in the hub of a wheel with a two stage epicyclic gear train. Active cooling of the motor is integral. Figure 1 shows the dynamometer test set-up for the drive wheel done by Fulmer. Figure 2 details the results of the dynamometer testing with seven forced cooling rates. The performance of the drive is consistently better than 280 in-lbs of torque. This converts to a propulsive drive force of approximately 46 lbs. The vehicle is required to traverse a 15 degree slope and accelerate to and maintain a speed of 5 mph. The drive is adequate for these requirements.
3.2 Power System
The power system architecture for the omni-directional vehicle is illustrated in figure 3. The system is a serial hybrid approach with a gasoline powered Honda generator for electrical power generation. Lead acid batteries serve as power storage. The primary DC bus is 48 volts. Secondary busses are sourced from the primary via DC-DC converters. A common ground is maintained for all DC busses. The generator can be replaced with wall power to facilitate static testing in the lab without the internal combustion engine running.

3.3 Computational Resources
The omni-directional vehicle autonomy is implemented on two COTS motherboards. One motherboard supports a quad core Intel Q9400 and the other a dual core Intel E8200. Both systems are powered by dc-dc atx power supplies. Inter system communication and development are facilitated with a gigabit router.

4. Sensors
The environment of the autonomous vehicle is explored using several types of sensors. A single LADAR is utilized for terrain classification and obstacle detection. Two cameras are used to find lines and obstacles. An inertial measurement unit, a GPS with RTK correction, and wheel odometry are utilized for localization.

4.1 Localization
Localization of the autonomous vehicle is critical for our implementation of the Smart Sensor concept and the reactive planner. The architecture requires that the sensor data be placed in a common reference frame for fusion. We use the global frame for this purpose. Several sensors are utilized to estimate the current position and orientation of the vehicle in the global frame. A Microstrain 3DM-G Inertial Measurement Unit (IMU), a Novatel Propack V3, and wheel odometry are used to estimate position and orientation.

4.2 Obstacle Detection
The CIMAR architecture implements reactive autonomy. This requires characterization of the environment local to the vehicle. Detection of objects and classification of drivable areas are needed.

4.2.1 LADAR
A Sick LMS-291S05 laser range finder is mounted to the front of the vehicle with an actuator to continuously adjust the pitch angle of the sensor. The purpose of using a ladar for mapping the vehicle environment is twofold: obstacle detection and terrain traversability estimation. Obstacle detection is possible by extracting objects from the surrounding ground plane point data while terrain traversability is evaluated by applying a heuristic method to the slope and curvature of the terrain. Figure 4 shows the configuration of the sensor.
The process of recreating the environment from the ladar data involves first transforming the polar data to the vehicle’s reference frame and then transforming these points to the global reference frame. The advantage of storing and processing the point data in the global reference frame is that base point data transformation is only required once whereas keeping the data in the vehicle reference frame would require transforming old data points as the vehicle moves and rotates. The data is stored as z-heights in a regular grid structure as opposed to an irregular network so the points must be snapped to the nearest grid point before being stored. Since the data is kept persistent over time, new data must be merged with this older data. A weighting factor based on age is used so that older data has less confidence than newer data. If two points are merged that are close (i.e agreement between old and new data), their weighting factor becomes higher. Figure 5 below illustrates the usage of addable and removable tiles that store the point data as well as the local region that will be processed shown in yellow.

A 30 m by 30 m region of the point data is selected and processed at each time step in order to evaluate traversability and obstacle boundaries. This region of the laser range data is processed and a traversability grid is formed. The traversability grid is a 121 by 121 element grid that is mapped to this region, yielding a 0.25 meter resolution. Each entry in the traversability grid is scalar value that represents that area’s traversability based on smoothness, slope, curvature, and elevation. This general value can be interpreted by the planning element based on the vehicle type so a region to be avoided by one vehicle type might be easily covered by a larger vehicle. Finally, the objects that have been extracted from the ground plane based on their height above the surrounding ground space is vectorized into a polygon and stored in the knowledge store as an obstacle.

