Perception and Control Strategies for Driving Utility Vehicles with a Humanoid Robot

Christopher Rasmussen[1], Kiwon Sohn[2], Qiaosong Wang[1], and Paul Oh[2]

***Abstract*— This paper describes the hardware and software components of a general-purpose humanoid robot system for autonomously driving several different types of utility vehicles. The robot recognizes which vehicle it is in, localizes itself with respect to the dashboard, and self-aligns in order to interface with the steering wheel and accelerator pedal. Low- and higher-level methods are presented for speed control, environment perception, and trajectory planning and following suitable for operation in planar areas with discrete obstacles as well as along road-like paths.**

I. Introduction

As part of the recently concluded DARPA Robotics Challenge (DRC) trials [1], contestant robots needed to carry out a number of navigation and manipulation tasks. These tasks were meant to represent a set of skills sufficient for a robot to move from the edge of a disaster zone such as a damaged nuclear power plant to its interior, where it could assess and possibly repair critical systems. The *Vehicle* stage of the challenge [2] called for the robot to drive a golf-cart-like utility vehicle around obstacles to a target location, get out, and walk away (aka *egress*).

In this paper we present techniques for autonomously driving several different utility vehicles using a humanoid robot (the *DRC-Hubo* robot and specific vehicles are described in Sec. II). These methods were developed using a robot entered in the 2013 DARPA DRC trials, but this is *not* a description of our approach to the *Vehicle* task there. At the competition, we used a pure tele-operation approach tuned specifically for the course and low-bandwidth conditions described in the rules [3]. Here we describe a more general set of *autonomous* skills for driving such vehicles in a variety of static environments.

Autonomously-driven vehicles of course have a long history [4], [5], [6], [7], [8] with prominent milestones at the 2005 DARPA Grand Challenge (DGC) [9] and 2007 DARPA Urban Challenge (DUC) [10], [11]. Since the DUC, much progress in the field has come in the industrial sector as automobile manufacturers and Google have extensively refined and tested *driverless car* technologies [12] and in some cases begun to offer them as safety options on production vehicles. These vehicles are effectively robots, but there are a number of significant differences between them and humanoid robots with respect to the structure and difficulty of the driving task.

First, when driverless car technologies are embedded in full-size vehicles, weight, size, and power limitations on the sensors and computers used are not severe. In contrast, the budget for all of these categories is quite limited on a battery-powered humanoid which must worry about balance for walking and maintaining adequate current to all of its joint motors. Moreover, sensor and robot geometry choices cannot be made solely to optimize driving performance, as the point of a humanoid robot is *versatility*. All design decisions impacting driving must be considered jointly with other critical tasks, which for the DRC included walking, stair/ladder climbing, and power tool and door handle manipulation.

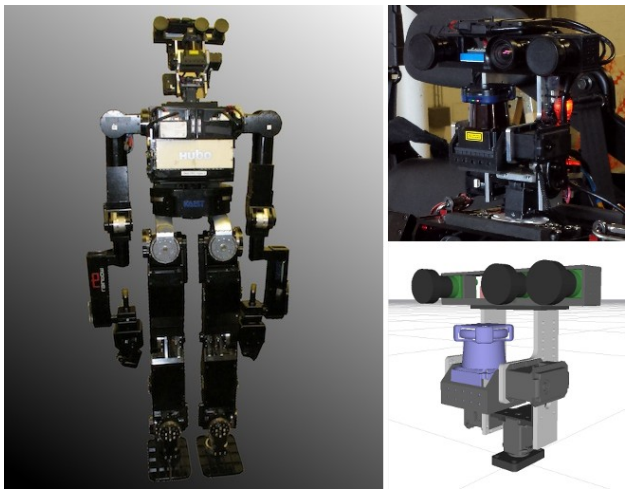
Fig. 1. DRC-Hubo in Polaris vehicle. (Top) Close-up during indoor testing; (bottom) At 2013 DARPA DRC trials (image courtesy of IEEE Spectrum Magazine)

Second, achieving adequate sensor coverage (i.e., "blind spot" elimination) on integrated driverless car systems is generally just a matter of arraying enough fixed camera/sonar/ladar/radar units around the vehicle periphery and on its roof. Conversely, many parts of the road scene are inherently occluded from a humanoid robot inside a vehicle even with omnidirectional sensors on its head. And since the weight/space budget mentioned above makes fewer sensors more desirable and therefore less complete coverage

*This work was supported by DARPA award #N65236-12-1-1005

[1]Dept. Computer & Information Sciences, University of Delaware, USA. E-mail: cer@cis.udel.edu

[2]Dept. Mechanical Engineering, Drexel University, USA



(a) (b)

Fig. 2. (a) DRC-Hubo, (b) Sensor head photo detail and CAD model in ROS `rviz` [18]

(a) (b)

Fig. 3. Vehicles driven: (a) Club Car DS and interior detail; (b) Polaris Ranger XP900 and interior detail. The windshield on the Club Car and the side netting on the Polaris were not present for the experiments reported here. The loose wires in the interior images are part of wireless e-stops added to the vehicles.

unavoidable, the robot must carry out *view planning* to decide where to "look" depending on what it is doing from moment to moment.

Finally, the current generation of driverless cars are typically drive-by-wire, making motion control (steering, acceleration/braking, and shifting) and vehicle state feedback (speed, steering angle, engine temperature, and so on) trivial to implement and basically error-proof. Although there are specialized machines for actuating steering, acceleration, braking, and gear shifting from the driver's seat with no permanent vehicle modifications [13], [14], [15], these take considerable time to set up and calibrate and have no other mobility, manipulation, or perception abilities. On the other hand, a humanoid robot manipulating the steering wheel and pedals is a mechanical system that must self-align, self-stabilize, and monitor for slips and other mishaps. Furthermore, vehicle state variables are not accessible by the robot through simple function calls; rather, they must either be visually read from the dashboard display or inferred from the robot's own sensors and transformed into the vehicle frame.

