Demonstrating HOUND: A Low-cost Research Platform for High-speed Off-road Underactuated Nonholonomic Driving

Sidharth Talia, Matt Schmittle, Alexander Lambert, Alexander Spitzer, Christoforos Mavrogiannis, Siddhartha S. Srinivasa

***Abstract*—Off-road autonomy, crucial for applications such as search-and-rescue, agriculture, and planetary exploration, poses unique problems due to challenging terrains, as well as due to the risk involved in testing or deploying such systems. Accessible platforms have the potential to widen the field to a broader set of researchers and students. Existing efforts in making on-road autonomy more accessible have seen success, yet aggressive off-road autonomy remains underserved. We seek to fill this gap by introducing HOUND, a 1/10th-scale, inexpensive, off-road autonomous car platform that can handle challenging outdoor terrains at high speeds. To aid development speed, we integrate HOUND with BeamNG, a state-of-the-art driving simulator to enable both software in the loop as well as hardware in the loop testing. To reduce the extent of ruggedization required, and thus cost, we integrate a rollover prevention system as a safety feature into the platform. Real-world trials over 50 kilometers demonstrate the platform's longevity and effectiveness over varied terrains and speeds. Build instructions, datasets and code disseminated via: https://sites.google.com/view/prl-hound/home**

I. Introduction

Off-road autonomy has various applications, including search-and-rescue [57], agriculture [16], and planetary exploration [53]. Here, off-road environments refer to environments with highly uneven terrain, with structures such as hills or ditches as tall or deep as, if not taller or deeper than, the vehicle itself. Off-road environments are not tailored to be traversed by vehicles; they present low traction and bumpy surfaces leading to difficulties in state estimation as well as control. Such environments can also be physically unforgiving, resulting in potential sensor degradation if not outright damage. Unlike on-road driving, there are no rules that explicitly dictate expected vehicle behavior. This can make benchmarking and comparison of off-road systems non-trivial.

Much of the progress in modern robotics has come from large efforts toward improving infrastructure for research and education. On-road autonomous driving, a generally resource-intensive field, is being democratized by efforts such as MuSHR [46] and F1-tenth [38], which lowered the barriers of entry by providing a small, low-cost, easy-to-use hardware/software platform. The AutoRally [21] platform had a similar aim for off-road aggressive driving, being less expensive and smaller than the platforms used in other works [9, 33]. While it created a significant cost reduction, this work seeks to further lower the cost through new low-cost hardware and novel algorithms previously not available, to enable off-road autonomy research for a larger audience.

Fig. 1: HOUND has been tested on dirt hills, grasslands, gravel trails, and tarmac.

As there are no explicit rules in off-road autonomy, evaluating an autonomy stack requires knowing whether its actions would incur damage or not. As such, we provide integration with BeamNG [5], a high-fidelity vehicle simulator used both in the industry as well as academia [6, 31], known particularly for its accurate crash simulation.

A major concern for high-speed off-road navigation tends to be rollovers [21], as repeated rollovers can damage the components over time, costing both money and time in repairs. In essence, the vehicle rolls over due to excess lateral acceleration. As such, we integrate a low-level rollover prevention system (**RPS**) as a safety feature into our platform that allows researchers to push the limits without needing to repair the platform regularly or using expensive, ruggedized hardware.

The main contribution of our work is an integrated platform (see Fig1), with open-source software, and hardware, that can be used by off-road autonomy researchers, and robotics researchers in general, at a relatively low cost of ≈$3000. Our open-source software stack removes a major barrier to research: the need to engineer an entire off-road autonomy stack before starting focused research. Further, we believe that a common accessible platform enables easy data sharing across research projects. We demonstrate the utility of the simulator by using it to validate the RPS. We evaluate the RPS both in an isolated fashion, as well as in conjunction with a high-level controller to show that it does not interfere excessively. Finally, we test the longevity of this system in the real world for **50 km** over 4 different terrains, reaching speeds up to **7** m/s and lateral accelerations up to **9** m/s^2.

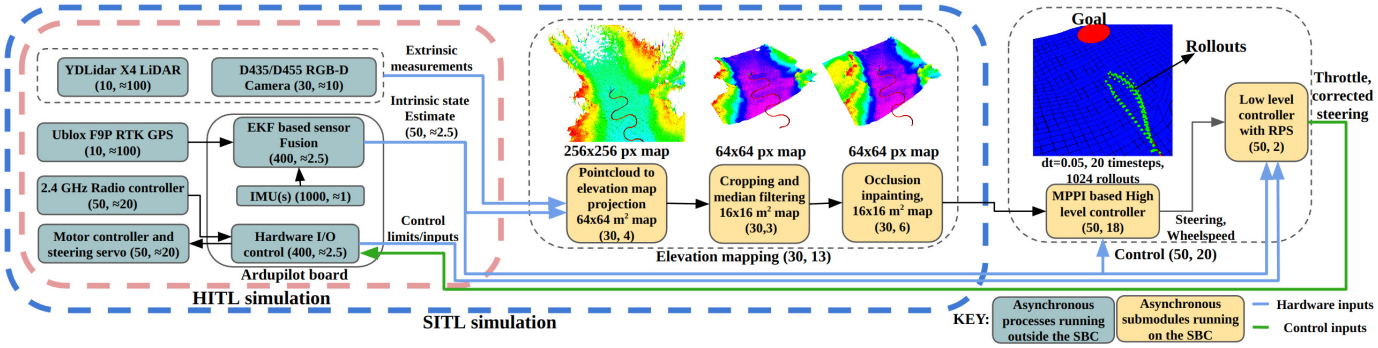


Fig. 2: Autonomy stack components, specified with (rate in Hz, latency in ms), with ≈ implying estimated latency based on maximum update rate. SITL spoofs perception, useful for isolating control problems, whereas HITL spoofs sensors, useful for testing deployed software.

II. Related Work

A. Aggressive off-road autonomy

Aggressive off-road autonomy presents unique challenges as compared to on-road driving. The surface-tire interactions become more relevant due to slippage, vibrations due to terrain unevenness can make inertial state estimation harder, and occlusion due to environmental structures such as trees and ditches can make accurate mapping challenging. As such, work has been done towards perceiving outdoor environments as probabilistic elevation maps [41, 34], and their extensions [48, 33] which use data-driven methods to address environment occlusion via inpainting. Work by Fork et al. [17] has extended planar dynamics models [20] towards non-planar surfaces, whereas other works [29, 19] have presented approaches for learning such 3-dimensional dynamics. In this work, we incorporate some of the recent advances in off-road autonomy for extrinsic state estimation [34, 48] and control [27, 52] into a real-world system.