4.2.2 Vision

The Line Finding Sensor is a vision based component which extracts line information from the local environment. For the OMNIGATOR, we use two mvBlueFox 120aC cameras which perform the roles of finding lines as well as obstacles. The extraction of lines is performed by using Hough transform. This is implemented using openCV libraries.

Detection of painted demarcations is accomplished by considering multiple environment cues which include intensity, color, shape and orientation. These points of interest are searched for and linear dominant elements are reported. The input image which is a three channel image is reconstructed into a single channel image and the Canny algorithm is applied to it in order to detect the dominant edges. The Canny algorithm computes first derivatives in x and y, which are then combined into four directional derivatives, where each of the maxima are assembled into edges.

The Probabilistic Hough Transform is applied to the binary image, which is a slight variant of the Standard Hough Transform. The Standard Hough Transform calculates the ‘rho’ and ‘theta’ value of all the lines present in the frame, where ‘rho’ and ‘theta’ are the distance and angle respectively, of the line from the origin (0, 0). In our case the starting position was the top right of the frame, which indicates a negative value of ‘rho’ meant the line was on the right half of the frame. The Progressive Probabilistic Hough Transform is a variation as it accumulates only a fraction of the points in the accumulator plane. This results in line segments with starting and ending points instead of an infinitely long line with the given slope. This was chosen over the Standard Hough transform as the first derivative of the set is more accurate than the ones derived by Standard Hough Transform.

For the line finding algorithm, we set an initial slope value and then perform a constant comparison with the previous slope to see that there is no major deviation from the previous slope, thus following the same line though it may have slight changes in slope as in the cases of curves. Once these points are extracted, the information is sent to the Knowledge Store. The points are packed in a particular order to indicate that one side of the line is traversable while the other is not.

Obstacle detection with the cameras has been reduced to a polygon recognition and reporting one. The obstacle in the path is found by detecting the contour. This is implemented using openCV libraries. The binary image with the edge pixels is classified into positive and negative regions. Depending on the intensity, the area is classified as either a contour or a hole. The contour is the approximated into an approximate polygon. This is done by taking the extreme ends of the contour and drawing a line between them. The polygon which is farthest from this line is taken as the third point and so on. The iteration stops when the distance between the point and the line is lesser than a specified value. Also while finding the polygons, noise may interfere with the data capture, so we assign another condition that the area of the contour detected must be greater than a particular value (to indicate that there is a polygon).

The points are found in a random order which is packed into a predefined clockwise order and sent to the Knowledge Store.
5. Control

5.1 Low Level Control
As an omni-directional mechanism, three hub drive BLDC motors are attached to a L-shaped bracket. Each wheel bracket is connected to the steering servo motor with a 48:1 gear head. Figure 6 shows the physical arrangement of the mechanisms.

A SmartMotor SM3440D from Animatics is used for the steering actuator. It is a complete servo system equipped with individual embedded motor controller, amplifier, and an encoder. Each SmartMotor is commanded through RS-232 serial communication.

Figure 6. Drive configuration

In order to control the vehicle platform, certain inputs have to be converted to a power to drive the actuators. In the hierarchy of the JAUS architecture, the primitive driver (PD) provides the method of interpreting plans to actuator command. PD requires low level or hardware level control system. Figure 7 shows the architecture designed to interface with actuation systems.

Figure 7. Control system layout

Figure 8 shows the simplified diagram of the PID control system developed for the omni-directional wheel motor.

\[ E = \text{GoalRPM} - \text{CurrentRPM} \]  

Where GoalRPM is desired velocity of the motor and CurrentRPM is the latest encoder count measured during a sampling time. The sampling time in this case is set to 8ms, a sampling rate of 125 Hz. With this feedback system, the discrete-time PID control law is shown as follows:

\[ \text{Output}_{\text{PID}} = K_p E + K_i \sum (E \cdot \Delta t) + K_d \frac{\Delta E}{\Delta t} \]  

Where \( K_p \), \( K_i \) and \( K_d \) are the controller gains of proportional, integral and derivative terms respectively.
Adjusting the gains used for the PID controller often becomes trial and error especially when trying to control motors with unknown characteristics. Since the nature of the omni-directional mobility requires synchronous motion, tuning of the PID motor controllers were focused on ensuring paralleled motion. The step responses of all three motors are shown in Figure 9. This result shows the performance of the newly implemented PID motor controller.