The main contribution of this paper is a demonstration of the feasibility of a general-purpose humanoid robot driving an unmodified vehicle. The only previous work we can find on humanoid robot vehicle handling is [16], [17], in which an HRP-1 drove a modified forklift and a backhoe, but all control was via tele-operation using a video feed. Here we present a set of perceptual and physical methods which are sufficient to (1) *interface* the robot with different vehicles such that it can reliably accelerate, stop, and steer them as commanded; and (2) perform simple sensing and motion planning while *driving* given its limited and often occluded views.

II. Equipment

A. DRC-Hubo

Our robot, based on the earlier-generation KAIST Hubo 2+ [19], is pictured in Fig. 2(a). It is 1.40 m tall with a wingspan of 2.04 m, weighs 60 kg, and has $N = 33$ degrees of freedom (DoF): 1 in the waist, 6 per leg, 7 per arm, 1 in the left hand fingers, 2 in the right hand fingers, and 3 in the neck/sensor head. Three fingers on each hand close together via one motor for power grasps, and on the right hand there is an additional "trigger" finger which moves independently. Each hand also has a *peg* opposite the palm/fingers side (short versions are shown in Fig. 2(a) and longer ones in Fig. 5) which can be used as a point contact when the robot is in a quadrupedal walking mode. We have also found them useful as essentially rigid fingers for gross manipulation tasks such as turning the steering wheel, explained in Sec. IV-A.

The sensor head on the robot, shown in Figs. 2(a) and (b), was designed and built by us. It pans $\pm 180^\circ$ and tilts $\pm 60^\circ$ without self-collision, and has the following sensors which are relevant to this work:

- **3 × Pt. Grey Flea3 cameras**, each with about a $90^\circ \times 70^\circ$ field of view (FOV), forming a synchronized stereo rig with baselines of 6 cm, 12 cm, and 18 cm.
- **Hokuyo UTM-30LX-EW laser range-finder** which scans at 40 Hz over a 270° FOV at an angular resolution of 0.25°. The minimum detectable depth is 0.1 m and the maximum is 30 m, and intensity-like reflectance information is provided for each point. The Hokuyo is mounted on a dedicated *tilting* servo which has a range of $\pm 60^\circ$ for point cloud capture
- **Microstrain 3DM-GX3-45 IMU** with 3-axis accelerometer, 3-axis gyro, and GPS receiver/antenna

B. Vehicles

Two different utility vehicles, shown in Fig. 3, comprise the set of *known* vehicles $\mathcal{V}$ used for this work: an electric Club Car DS and a gas-powered Polaris Ranger XP900. Common features of these vehicles which distinguish them from passenger cars are an open cabin with the roof

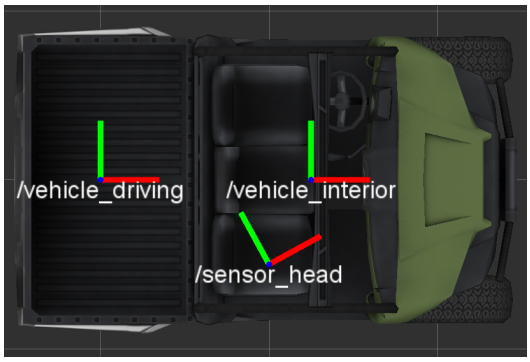


Fig. 4. Orthographic projection of major coordinate frames (vehicle shown is CAD model of Polaris from Gazebo simulator). The `sensor_head` frame moves with the robot. Throughout this paper we use the ROS [18] convention for coordinates of $+X$ forward, $+Y$ left, and $+Z$ up.

supported by relatively thin *pillars*; and a bench-like front seat and no center console, making movement between the passenger's and driver's side during ingress/egress possible. However, there are some key geometric differences. The Polaris is larger overall, and it has a bump on the floor covering the drive shaft, visible in Fig. 3(b). Driving disparities (different steering ratios, implications of electric motor vs. gas engine, etc.) are discussed in Sec. IV-C.

III. Interfacing

In the 2013 DRC trials *Vehicle* task [2], humans were allowed to set the robot up inside a vehicle in a *drive-ready* posture $\mathbf{X}_{drive}$ rather than having it attempt to approach the vehicle, step up, and seat itself (aka *ingress*). By *posture* we mean the complete robot state which combines its joint state $\mathbf{\Theta} = [\theta_1, \theta_2, \ldots, \theta_N]$ and its pose $\mathbf{P} = [x, y, z, \alpha, \beta, \gamma]$ in the `vehicle_interior` frame.[1] For DRC-Hubo this posture, an example of which is pictured in Fig. 5(a), consists of: (a) the robot's torso offset to the right of the steering wheel; (b) its left hand peg inserted between the steering wheel "spokes"; (c) its right hand resting on its lap or grasping a vehicle support structure; and (d) its left foot near the accelerator pedal with its right foot flat on the floor.

This general posture works well for a variety of different utility vehicles, but it must be parametrized by the geometry of each vehicle in terms of steering wheel height off the floor, tilt, radius, and spoke arrangement; roof pillar spacing, angle, and cross-sectional shape/thickness; lateral location of accelerator; and so on. Thus, each drive-ready posture is specific to a vehicle v: $\mathbf{X}^v_{drive} = \{\mathbf{\Theta}^v_{drive}, \mathbf{P}^v_{drive}\}$, and renderings of these are shown for each vehicle in $\mathcal{V}$ in Fig. 5(b) and 5(c).

Manually placing the robot in $\mathbf{X}^v_{drive}$ with adequate pose precision is a time-consuming and strenuous task, and of course it requires a human to specify which vehicle the robot is in. Therefore, we modify the robot insertion process to make it both faster and vehicle-independent. To do so,

[1]This frame, shown in Fig. 4, is convenient for robot motion planning inside the vehicle. Its origin is the intersection between the front of the seats, the floor, and vehicle centerline.