B. Research platforms

Full-scale autonomous driving systems are expensive, require vast resources, such as a large testing area, and can pose safety risks. As such, small-scale research platforms have become a popular entry point to autonomous driving research [35, 26, 13, 46, 38]. Notably, the MuSHR[46] and F1-tenth[38] platforms, inspired by the MIT-RACECAR[35], are popular platforms that have enabled a wide variety of research, even in external institutions [24, 8, 51]. The AutoRally [21] platform further enabled research in outdoor environments [59, 39], but is significantly more expensive due to the use of industrial-grade sensors and the ruggedization required for off-road driving (see Table I). In this work, we incorporate a rollover prevention system into the autonomy stack to reduce the cost associated with ruggedization. Our work is related to the VertiWheeler [14], a recent work focusing on off-road navigation for rock crawling at lower speeds, but our focus is more on high-speed off-road navigation with rollover prevention. FastRLap [47] is a recent work focusing on using reinforcement learning for outdoor navigation with a small-scale platform, whereas we focus on creating a standardized small-scale research and education platform for outdoor autonomy.

C. Rollover prevention

Rollover prevention is a popular problem in the domain of utility-grade automobile research, as it poses a safety risk to large vehicles or vehicles with a high center of mass during turning or side-slope traversal [45, 43, 55]. In this context, an Advanced Driver Assistance System(ADAS) uses a rollover index **(RI)** such as Load Transfer ratio [56, 15], Force angle measure [22, 40] or Time to Rollover [11] to detect an imminent rollover and either alert the driver or additionally apply remedial controls [54]. For autonomous systems, works such as [3, 29] propose using the rollover indicator as part of the cost for the model predictive controller (MPC). For safety-critical tasks such as rollover prevention, erroneous extrinsic/intrinsic state estimation or dynamics mismatch can result in catastrophic failure. In our work, we incorporate a low-level reactive controller that uses Force angle measure [40] as the RI, inspired by Yedavalli and Huang [60], into the autonomy stack that only depends on an inertial measurement unit **(IMU)** and a wheel speed sensor. Additionally, we show that such a system can work in conjunction with an MPC that uses the rollover cost without loss of performance while reducing the chance of incidental rollovers.

III. System Overview

A. Hardware specifications

The HOUND's physical platform is built using the Blackout SC-1/10 platform, also used in the MuSHR [46] due to its low cost and low weight as compared to a 1/5th scale platform. As the autonomy stack is parameterized by vehicle properties, the use of a bigger/different chassis is not precluded by our work. Details regarding the sensor stack are shown in Fig. 2, Table I. The onboard single-board-computer **(SBC)** is an NVIDIA Jetson Orin NX as it provides a good compromise between cost and size-weight-and-power (**SWaP**) when compared to the Nano and AGX variants. The physical platform weighs close to 4 Kg and can reach speeds beyond 12 m/s on tarmac. For system hardware specifications, please see table II, Fig 2. In contrast to previous outdoor autonomy platforms such as AutoRally [21], and FastRLap [47], which perform GPS-IMU fusion on the SBC, we use the Ardupilot [2] framework. Here, a separate microcontroller board running the Ardupilot software [32] runs an EKF [42], currently for GPS-IMU fusion,

Platform	Focus	Sensing	Full cost	Autonomy cost(sensing & compute)	Hardware / Software Rollover protection
F1Tenth [38]	Indoor	IMU, RGBD, LiDAR	$3,000	$2,000	None / None
MuSHR [46]	Indoor	IMU, RGBD, LiDAR	$1,000	$500	None / None
AutoRally [21]	Outdoor	IMU, Stereo RGB, GPS	$15,000	$9,000	Steel body, industrial grade sensors / None
HOUND	**Outdoor**	**IMU, RGBD, LiDAR, GPS**	**$3,000**	**$2,000**	**Enclosed plastic shell / RPS**

TABLE I: Cost/features comparison among platforms. Costs approximated to the nearest 1000. Autorally's use of industrial grade sensors results in an autonomy cost $\approx$ 4x that of the HOUND, which uses commercial grade sensors and RPS to protect against rollover damage.

Specification	Value
Max. wheelspeed (max V_w)	$23.0m/s$ (no load)
Static rollover limit (RI_L)	≈ 0.9
Battery backup (idle/driving)	80 mins/15 mins @ 5m/s
Weight (with battery)	$\approx 4.0Kg$
Component	**Name**
High-level computation	Nvidia Jetson Orin NX 16 GB
Ardupilot board	mRo PixRacer Pro (3 IMUs, 1 Barometer)
GPS	Ublox F9P-01B RTK
RGB-D camera	Intel Realsense D435/D455
LiDAR	YDLidar X4
Motor control/feedback	Flipsky FSESC 4.12
Chassis	Redcat racing Blackout SC-1/10
Steering Servo	ProTek RC 170SBL
Battery	Spektrum SPMX324S100 (47.36Wh)

TABLE II: Note that while the maximum wheelspeed can reach 23 m/s, autonomy experiments are not expected to exceed $7-9$ m/s.

but optionally for fusing additional odometry measurements, either from visual odometry or motion capture. Ardupilot has been used in this capacity before by works [36, 25]. Additionally, it provides a hardware bridge between the SBC and the chassis with several fail safes as well as various tools such as Missionplanner [12] for convenient field testing. The above components are housed inside a 3D-printed body (see Fig. 3) designed to protect against environmental factors and incidental rollovers, up to a limit. For more details, refer to Appendix VII-A and VII-B.