![Motor Control Response](image)

**Figure 9. Step response**

### 5.2 Autonomy

The autonomy deployed on this omni directional ground vehicle is derived from the CIMAR architecture that was developed over the course of the DARPA Ground Vehicle Challenges. The architecture implemented traversability grids in the second Grand Challenge[2], and multiple behavior modes were implemented in the Urban Challenge. The development of the architecture is ongoing. The current work focuses on improving situational awareness through utilization of a knowledge store.

The architecture is implemented in a JAUS 3.3 compliant manner. JAUS defines a component as a collection of software that performs a defined task. JAUS specifies a group of messages that allow components to communicate. The standard leaves room for custom messages and experimental components. Some of the architecture is experimental.

The Smart Sensor Concept developed for the Second Challenge takes raw data from the sensor level and converts it into a format that is fusible with other sensor data, whether the sensor is of the same type or not[2][6]. The traversability grid provided the means for this fusion to happen. Tested heuristics were utilized to fuse the data to allow planning based on multiple sensor input with varying time and space resolutions. The addition of a knowledge store is requiring modification of the Smart Sensor output. We have maintained the traversability grid and the appropriate representation of the reactive environment in which we plan in. We have the Smart Sensors extract terrain data and represent it as a traversability grid. Obstacle data is reduced to vector representation and stored in the Knowledge Store. This data is fused with the traversability data in a process call the Arbiter.

The fused result is then used in a receding horizon planner based on an A* search [4]. This component has previously been utilized for skid steered and Ackerman steered vehicles. The generational trajectories utilized by the A* search are derived from a vehicle kinematics model. The omni-directional vehicle’s kinematics are theoretically unrestrained in the plane (3 degrees of freedom). As mentioned in section 2, the steering and wheel velocities cannot be controlled instantaneously, so the vehicle dos have slightly non-holonomic behavior. A ray based kinematic model is utilized as the kinematic model for the omni-directional model at the reactive planning stage, and the low level control is scheduled to compensate for the non-holonomic behavior by retarding linear velocity in favor of correct heading.

The vehicle is expected to complete two primary behaviors: waypoint following and corridor following. In waypoint following, the vehicle is given a list of waypoints in Latitude and Longitude. The vehicle is then expected to navigate from point to point, avoiding obstacles. In corridor following, the vehicle is put in a corridor defined by lines on the terrain, and expected to follow the corridor avoiding obstacles until the corridor ends. Both of these navigational behaviors can be implemented by driving the optimization scheme of the receding horizon planner. A desired path is imposed on the traversability grid that the Arbiter produces, leading to the desired path being found by the planner provided no environmental factors (from sensors) restrict the plan.

Figure 10 illustrates the system diagram. The perception elements consist of the LADAR and vision smart sensors and the GPOS component. The LADAR component utilizes the input from a LADAR sensor and the GPOS data to define the traversability of the terrain local to the vehicle. It also detects obstacles and commits them to the knowledge store component. The vision component extracts line objects and obstacles from the field of view of two cameras. This information is vectorized and stored in the knowledge store. The GPOS component produces a current estimate of the vehicle’s global position and pose.

The subsystem commander establishes which navigational behavior is active and monitors system health. The corridor driver queries the knowledge store for line data to define a corridor. It establishes a series of goals for any instant in time based on the line data. These goals are sent to the high level planner which directs the local path planner. The waypoint driver takes an a priori list of waypoints and sends them to the high level planner in a similar manner.
6. REFERENCES