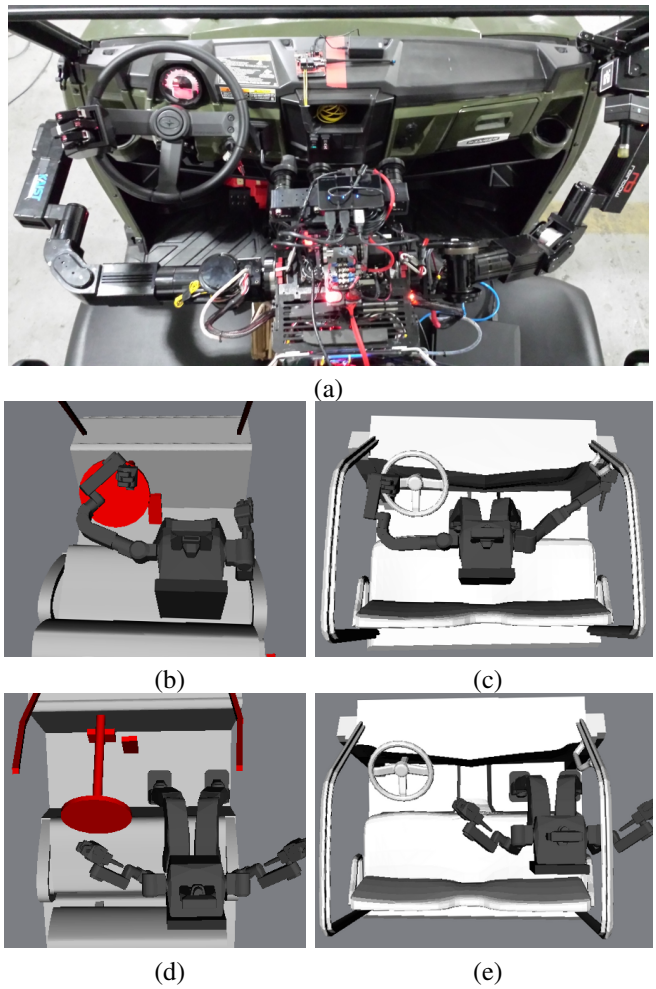


Fig. 5. (a) Top view of a drive-ready posture in Polaris vehicle; (b) $\mathbf{X}^{clubcar}_{drive}$ in OpenRAVE simulation [20]; (c) $\mathbf{X}^{polaris}_{drive}$ in OpenRAVE; (d, e) Neutral joint state $\mathbf{\Theta}_{neutral}$ in Club Car and Polaris (exact pose is not required)

we assume that a high-resolution 3-D point cloud $\mathcal{C}^v_{dash}$ of the *dashboard*[2] of each vehicle $v \in \mathcal{V}$, aligned with the `vehicle_interior` frame, is available as a *reference* for the robot. The dashboard reference clouds used here (shown from different views in Fig. 8 and Fig. 9) were acquired by the ladar on the sensor head tilting at $1^\circ/s$, voxelized to 0.025 m resolution, trimmed of all background features, and are XYZ only.

Autonomous interfacing works as follows:

(1) The robot is placed (or arrives on its own) in the passenger seat in a neutral/vehicle-agnostic sitting position $\mathbf{X}_{neutral}$, illustrated in Fig. 5(d) and 5(e) for each vehicle. The passenger side is advantageous kinematically for steering and entering on that side avoids collision issues with the steering wheel.

(2) The robot obtains a 3-D point cloud/image capture

[2]We define the *dashboard* loosely as the portion of the vehicle interior vertically above the floor and below head height; and longitudinally between the front of the seat and the beginning of the hood. The entire steering wheel assembly is included.

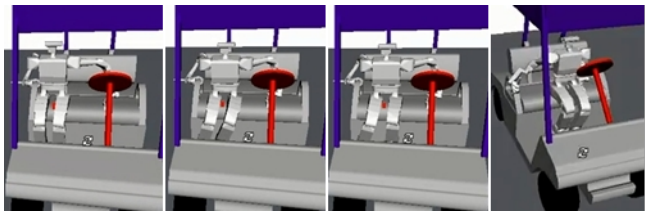


Fig. 6. One cycle of *scooting* motion as robot moves from neutral to drive-ready position, simulated for the Club Car in OpenRAVE

$\mathcal{C}_{neutral}$ of the rough dashboard area in front of it in order to infer the vehicle $\bar{v}$ it is in.

(3) It precisely estimates its initial pose within the vehicle $\mathbf{P}^{\bar{v}}_{neutral}$ by registering $\mathcal{C}_{neutral}$ to the reference $\mathcal{C}^{\bar{v}}_{dash}$.

(4) It formulates and executes an *interfacing* plan to move to $\mathbf{X}^{\bar{v}}_{drive}$. The leftward movement on the seat to reach the drive-ready posture is a stereotyped quadrupedal motion which we call "scooting". After each motion cycle, one of which is shown in Fig. 6, a new point cloud is scanned and step (3) is repeated to decide how far to move or to stop.

A. Vehicle recognition

From any pose inside or near the vehicle, the robot obtains a point cloud of the dashboard region $\mathcal{C}_{neutral}$ by scanning the sensor head ladar as detailed above for the creation of the reference clouds. Examples of such clouds taken from different locations and angles in both vehicles are shown in Fig. 8(a-f) and Fig. 9(a-c).

Recognizing the current vehicle by matching $\mathcal{C}_{neutral}$ to one of the "templates" $\{\mathcal{C}^1_{dash}, \ldots, \mathcal{C}^{|\mathcal{V}|}_{dash}\}$ is a shape retrieval/classification problem [21], [22], [23], [24]. However, the 3-D relief of the two dashboards is similar enough that the error after performing a fit to each reference cloud is not a reliable match indicator. Instead, we exploit the size discrepancy between the vehicles to discriminate them. In particular, the Polaris dashboard is about 0.35 m wider, so after doing a robust vertical plane fit to $\mathcal{C}_{neutral}$, the maximum lateral distance between inliers can be thresholded, following techniques we described in [25], to infer $\bar{v}$. An alternative method to estimate the width is to first detect the roof pillars, which form parallel, nearly vertical cylinders in $\mathcal{C}_{neutral}$. In Sec. III-B we do the equivalent with a single level ladar scan.