B. State estimation

The intrinsic state estimation, which includes pose-twist estimation, and IMU filtering, is done on the Ardupilot board and communicated to the SBC (see Fig. 2 for details). For extrinsic state estimation, we use GPU-based probabilistic elevation mapping [34] in conjunction with a learning-based elevation inpainting system [48] to address occlusions. The elevation map is body-centric. On the Orin NX, the latency of projecting the pointcloud to the elevation map is invariant up to a relatively large map size, however, the latency of inpainting scales quadratically with the map size. Thus, we maintain a large "raw" elevation map, and inpaint only a smaller, cropped portion of it, big enough for navigation at 6-8 m/s with planning horizons up to 1 second. Before inpainting, spikes in the elevation map that occur due to noise in depth sensing are median filtered. This minimizes the latency without excessively compromising on map quality and makes inpainting easier over time when the car operates in a small loop. Details regarding the map size, latencies and update rates can be found in Fig. 2.

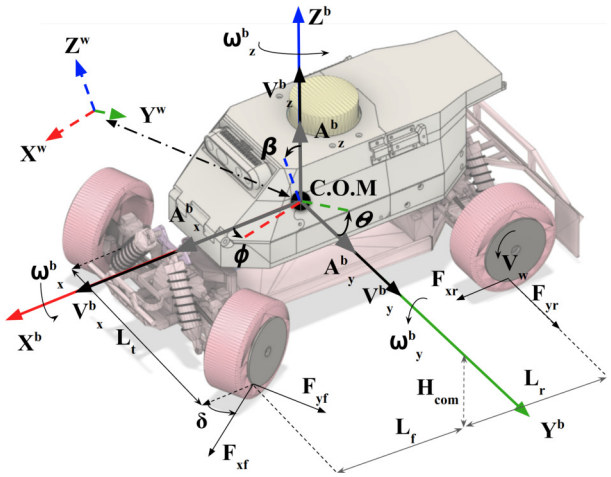


Fig. 3: Illustration of the coordinate frame used by HOUND

C. High-level controller

The controller's state vector consists of world frame position X^w, Y^w, Z^w, world frame roll-pitch-yaw ϕ, θ, ψ, body frame velocity V^b, body frame linear acceleration A^b, and body frame rotation rates ω^b in the reference frame shown in Fig. 3. The controller also takes the elevation map, and a path $G = (g^1, \ldots, g^m), g^j \in \mathbb{R}^2$ in the world frame. The controller produces wheel speed V_w and steering angle δ as the output. The end users may implement any controller that follows the aforementioned input-output scheme. As the default high-level controller, we implement a variant of MPPI [27] as our sampling-based controller using Pytorch and PyCUDA.

1) Dynamics model: We use a single-track bicycle model [52], which considers non-planar surfaces, with a simplified Pacejka tire model [1], as it performs better than the kinematic no-slip model [28] near the tire's grip limit [49]. The dynamics model is used to predict future states for a given control sequence $U = (u^1, \ldots, u^m), u^j = (\delta^j, V_w^j)$. We assume that the tires always remain in contact, as such the height and tilt –z, ϕ, θ–are obtained by projecting the location of the tires on the elevation map and $V_z^b = 0$, as done by Meng et al. [33]. The body rates ω_x^b, ω_y^b are obtained by transforming world frame roll-pitch-yaw rates into the body frame [30]. The body frame forces F_x^b, F_y^b, F_z^b and rotational acceleration $\dot{\omega}_z^b$ are given by:

$$\begin{aligned} F_x^b &= F_{xr} + F_{xf}\cos(\delta) - F_{yf}\sin(\delta) + mg\sin\theta, \\ F_y^b &= F_{yr} + F_{yf}\cos(\delta) + F_{xf}\sin(\delta) + mg\sin\phi, \\ F_z^b &= m(g\cos\beta - V_x\omega_y + V_y\omega_x), \\ \dot{\omega}_z^b &= ((F_{xf}\sin(\delta) + F_{yf}\cos(\delta))L_f - F_{yr}L_r)/J_z \end{aligned} \tag{1}$$

Where F_{xf}, F_{yf} represent the wheel-frame longitudinal and lateral forces generated by the front tire, F_{xr}, F_{yr} represent the same for the rear tires, β represents the angle made by the body-frame Z-axis with the world's Z axis(see Fig. 3), L_f, L_r represent the distance from the center of mass to the front and the rear axle respectively, m represents the mass of the vehicle, J_z represents the mass moment of inertia around its Z axis. These lateral and longitudinal accelerations are then used to update the body frame velocity V^b, which is transformed into the world frame to update the position.

2) Cost function: For the cost along the trajectory, following recent work [33, 29], we aim to regulate vertical force F_z^b, angle to the world frame Z axis β, body frame forward velocity V_x^b, rollover index RI (see Eq. 4), cross-track error $E_x(p)$ and Euclidean distance to goal along the path $E_g(p)$, $p = (X^w, Y^w), p \in \mathbb{R}^2$.

$$\begin{aligned} C = W_f T(F_z^b, F_L^b) + W_\beta T(\beta, \beta_L) + W_v T(V_x^b, V_L) + \\ W_r T(RI, RI_L) + W_s E_x(p) + W_g E_g(p) \end{aligned} \tag{2}$$

Where C is the total cost that gets summed along the trajectory. Note that the terms with a "L" subscript represent the limiting value for the corresponding variable, and $T(x, L) = \max(0, x-L)$. $W_f, W_\beta, W_v, W_r, W_s, W_g$ are weights on these costs.

D. Rollover Prevention

While we build on existing work (see II-C), we outline the specific implementation of rollover prevention used in our work as our constraints may differ from those of prior works. The RPS is part of the Low-Level Controller. The Low-Level Controller allows some form of operator intervention at all times, thus, the operator can always prevent collisions based on visual contact. However, rollovers are caused by inertial effects not felt outside the vehicle, which makes them much harder for the operator to prevent. The RPS is safety-critical, therefore, we avoid dependence on intrinsic and extrinsic state estimation, which may be erroneous. As such, the method outlined here only depends on wheel speed and IMU measurements. We also do not assume knowledge of the tire parameters and so, we assume no slip. We show that the dependence on this assumption is not strong. Additionally, the RPS must not impose substantial computing requirements as it must run constantly in the background at a high update rate. Due to these constraints on state estimation, knowledge of vehicle parameters, and compute power, model-based based approaches to rollover prevention, such as [29, 33] do not address our problem. We assume that the steering actuation is instantaneous. Finally, we assume that the wheels remain in contact with the ground at all times. More details for the RPS and low-level controller can be found in Appendix VII-C

Consider a car turning on a sloped surface, which induces a positive roll angle ϕ, being observed in a non-inertial frame. The lateral and vertical accelerations (A_y^b, A_z^b resp.) produce counteracting moments around the contact point of the right-side tires with arm lengths H_{com}, $L_t/2$, resp. where H_{com} refers to the height of the center of mass above the contact surface under the vehicle and L_t refers to the track width of the vehicle(see Fig. 3). A rollover is imminent when the moment produced by A_y^b exceeds that of A_z^b [40]. Let this critical lateral acceleration be A_y^c.

$$\begin{aligned} &|A_y^b| H_{com} < A_z^b L_t/2 \quad OR \quad |A_y^b| < A_y^c, \\ &A_y^c = A_z^b L_t/(2H_{com}) \end{aligned} \tag{3}$$

We use the force-angle-measure as our rollover[40] index(**RI**) which is the ratio of lateral to vertical acceleration.

$$\begin{aligned} RI &= A_y^b/A_z^b, \\ RI_L &= L_t/(2H_{com}) \end{aligned} \tag{4}$$

Where RI_L refers to the limiting or critical value of the rollover index beyond which a rollover is imminent($|RI| \geq RI_L$), following from the expression of A_y^c in 3. Note that the term "angle" is used in the name as one can interpret the RI as $\arctan(A_y^b/A_z^b)$, though we use it as a ratio for mathematical convenience.

The RPS consists of two parts; the static limiter, and the feedback controller.

1) Static Limiter: Assuming no slip, the wheel speed V_w and steering input δ produce a centrifugal force that can be used to obtain the steering angle limit. As the tires will always have some slip, this limiter will underestimate the steering angle limit, and so we also introduce a user-tunable "slack" δ_s. We also account for gravity-induced lateral acceleration.

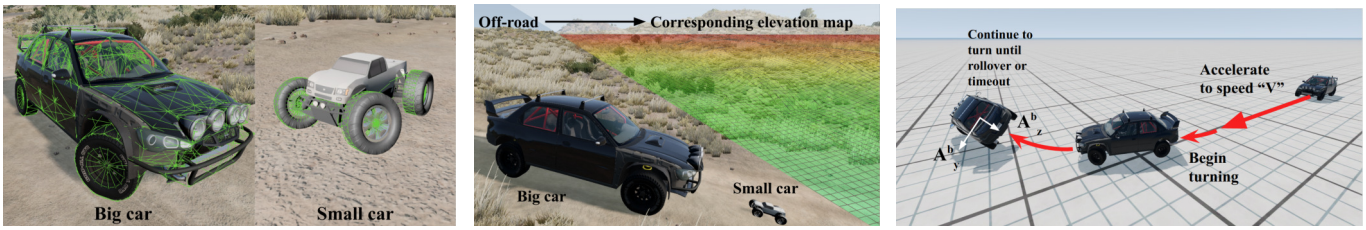
$$\begin{aligned} A_y^b &= V_w^2 \tan(\delta)/L_{fr}, \\ \delta_C &= \pm \tan^{-1}((A_y^c \mp g \sin(\phi)) L_{fr}/V_w^2) \pm \delta_s \end{aligned} \tag{5}$$

Where L_{fr} represents the wheelbase. Note that the slack variable allows for a configurable reduction in the conservatism that arises from the no-slip assumption and is tuned empirically on the platform.

2) Feedback Control: Feedback control is necessary as the static limiter can suffer rollovers from model mismatch or environmental factors. This also helps the end-user tune δ_s; ideally, the feedback mechanism should never kick in, while using as much slack as possible. The feedback controller uses IMU feedback to satisfy Eq. 3 with minimal steering angle change. To handle non-linearities, we use feedback linearization before applying LQR to this system. Assume that for satisfying Eq. 3, the small change in lateral acceleration is ΔA_y, then (from 5) the required change in steering angle can be found from the lateral acceleration as:

$$\Delta\delta = \Delta A_y (\cos^2(\delta) L_{fr}/V_w^2) \tag{6}$$

Observe that this approach would continue to adjust the steering angle until ΔA_y goes to 0, regardless of tire slip, which weakens the dependence on the "no-slip" assumption. The LQR controller then provides a setpoint for the ΔA_y to regulate both the lateral acceleration error and the roll rate. The state transition equation, state penalty and control penalty



(a) Beam-node based physics simulation (b) Elevation map replication for off-road map (c) Isolated rollover setting on flat ground

Fig. 4: BeamNG's soft-body physics simulation (4(a)) allows accurate simulation of second-order inertial effects, useful for problems such as rollover prevention(4(c)). The integration with BeamNG spoofs the elevation map(4(b)) for the off-road environment to isolate the control problem.

matrices, assuming Δt is the update time, are given by:

$$\begin{bmatrix} \frac{A_y^b}{A_z^b} - RI_L \\ \omega_x \end{bmatrix}_{t+1} = \begin{bmatrix} 1 & 0 \\ K & 1 \end{bmatrix} \begin{bmatrix} \frac{A_y^b}{A_z^b} - RI_L \\ \omega_x \end{bmatrix}_t + \begin{bmatrix} 1 \\ K \end{bmatrix} \Delta \frac{A_y^b}{A_z^b},$$
$$K = \Delta t A_z^b \frac{H_{com}}{J_x/m}, \quad Q = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}, \quad R = 1 \tag{7}$$

Where J_x, m refer to the roll moment of inertia and mass respectively. Note that in practice additional constraints may be used in Eq. 3 to reduce A_y^c for safety reasons.

E. Simulator interface

We use BeamNG [5], a high-fidelity soft-body physics simulator geared towards mobile systems, that simulates physical systems as a collection of beams and nodes (see Fig. 4(a)). BeamNG has been used in the industry, as well as in academia [6, 31]. We build our SITL and HITL stack around this simulator. The SITL can replicate the elevation map (see Fig. 4(b)), useful for isolating control problems and benchmarking elevation mapping. HITL is used for testing the software directly on the SBC (see Fig. 2). For both the HITL and SITL, the simulator can run on a separate computer connected by WiFi/Ethernet, allowing accurate estimation of compute load on the SBC. Note that our interface can be used for studying off-road autonomy in general, though we only use two particular vehicles for our experiments (see 4(b)).