B. In-vehicle localization

Believing it is in $\bar{v}$, the robot currently has two ways of estimating its pose $\mathbf{P}^{\bar{v}}$ relative to `vehicle_interior`. The first way, which takes several seconds, is to perform a RANSAC-style [26] robust registration of $\mathcal{C}_{neutral}$ and $\mathcal{C}^{\bar{v}}_{dash}$. Briefly, a minimum set of correspondences to compute a rigid transform $[\mathbf{R}|\mathbf{t}]$ between the two point clouds are repeatedly chosen at random, and the number of inliers calculated for each sample. The transform with the most inliers after a maximum number of iterations is then estimated using a least-squares fit. In order to improve the quality of

(a) (b)

Fig. 7. (a) Peg-in-wheel method for steering (Polaris); (b) Foot arrangement for pedal control (Club Car)

point correspondences, *feature signatures* are calculated to aid matching. We use the Point Cloud Library's (PCL) [27] Sample Consensus Initial Alignment with Fast Point Feature Histograms (FPFH) [28] to perform this step. The initial aligning transform is then refined using Iterative Closest Points (ICP) [29].

Another, much faster but only partial technique for estimating the sensor head pose works with a single approximately level ladar scan. Assuming the robot head is above the height of the dashboard, this scan slices through the vehicle roof pillars, which form two tight clusters. After removing points more than a few meters away, there may still be some objects inside the vehicle (such as the steering wheel or the robot's own hands), so the robot does Euclidean cluster extraction [27] with a small maximum cluster size. All cluster pairs are then checked for 2-D geometric feasibility (distance, angle, etc.), and the most likely feasible pair is used to extract the sensor head yaw and $\mathbf{t}_x, \mathbf{t}_y$.

IV. Driving

We make several important assumptions about the state of the vehicle when the robot begins interfacing: (1) It is powered on or the engine is running; (2) It is in forward mode (Club Car) or drive gear (Polaris); (3) The tires are straight and therefore the steering wheel's orientation is known.

The drive-ready positions depicted in Fig. 5 for the Club Car and Polaris show the robot's right hand resting on its lap and grasping a roof pillar, respectively. At the very low speeds we have currently tested, forces on the robot due to vehicle dynamics are small and therefore active bracing is not *required* in either vehicle. Grasping *is* essential for the stages of ingress/egress in which the robot transitions between the vehicle exterior and interior, but they are outside this paper's scope (more details can be found in [30]).

A. Steering actuation and sensing

The robot turns the steering wheel by *dialing*: it moves its left hand in a circular trajectory with the peg inserted between the steering wheel spokes as shown in Fig 7(a). This motion was chosen over holding the wheel in the traditional manner because it has no singularities, avoiding the necessity of regrasping when turning through large angles. The lack of a slip ring in DRC-Hubo's wrist roll joint precludes grasping

the center of the wheel and turning it directly (the method of autopilots such as [14], [15] and several 2013 DARPA DRC trials teams).

The ideal steering trajectory is determined by the steering circle center, radius, and tilt angle. It is predetermined for each drive-ready position but can be reparametrized from sensor data. The arm joint trajectories for the peg to trace the circle are generated by approximating it with a polygon (the circle is discretized at 30° intervals here), solving the inverse kinematics offline for each vertex to make a look-up table, and linearly interpolating arm joint angles online for steering hand angles between the vertices. Using this method the robot can move its hand and therefore the wheel in a circle at up to $|\dot{\phi}_{hand}|^{\max} = 120°/s$.

A key limitation of this method is *backlash*: the robot hand motion only causes steering wheel motion when it is "pushing" one of the spokes—when the steering direction is changed, there is a gap that the peg must cross before engaging another spoke. As can be seen in Fig. 3, the smallest (depending on which pair of spokes the peg is inserted between) backlash angle β_v for vehicle v is about 90° for the Polaris and 60° for the Club Car. Consequences of the backlash are: (1) Additional bookkeeping in the steering wheel controller, as the hand angle is not necessarily the same as the steering wheel angle (aka $\phi_{hand} \neq \phi_{steer}$); (2) A delay to change the sign of the steering rate $\dot{\phi}_{steer}$ of at least $\tau_{\beta_v} = \dot{\phi}_{hand}/\beta_v$ s; and (3) Inability to resist external forces on the tires due to slope, bumps, etc. which might tend to amplify the turning rate.

DRC-Hubo's left arm joint encoder values and forward kinematics are used to derive ϕ_{hand}, which with a known initial steering wheel angle yields ϕ_{steer}. This can be refined somewhat by sensing contact between the peg and spoke using a force-torque sensor in the wrist. Even with some misalignment, given the current peg length there is no danger of it slipping out of the steering wheel plane during motion.

B. Speed actuation and sensing

The robot affects the vehicle speed solely through varying pressure on the accelerator pedal by changing its left ankle pitch joint angle θ_{LAP} as shown in Fig 7(b), where $\theta_{LAP} = 0$ indicates that the foot and lower leg form a 90° angle. At the low speeds we have driven (≤ 2.5 m/s), a range of $\theta_{LAP} \in [-35°, 5°]$ suffices ($|\dot{\theta}_{LAP}|^{\max} = 400°/s$). At $\theta_{LAP}^{\min}$ the foot is not contacting the accelerator pedal and thus is designated the *stop* angle. This "stopping by deceleration" strategy is adequate because all vehicles in $\mathcal{V}$ coast to a stop in a very short distance at such low speeds, even while still in drive in the case of the Polaris. It is only a problem if the vehicle tries to stop on a slope, which we have avoided in testing.