IV. Evaluation

A. Experimental evaluation of simulator accuracy

We evaluate how well the simulator predicts real-world behavior by comparing its dynamics prediction errors against the errors of two simpler mathematical models. For simplicity, we perform this evaluation on a flat surface.

Hypothesis H1: The simulator's error is <= error for either of the other models for all metrics.

Scenario: In the real world, 3 trajectories, totaling $\approx 100s$ of data, are collected on an outdoor, flat tiled surface at 50Hz. Using IMU data, the coefficient of friction is estimated to be $\approx 0.65 \pm 0.1$. Car specifications were measured directly. The car is moved in circular patterns at speeds up to $\approx 8m/s$, and accelerations up to $\approx 7m/s^2$. The dynamics model is initialized with real data only at the start of a new control sequence.

Metrics: We calculate L2 norm errors for body frame acceleration, rotation rate, and velocity, normalizing them by the largest error among models to allow relative comparison.

Models:

- BeamNG: represents the simulator's error.
- slip3d: represents the error for the model described in III-C1.
- noslip3d: represents the error for a no-slip kinematic bicycle model [28], with its velocity projected into 3D using the elevation map, as also done by Meng et al. [33].

Result: Fig. 6 shows that the simulator's error is slightly lower than the next best in Velocity and rotation rate, and much lower in Acceleration($p < 0.02$ for all). Note that all error bars represent 95% confidence intervals. Thus, hypothesis **H1 is confirmed**.

B. Validating RPS's utility as a safety system

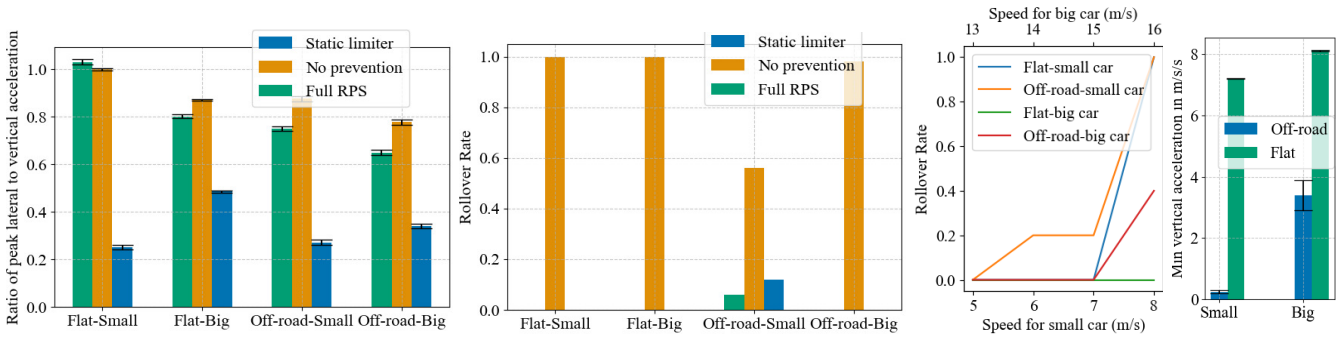
We evaluate the utility of the RPS by evaluating how much it restricts the vehicle's maneuverability, and how much it prevents rollovers. To stress the RPS, we set the tire friction for the small car to be 50% more than the default and exaggerate the roll characteristics through suspension tuning. Note that the simulator automatically reduces friction for off-road settings by $\approx 20\%$. This, combined with the bumpiness of off-road terrain makes wheel-slip more likely in the off-road settings.

1) Isolated evaluation of rollover prevention: We test the RPS by forcing the vehicle into a rollover by applying maximum steering to one side while moving at a fixed speed **V** (see Fig. 4(c)).

Hypothesis H2: The Full RPS(III-D) achieves a greater ratio of peak A_y^b/A_z^b than the static limiter(III-D1) while having the same or lower rollover rate.

Scenarios: We use the "Small car" and the "Big car"(see Fig. 4(a)), over two terrains, Flat(Fig. 4(c)) and Off-road(Fig. 4(b)), for 50 iterations each. Fig. 4(c) describes the setting, where **V** is increased linearly with each iteration, from $[4.8, 7.2]m/s$ for the small car, and $[9.6, 14.4]m/s$ for the big car. We also test the RPS's failure rate over a broader range of speeds, for 5 iterations at 4 speeds for each scenario.

Metric: We use the rollover rate and the peak ratio of lateral to vertical acceleration achieved once the car begins turning, up to a roll angle of 90 degrees.



(a) Peak A_y^b/A_z^b (b) Rollover rate (c) Rollover rate vs speeds (d) Min. A_z^b

Fig. 5: Static limiter obtains lower A_y^b/A_z^b in 5(a) and more rollovers(5(b)). In 3 out of 4 scenarios, the rollover rate is 0 for both static limiter and full RPS in 5(b). 5(c), 5(d) show the breaking of instantaneous steering and constant ground contact assumption.

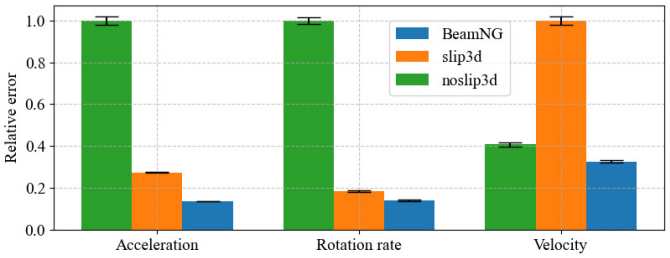


Fig. 6: BeamNG performs as well as if not better than other models.

HLC with	Scenario	$TTC_p(s)$	$TTC_{np}(s)$	$\alpha_z(rad/s^2)$	$R.O._{avg}$
RPS OFF	Tight	11.04± 1.56	8.76± 1.47	207.74± 157.77	2.28
RPS ON	Tight	**8.39**± 0.13	**8.39**± 0.13	**170.60**± 18.45	**0**
RPS OFF	Shallow	17.63± 0.76	17.47± 0.59	210.05± 86.92	0.16
RPS ON	Shallow	**17.07**± 0.15	**17.07**± 0.15	**117.53**± 14.28	**0**

TABLE III: HLC w/ RPS ON performs at least as well as if not better than HLC w/ RPS OFF and makes the trajectories smoother, indicated by the lower α_z

Algorithms: We compare "No prevention", "Static limiter" (III-D1) with δ_s set to 0 and "Full RPS" (III-D) with δ_s set to 30 % of the maximum steering angle. It should be noted that as platforms apart from ours shown in table I do not posses RPS, they are represented by the "No prevention" algorithm in these experiments.

Result: Across all scenarios, full RPS achieves at least 83% of the ratio achieved with no prevention. Fig. 5(a) and Fig 5(b) show that the full RPS obtains a higher ratio of A_y^b/A_z^b and a lower rollover rate when than the static limiter($p < 0.02$ for all). Thus, hypothesis **H2 is confirmed**.

2) In the loop evaluation of rollover prevention: We test the RPS (III-D) in conjunction with the high-level controller **(HLC)** (III-C). To simulate dynamics mismatch, we set the friction coefficient used by the model to be 33% lower than its value set in simulation. **Scenarios:** We use the small car on flat ground, with two trajectories, for 50 iterations. The first forces hard turning to reach the end, increasing rollover probability (Tight), and the second provides shallow turns and long straights (Shallow). We randomly add noise to the start location for each iteration. Upon rollover, the vehicle is reset to the closest location on the target trajectory, and given a 1-second penalty to reflect the delay for resetting the vehicle in the real world. This is an optimistic comparison, as in the real world, rollovers can be fatal to the hardware.

Hypothesis H3: The Full RPS system does not excessively interfere with the performance of the system.

Metric: We measure the time to reach the final goal with the 1-second penalty (TTC_p) and without it (TTC_{np}), maximum yaw acceleration just before rollover α_z, and average number of rollovers ($R.O._{avg}$) for an iteration.

Algorithms: We compare the HLC(III-C) running without RPS ("RPS OFF") against running it with RPS ("RPS ON"). It should be noted that "RPS OFF" represents the approach of preventing rollovers by incorporating the rollover index into the cost function of a model predictive controller, as in [29, 33].

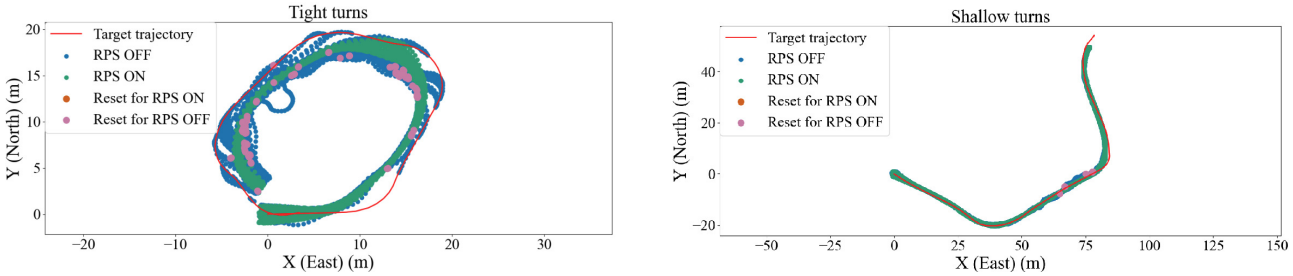
Result: Table III, shows that the completion time TTC_p is lower with the RPS ON for the Tight turns scenario, and slightly lower for the Shallow turns scenario($p < 0.02$ for both). Thus, RPS does not interfere with HLC unless necessary, **confirming hypothesis H3**. The TTC_{np} in table III also shows that HLC with RPS ON is at least as good as if not slightly better even when not considering the 1-second penalty, from which we infer that using RPS does not jeopardize the performance of the system.

C. Real world Evaluation

Scenario: For real-world evaluation we only run the system with the RPS "ON", with $\delta_s = 0.3$, in the manual mode as well as the autonomous mode using the HLC(III-C), using the default MPPI controller for waypoint following, with **no repairs** between runs.

Hypothesis H4: The complete stack survives high lateral accelerations ($> 8m/s^2$) and speeds ($> 5m/s$) over a large distance($> 10km$) with minimal part damage.

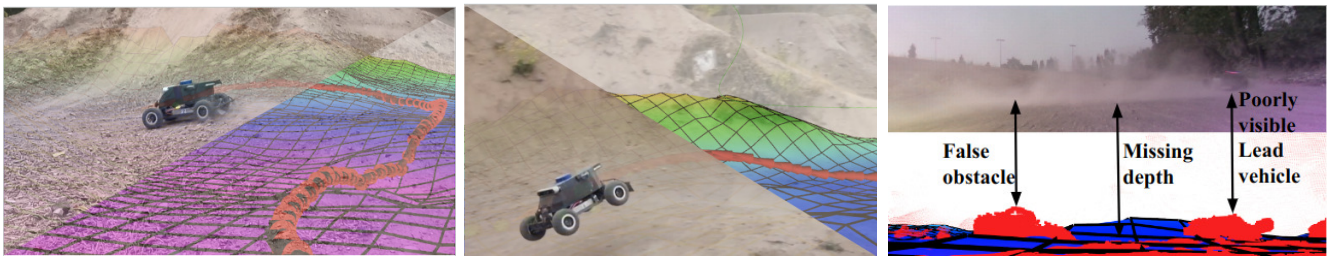
Metrics: We measure the "peak" lateral acceleration $P(A_y^b)$, speed $P(V_x^b)$, delayed vertical acceleration just before rollover $D(A_z^b)$, number of rollovers ($R.O.$), and distance covered (S). The Peak values represent the top 99.7th percentile of the data, corresponding to $\approx$ 13 seconds. Note that the acceleration and speed values are low-pass filtered with a cutoff frequency of 2 Hz to remove large spikes in



(a) Trajectory plots for tight-turns test case

(b) Trajectory plots for shallow-turns test case

Fig. 7: RPS reduces rollovers, not only when the vehicle needs to turn harder(Fig 7(a)) but also when the vehicle might rollover incidentally due to yaw instability from bumps in an off-road environment, indicated by the waviness of the RPS OFF trajectory and the high yaw accelerations α_z in Table III.



(a) Off-road driving on uneven terrain

(b) Airborne situations

(c) Sensor degradation, occlusion from dust

Fig. 8: High speed offroading 8(a)) often results in airborne situations(8(b)). Dust trails in multi-agent settings (8(c)) present interesting challenges.