Active braking is not implemented for several reasons. In the current drive-ready position, the hip yaw required to translate the left foot over to the brake pedal would violate joint limits on DRC-Hubo. A two-footed approach (left on brake, right on accelerator) is possible, but would require a sitting position that is (a) less stable because of the narrower stance, (b) not as good kinematically for steering, and (c) more difficult to get into and out of without colliding during the scooting phase described in Sec. III.

The estimated current vehicle velocity $\tilde{\mathbf{v}}(t)$ is obtained through stereo visual odometry [31], [32]. Specifically, we use synchronized images from the sensor head's 18-cm baseline stereo camera pair taken at 15 Hz to compute a frame-to-frame rigid transform with the `libviso2` library [33], constrain it to planar motion, and transform it into the `vehicle_driving` frame.

C. Motion control

The vehicle speed is controlled very simply. Given a target speed $\mathbf{v}_x^{target}$, the difference between it and the current forward velocity component $\mathbf{v}_x^{target} - \tilde{\mathbf{v}}_x(t)$ is computed and low-pass filtered as $\epsilon(t)$ to get a new pedal command:

$$\theta_{LAP}(t+1) = \begin{cases} \theta_{LAP}^{\min} & \text{if } \mathbf{v}_x^{target} = 0 \\ \theta_{LAP}(t) + \Delta\theta_{LAP} & \text{if } \epsilon(t) \geq T \\ \theta_{LAP}(t) - \Delta\theta_{LAP} & \text{if } \epsilon(t) \leq -T \\ \theta_{LAP}(t) & \text{otherwise} \end{cases} \tag{1}$$

where currently $T = 0.5$ m/s and $\Delta\theta_{LAP} = 1°$.

There are currently two strategies for overall motion control of the vehicle, detailed below.

a) Stop-to-steer: To remove the concerns from Sec. IV-A about the limited turning rate achievable with $|\dot{\phi}_{hand}|^{\max}$ and the backlash lag τ_{β_v} on direction changes, the robot is constrained to only turn the steering wheel while stopped. This results in the vehicle following a Dubins path consisting of circular arc and straight line segments, which may be the original form of the motion plan or simply an approximation of a more complicated trajectory, subject to a full stop between each segment.

Suppose the current step i of an M-step plan (see Sec. IV-D) calls for the stopped vehicle to traverse a segment with curvature κ_i and arc length d_i. With Ackermann steering, the steering ratio r_v, wheelbase w_v, and steering wheel backlash β_v of the vehicle imply a steering *hand* target angle $\phi_{hand}^{target} = f(\kappa_i, r_v, w_v, \beta_v)$, which is then executed. For high κ, this may take several seconds to complete.

$\mathbf{v}_x^{target}$ is then set (1 m/s is the current default), and the vehicle speed controller will slowly push down the pedal until the vehicle begins to move and reaches its target speed. As the vehicle moves, the forward motion component $\tilde{\mathbf{v}}_x(t)$ is integrated until d_i is exceeded, at which point $\mathbf{v}_x^{target}$ is set to 0 and the next plan step $i+1$ is obtained. If at any time during motion an imminent collision is detected because of dynamic obstacles or deviation from the planned trajectory, motion is also halted.

b) Continuous trajectory following: In this mode the robot attempts to follow an arbitrary continuous trajectory without stopping completely. We use a version of the *cross-track error* steering controller from [9]. Modifying their notation slightly to fit ours:

$$\phi_{tires}(t) = \Psi(t) + \arctan \frac{ky(t)}{\mathbf{v}_x(t)} \tag{2}$$

where $\Psi(t)$ is the difference between the tangent to the trajectory and the current vehicle heading, $y(t)$ is the lateral error between the vehicle centerline and the trajectory itself, and k is a gain factor that governs how sharply the vehicle attempts to return to the trajectory curve. The steering command that is generated based on the difference between the actual ϕ_{steer} and the target one is damped to account for the delay while the robot's hand moves, and the target vehicle speed is reduced in proportion to the turn magnitude.

D. Higher-level perception and path planning

There are manifold possibilities for driving perception strategies, but for this prototype we follow [9] and other early DGC participants and simply use the robot sensor head's ladar in a sweep configuration as its chief obstacle detector. When not moving, the robot can obtain a point cloud of the scene outside the vehicle with a single ladar sweep and detect obstacles as outliers to a planar ground fit.

For more efficient storage and processing, ladar scan points from the point cloud are inserted into a 3-D occupancy grid—here we use OctoMap [34] with a minimum cell size of 0.2 m. The map is *rolling*—we are only concerned with what is immediately in front of and around the robot, rather than trying to build a full map of the environment it is traveling through—so the occupancy grid is roughly 10 m on a side, centered forward of the `vehicle_driving` origin.

Each scan point can be annotated with relevant information such as its intensity or obstacle/free classification before insertion into the occupancy grid. In the latter case, this leads to aggregation of multiple observations such that we can get the freespace likelihood for each grid square. The motion estimates furnished by the visual odometry module are crucial in allowing the robot to reason about obstacles it observed previously but can no longer see.

Projecting the current 3-D occupancy grid down to a 2-D costmap allows the robot to generate a feasible trajectory to a goal pose by calling a path planner such as the Search-Based Planning Library (SBPL) [35], [36] which is aware of the vehicle's Ackermann motion constraints. Human operators may also provide high-level plans to the robot by inspecting the current obstacle costmap and supplying Dubins-like segments to be executed via the stop-to-steer method of Sec. IV-C.

A final perception and planning mode we have developed is for path- or road-following. Following our methods for trail-following detailed in [37], [38], we can track a path in front of the vehicle by formulating a *path likelihood* function which measures how well a low-dimensional path shape hypothesis (width, lateral offset, relative heading, and curvature) agrees with sensor measurements and continuously optimizing it via particle filtering. The likelihood function used here simply measures the proportion of ground fit inliers to outliers taken from the costmap introduced above, although more sophisticated appearance cues could easily be incorporated. The centerline of the tracked path supplies the cross-track error and path segment orientation needed for the continuous trajectory following method in Sec. IV-C.

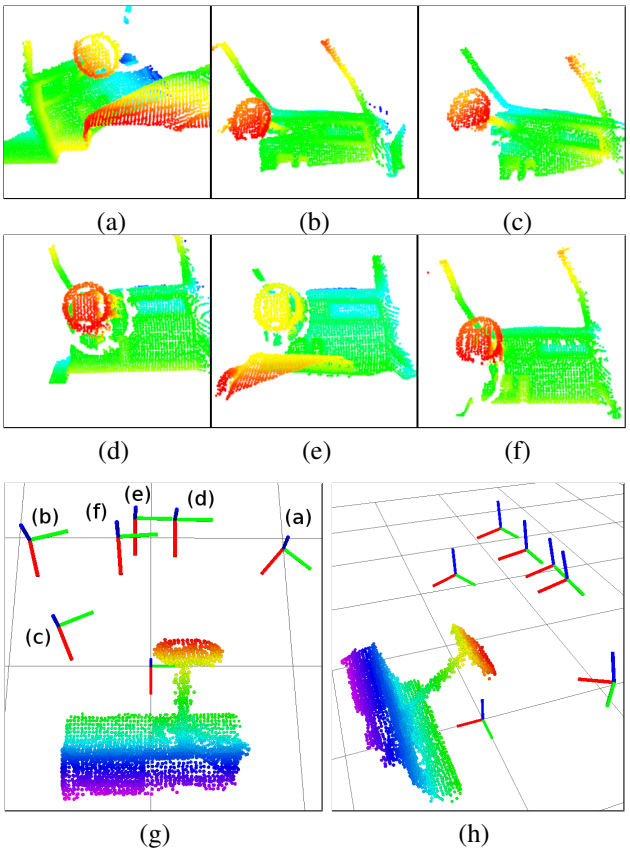


Fig. 8. Club Car dashboard registration for sensor head pose estimation. (a-f) Hokuyo point clouds acquired from different poses, in `sensor_head` frame; (g, h) Views of estimated sensor head locations with respect to the dashboard reference cloud and origin (all point clouds colored by X, grid squares are 1 m)

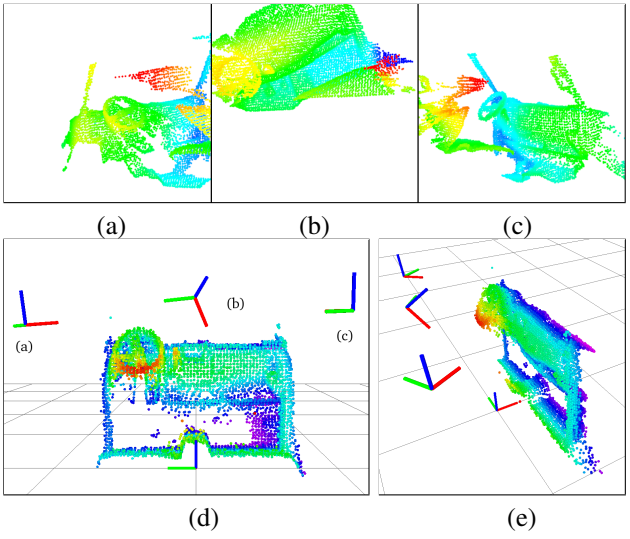


Fig. 9. Polaris dashboard registration for sensor head pose estimation. (a-c) Hokuyo point clouds; (d, e) Views of estimated sensor head poses with respect to reference cloud

V. Results

Dashboard point cloud registration and sensor head pose estimation results are shown for the Club Car in Fig. 8 and

the Polaris in Fig. 9. Scans were captured from about 10 different locations for each vehicle along the driver-passenger seat, with some slightly outside the envelope of the vehicle. Some poses were quite close together or varied only by orientation, so for clarity we have omitted these. On an Intel Core i7-3930K processor at 3.8 GHz with 64 Gb of RAM, the mean combined time of the FPFH and ICP stages of dashboard point cloud registration for the Club Car and Polaris data is about 5 seconds. This operation only needs to be run a few times during interfacing, so the wait is not unreasonable.

In all but one case, the results returned were very good, and for that exception there was a roll error of about 10°. Because of the near-planarity of the Polaris dashboard reference cloud, there were some ambiguities due to symmetries that were eliminated by enforcing some mild constraints on the sensor orientation and position.

All steering and speed actuation methods described in Sec. IV-A and Sec. IV-B have been extensively tested on DRC-Hubo in both the Club Car and Polaris in a variety of environments. Several versions of the *stop-to-steer* motion control method from Sec. IV-C were demonstrated in the Polaris in both indoor and outdoor environments, shown in Fig. 1. A sequence from one of many indoor obstacle avoidance tests with DRC-Hubo at the wheel, using human-provided plan segments, is shown in Fig. 10.

The obstacle detection, mapping, search-based planning, and continuous trajectory following methods from Sec. IV-D have been demonstrated in an integrated fashion in simulation (using ground-truth odometry). A snapshot of a Gazebo simulation is shown in Fig. 11(a), with the generated Octomap and planned trajectory several frames later shown in Fig. 11(b).

The perception and path tracking algorithms have further been validated offline using data collected with the sensor head while driving manually 5+ km around campus and golf course testing areas (samples pictured in Fig. 11(c) and (d)). Steps of the pipeline from an area where the vehicle was driven through a 3-point turn are represented in Fig. 11(e). The two left images show the scene and its corresponding point cloud, while the two right images show the ground/obstacle segmentation and height-colored Octomap of the point cloud, with the visual odometry-derived motion estimate of the maneuver overlaid.

VI. Conclusion

This work is a preliminary demonstration of the feasibility of humanoid robots driving vehicles, and much remains to be done to make the system more practical, robust, and general. Beyond more integrated live testing of continuous trajectory following and path tracking, of immediate interest would be adding skills to relax the assumptions in Sec. IV, including turning the vehicle on/off with a key or switch and gear shifting to allow reverse maneuvers in the motion planner. Both of these would require more dextrous manipulation, including force feedback, and visual analysis for hand-eye coordination.