Mode	$P(A_y^b)(m/s^2)$	$P(V_x^b)(m/s)$	$R.O.$	$D(A_z^b)(m/s^2)$	S(Km)
Auto	7.41	6.95	2	3.29	≈12
Manual	8.92	8.01	1	0.75	≈38

TABLE IV: Low $D(A_z^b)$ indicates weightlessness or loss of ground contact

accelerations. An R.O. is when the roll angle $> 1.0\text{rad}$ while the vehicle speed $> 0.5m/s$. Delayed values are the values 0.2 seconds before the R.O.

Result: Real world experiments over ≈50Km (Table IV) show that the complete stack obtains speeds $> 5m/s$, lateral accelerations above $> 8m/s^2$, and 3 rollovers (R.O.). The rollovers damage only a sacrificial, easy-to-replace LiDAR mount in the last run (see Appendix VII-B). Over the 50 kilometers, the mean and standard deviation of the speed was $(2.18, 2.38)m/s$, and the mean and standard deviation of the lateral acceleration was $(2.49, 2.57)m/s^2$. Thus, hypothesis **H4 is confirmed**. For further qualitative assessment of our real-world experiments, we refer the reader to the publicly released dataset.

V. Discussion

Autonomy stack: The autonomy stack presented in this work provides a useful reference point that combines relatively mature state-of-the-practice methods. Considering the low inertia of the system, we aimed for a sub-50 ms perception-to-control latency in the autonomy stack(see Fig.2). Due to SWaP constraints, this precluded using additional capabilities, such as semantic segmentation [37], as well as the use of learned dynamics models [29] without increasing the latency or memory requirements. We believe, however, that a standardized platform such as this can make it easier for researchers to improve individual sub-components over time, such that additional capabilities become available eventually.

Choice of simulator and simulation fidelity: While many good simulators for robotics exist today, we chose BeamNG as it is geared specifically towards simulating vehicles, and is used by the industry for the same. It is geared towards simulating vehicle damage, which is particularly useful for inferring whether an off-road autonomy stack is safe or not, without having to create hand-crafted constraints on velocities, and accelerations, which can create misaligned incentives if the simulator is being used for training a reinforcement learning agent. We do not preclude future extensions that allow the use of other simulators.

We only verify simulation fidelity for operation on flat surfaces. Replication of real-world elevation maps, their integration into the simulator, and the acquisition of precise GPS coordinates to obtain more accurate benchmarks can be addressed by future work.

Reducing cost and increasing accessibility In Table I, we compare the cost of a 1/5th scale platform (AutoRally [21]) with our system that uses a 1/10th scale platform. While a bigger vehicle would be more expensive, note that the cost of sensing and computing is not as closely tied to the size, and is higher due to the use of industrial-grade components.

We find from 50 kilometers of real-world testing that low-cost commercial-grade hardware can indeed be used for such a platform. We believe that this is at least in part made possible through the use of the RPS safety mechanism. While a few rollovers would not cause significant damage to our platform, a rollover every 100 meters of travel would incur 500 rollovers for 50 kilometers, requiring repeated repairs, and incurring a greater cost over time than using ruggedized hardware.

Notes on rollover prevention: It should be noted that in the IV-B2 experiment, while the high-level controller could use either adaptive control methods or online sys-ID as in [4]

to address a dynamics mismatch, it is introduced on purpose to demonstrate how the RPS can help in the case of a faulty high-level controller. From Table III, it can be seen that the dynamics mismatch causes the HLC to apply larger than necessary inputs that result in higher yaw acceleration($p < 0.02$). In contrast, corrections from RPS smoothen out the trajectories (see Fig. 7). In contrast to the isolated setting(see Fig. 5(c)), in the closed loop experiments there are no rollovers (see table III) despite the peak speeds exceeding 8.0 m/s for both scenarios, as the HLC does not provide unreasonable commands to cause rollovers on purpose.

In our isolated experiments, we also push the system to unreasonably high speeds to understand the limits of the safety mechanism. The unevenness of off-road surfaces results in airborne (Fig. 8(b)) situations, apparent from the lower vertical acceleration(Fig. 5(d)), breaking the assumption of constant ground contact – broken more often in the real world(Table IV, Fig. 8(b)). To the best of our knowledge, autonomous mid-air control of cars is currently an open problem that could be addressed by future work.

In Fig5(a), it can also be seen that the lateral acceleration ratio for the small car on the flat surface is slightly ($\approx 3\%$) higher when using RPS as compared to when not using any prevention. Note that due to the vehicle's suspension, the body of the vehicle can roll slightly even before RI_L is reached. Due to an effect known as "jacking"[18] the center of mass moves upwards relative to the outside tire's contact point due to this body roll. Once the center of mass moves upwards, the RI_L(see Eq. 4) becomes smaller, creating a positive feedback loop; requiring less lateral traction to sustain the rollover. This happens much faster on the smaller vehicle owing to its lower roll inertia (≈ 3.5 times faster on average). When a rollover is prevented, the center of mass is kept low. This allows the system to sustain slightly higher lateral accelerations. Note that we consider this to be an unintended effect of the rollover prevention system – not something that the system was designed to do.

Sensor degradation: During our field experiments, we observed that in dry off-road environments, apart from sensor degradation from dust coverage [10], dust trails from a lead vehicle can cause issues such as false depth or no depth (see 8(c). Depending on the density of the dust particles, this may obstruct lidar as well, which can create interesting situations for multi-agent off-road navigation problems that future work could address.

Licensing: The HOUND software stack is built using multiple open-source components, with either MIT, BSD-3, or GPL-v3 licenses. The BeamNG simulator is an exception to this. While the HOUND-BeamNG integration is open-source, the simulator itself is not. However, at the time of writing, BeamNG is available for academic purposes at no cost.

VI. Conclusions and Future Work

The paper presents an inexpensive, 1/10th scale research platform for **H**igh-speed **O**ff-road **U**nder-actuated **N**onholonomic **D**riving(**H.O.U.N.D.**). HOUND integrates state-of-the-art methods for terrain mapping, localization, and model predictive control geared towards aggressive offroad driving into a single platform. The platform is integrated with a high-fidelity vehicle simulator, BeamNG, to allow both software and hardware-in-the-loop testing. We deploy HOUND in the real world, at high speeds, on four different terrains covering 50 km of driving, and highlight the utility of rollover prevention for traversing difficult terrain at high speed through real-world experiments.