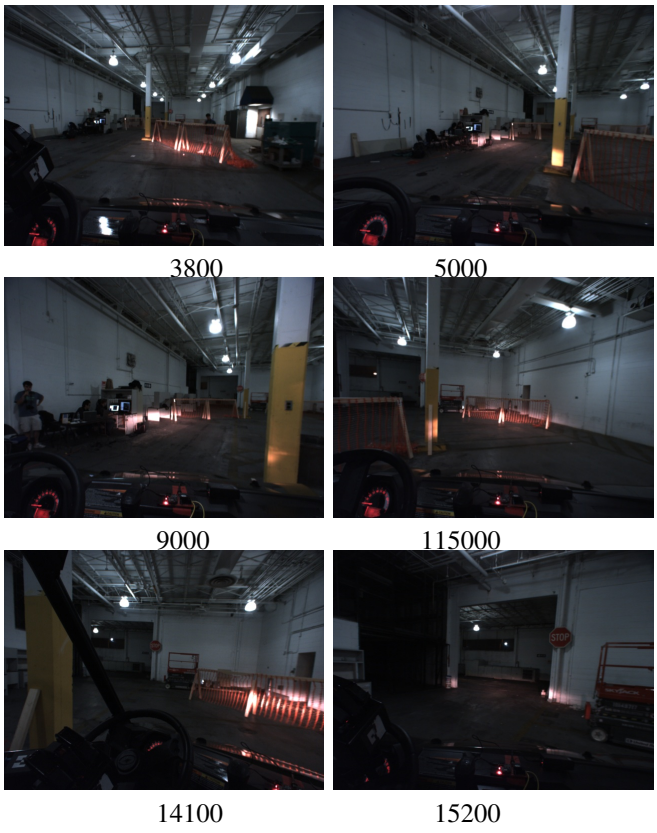


Fig. 10. Image sequence (with frame numbers) from sample successful Polaris traversal of garage obstacle zone. Stop-to-steer tele-operation was used.

There are no major barriers to extending the perception and motion planning system to work in more situations by using visual segmentation and detection of lane lines, signs, and so on, as well as dealing with dynamic agents in the environment. Adding more flexibility here would require a full consideration of the view planning problem mentioned in the introduction, as the robot would need to not only pan and tilt the sensor head but possibly lean its torso in certain situations to mitigate blind spots. Visual information would also help with vehicle recognition within a larger set of possibilities than the two examined here. With autonomous ingress such recognition would likely occur outside the car, but part detection and tracking inside the car could help speed pose estimation and allow the reading of visual indicators like the speedometer and gear state.

References

[1] DARPA, "DARPA Robotics Challenge website," 2013, available at `http://darparoboticschallenge.org`. Accessed December, 2013.

[2] ——, "DARPA Robotics Challenge Vehicle task description," 2013, available at `http://darparoboticschallenge.org/node/159`. Accessed December, 2013.

[3] ——, "DARPA Robotics Challenge DRC Trials Rules (Release 7)," 2013, available at `http://www.theroboticschallenge.org/files/DRCTrialsRulesRelease7DISTAR22157.pdf`. Accessed December, 2013.

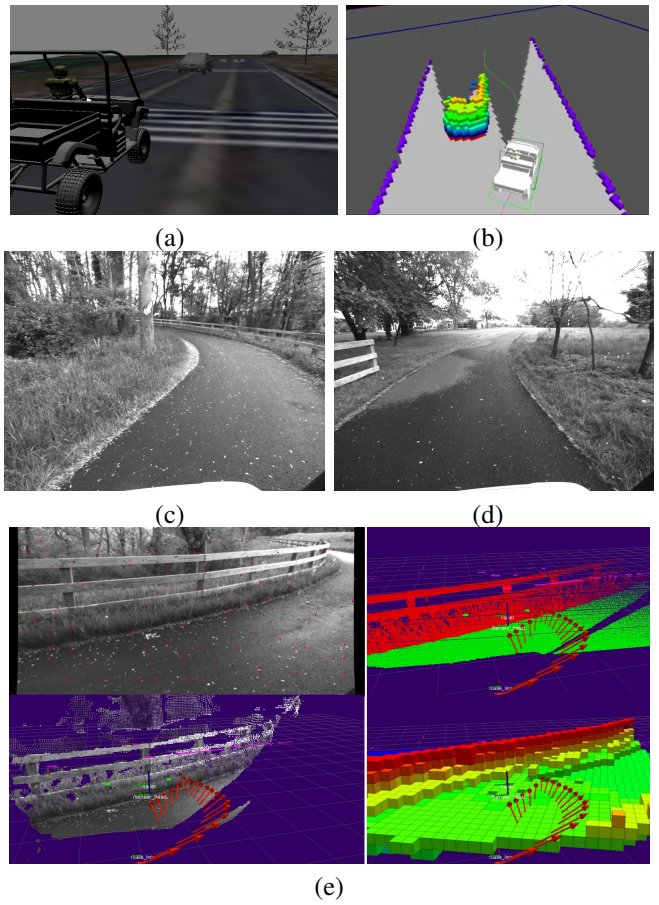


Fig. 11. (a) ROS Gazebo driving simulation; (b) Corresponding `rviz` visualization of occupancy grid, plan generation, and trajectory following; (c, d) Sample terrain in golf course data collection area; (e) Upper-left: golf course scene; lower-left: dense stereo reconstruction with motion history overlaid (vehicle stopped and reversed); upper-right: freespace/obstacle segmentation after ground plane fit; lower-right: Octomap occupancy grid colored by height

[4] J. Crisman and C. Thorpe, "UNSCARF, a color vision system for the detection of unstructured roads," in *Proc. IEEE Int. Conf. Robotics and Automation*, 1991, pp. 2496–2501.

[5] E. Dickmanns and B. Mysliwetz, "Recursive 3-d road and relative ego-state recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 2, no. 14, pp. 199–213, 1992.

[6] D. Pomerleau, "RALPH: Rapidly adapting lateral position handler," in *Proc. IEEE Intelligent Vehicles Symposium*, 1995, pp. 506–511.

[7] B. Southall and C. Taylor, "Stochastic road shape estimation," in *Proc. Int. Conf. Computer Vision*, 2001, pp. 205–212.

[8] N. Apostoloff and A. Zelinsky, "Robust vision based lane tracking using multiple cues and particle filtering," in *Proc. IEEE Intelligent Vehicles Symposium*, 2003.

[9] S. Thrun, M. Montemerlo, *et al.*, "Stanley, the robot that won the DARPA grand challenge," *J. Field Robotics*, vol. 23, no. 9, 2006.

[10] C. Urmson *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *J. Field Robotics*, vol. 25, no. 1, 2008.

[11] A. Huang, D. Moore, M. Antone, E. Olson, and S. Teller, "Multi-sensor lane finding in urban road networks," in *Robotics: Science and Systems*, 2008.

[12] B. Bilger, "Auto correct: Has the self-driving car at last arrived?" *The New Yorker*, November 2013.

[13] S. Shoval, J. Zyburt, and D. Grimaudo, "Robot Driver for Guidance of Automatic Durability Road (ADR) Test Vehicles," in *Proc. IEEE Int. Conf. Robotics and Automation*, 1998.

[14] Stahle Robot Systems, "AUTONOMOUS DRIVING SYSTEM SFP2000FF for cars," 2014, available at `http://www.stahle.com`. Accessed January, 2014.

[15] Kairos Autonomi, "Pronto4 Agnostic Autonomy System for Existing Vehicles or Vessels," 2014, available at `http://www.kairosautonomi.com/pronto4_system.html`. Accessed January, 2014.

[16] H. Hasunuma, M. Kobayashi, H. Moriyama, T. Itoko, Y. Yanagihara, T. Ueno, K. Ohya, and K. Yokoi, "A Tele-operated Humanoid Robot Drives a Lift Truck," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2002.

[17] K. Yokoi, K. Nakashima, M. Kobayashi, H. Mihune, H. Hasunuma, Y. Yanagihara, T. Ueno, T. Gokyuu, and K. Endou, "A Tele-operated Humanoid Robot Drives a Backhoe in the Open Air," in *Proc. Int. Conf. Intelligent Robots and Systems*, 2003.

[18] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: An Open-Source Robot Operating System," in *Proc. ICRA workshop on Open-Source Software*, 2009.

[19] R. Ellenberg, R. Sherbert, P. Oh, A. Alspach, R. Gross, and J. Oh, "A common interface for humanoid simulation and hardware," in *Proc. IEEE Inter. Conf. on Humanoid Robots*, 2010.

[20] R. Diankov, "Automated construction of robotic manipulation programs," Ph.D. dissertation, Carnegie Mellon University, Robotics Institute, August 2010.

[21] A. Frome, D. Huber, R. Kolluri, T. Bülow, and J. Malik, "Recognizing objects in range data using regional point descriptors," in *Proc. European Conf. Computer Vision*, 2004.

[22] J. Assfalg, M. Bertini, and A. D. Bimbo, "Content-based retrieval of 3-d objects using spin image signatures," *IEEE Trans. Multimedia*, vol. 9, no. 3, 2007.

[23] R. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3D Recognition and Pose Using the Viewpoint Feature Histogram," in *Proc. Int. Conf. Intelligent Robots and Systems*, 2010.

[24] W. Wohlkinger and M. Vincze, "Ensemble of shape functions for 3D object classification," in *IEEE Inter. Conf. on Robotics and Biomimetics*, 2011.

[25] C. Rasmussen, K. Yuvraj, R. Vallett, K. Sohn, and P. Oh, "Towards Functional Labeling of Utility Vehicle Point Clouds for Humanoid Driving," in *IEEE Inter. Conf. on Technologies for Practical Robot Applications*, 2013.

[26] M. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[27] R. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *Proc. IEEE Inter. Conf. on Robotics & Automation*, 2011.

[28] R. B. Rusu, "Semantic 3d object maps for everyday manipulation in human living environments," Ph.D. dissertation, Computer Science department, Technische Universitaet Muenchen, Germany, October 2009.

[29] P. Besl and N. McKay, "A method for registration of 3-d shapes," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[30] K. Sohn, "Optimization of humanoid's motions under multiple constraints in vehicle handling task," Ph.D. dissertation, Mechanical Engineering and Mechanics, Drexel University, May 2014.

[31] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry for ground vehicle applications," *J. Field Robotics*, vol. 23, no. 1, 2006.

[32] A. Howard, "Visual odometry for motion estimation," in *Proc. Int. Conf. Intelligent Robots and Systems*, 2008.

[33] A. Geiger, J. Ziegler, and C. Stiller, "StereoScan: Dense 3D Reconstruction in Real-time," in *IEEE Intelligent Vehicles Symposium*, 2011.

[34] A. Hornung, K. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013.

[35] M. Likhachev. (2013) Ros search-base planning library stack. [Online]. Available: http://www.ros.org/wiki/sbpl

[36] M. Pivtoraiko and A. Kelly, "Generating Near Minimal Spanning Control Sets for Constrained Motion Planning in Discrete State Spaces," in *Proc. IEEE/RSJ Inter. Conf. on Intelligent Robots & Systems*, 2005.

[37] C. Rasmussen, Y. Lu, and M. Kocamaz, "Integrating stereo structure for omnidirectional trail following," in *Proc. Int. Conf. Intelligent Robots and Systems*, 2011.

[38] ——, "A trail-following robot which uses appearance and structural cues," in *Proc. Int. Conf. on Field and Service Robotics*, 2012.