At the moment, the perception system uses elevation maps to represent the environment. This requires less computation than a 3D representation and is convenient for predicting vehicle dynamics. However, the assumptions imposed on the perception and dynamics models make them a lossy compression of the real world. For instance, an elevation map might consider a fallen log of wood as a hill, leading to errors in physics prediction, which may lead to errors in navigation. Future work may explore how this problem may be addressed in a compute and memory-efficient manner for real-time operation on such robots. Smaller robots are safer and easier to operate, which makes dataset generation easier compared to using full-size vehicles. Future work should investigate the re-targeting of datasets collected on small platforms to their use on large and full-size platforms.

Acknowledgements

This work was (partially) funded by the National Science Foundation NRI (#2132848) and CHS (#2007011), DARPA RACER (#HR0011-21-C-0171), the Office of Naval Research (#N00014-17-1-2617-P00004 and #2022-016-01 UW), and Amazon.

Authors are with the Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle, WA, USA. Email: {sidtalia, schmttle, cmavro, spitzer, lambert6, siddh }@cs.washington.edu.

References

[1] Manuel Acosta, Stratis Kanarachos, and Michael E Fitzpatrick. A hybrid hierarchical rally driver model for autonomous vehicle agile maneuvering on loose surfaces. In *ICINCO (2)*, pages 216–225, 2017.

[2] Ardupilot. Ardupilot, 2009.

[3] Mansour Ataei, Amir Khajepour, and Soo Jeon. Model predictive rollover prevention for steer-by-wire vehicles with a new rollover index. *International Journal of Control*, 93(1):140–155, 2020.

[4] Lucas Barcelos, Alexander Lambert, Rafael Oliveira, Paulo Borges, Byron Boots, and Fabio Ramos. Dual online stein variational inference for control and dynamics. *arXiv preprint arXiv:2103.12890*, 2021.

[5] BeamNG GmbH. BeamNG.tech, 2022. URL https://www.beamng.tech/.

[6] Christian Birchler, Nicolas Ganz, Sajad Khatiri, Alessio Gambi, and Sebastiano Panichella. Cost-effective simulation-based test selection in self-driving cars software with SDC-Scissor. In *2022 IEEE international conference on software analysis, evolution and reengineering (SANER)*, pages 164–168. IEEE, 2022.

[7] Carrie G Bobier and J Christian Gerdes. Staying within the nullcline boundary for vehicle envelope control using a sliding surface. *Vehicle System Dynamics*, 51(2):199–217, 2013.

[8] Rogerio Bonatti, Sai Vemprala, Shuang Ma, Felipe Frujeri, Shuhang Chen, and Ashish Kapoor. Pact: Perception-action causal transformer for autoregressive robotics pre-training. *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[9] Xiaoyi Cai, Michael Everett, Jonathan Fink, and Jonathan P How. Risk-aware off-road navigation via a learned speed distribution map. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2931–2937. IEEE, 2022.

[10] Daniel W. Carruth, Clayton T. Walden, Christopher Goodin, and Sara C. Fuller. Challenges in Low Infrastructure and Off-Road Automated Driving. In *2022 Fifth International Conference on Connected and Autonomous Driving (MetroCAD)*, pages 13–20, 2022. doi: 10.1109/MetroCAD56305.2022.00008.

[11] Bo-Chiuan Chen and Huei Peng. Differential-braking-based rollover prevention for sport utility vehicles with human-in-the-loop evaluations. *Vehicle system dynamics*, 36(4-5):359–389, 2001.

[12] Jesse Chintanadilok, Sahaj Patel, Yilin Zhuang, and Aditya Singh. Mission planner: An open-source alternative to commercial flight planning software for unmanned aerial systems: Ae576/ae576, 8/2022. *EDIS*, 2022(4), 2022.

[13] Mark Cutler, Thomas J Walsh, and Jonathan P How. Reinforcement learning with multi-fidelity simulators. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3888–3895. IEEE, 2014.

[14] Aniket Datar, Chenhui Pan, Mohammad Nazeri, and Xuesu Xiao. Toward Wheeled Mobility on Vertically Challenging Terrain: Platforms, Datasets, and Algorithms. *arXiv preprint arXiv:2303.00998*, 2023.

[15] Moustapha Doumiati, Aa Victorino, Ali Charara, and Daniel Lechner. Lateral load transfer and normal forces estimation for vehicle safety: experimental test. *Vehicle System Dynamics*, 47(12):1511–1533, 2009.

[16] Tom Duckett, Simon Pearson, Simon Blackmore, Bruce Grieve, Wen-Hua Chen, Grzegorz Cielniak, Jason Cleaversmith, Jian Dai, Steve Davis, Charles Fox, et al. Agricultural Robotics: The Future of Robotic Agriculture 2018. *URL https://uwe-repository. worktribe. com/output/866226*, 1806.

[17] Thomas Fork, H Eric Tseng, and Francesco Borrelli. Models for ground vehicle control on nonplanar surfaces. *Vehicle System Dynamics*, pages 1–25, 2023.

[18] MB Gerrard. Roll centres and jacking forces in independent suspensions-a first principles explanation and a designer's toolkit. Technical report, SAE Technical Paper, 1999.

[19] Jason Gibson, Bogdan Vlahov, David Fan, Patrick Spieler, Daniel Pastor, Ali-akbar Agha-mohammadi, and Evangelos A Theodorou. A Multi-step Dynamics Modeling Framework For Autonomous Driving In Multiple Environments. *arXiv preprint arXiv:2305.02241*, 2023.

[20] Jonathan Y Goh, Tushar Goel, and J Christian Gerdes. Toward automated vehicle control beyond the stability limits: drifting along a general path. *Journal of Dynamic Systems, Measurement, and Control*, 142(2), 2020.

[21] Brian Goldfain. *Autonomous rally racing with AutoRally and model predictive control*. PhD thesis, Georgia Tech, 2019.

[22] Karl Iagnemma, Adam Rzepniewski, Steven Dubowsky, and Paul Schenker. Control of robotic vehicles with actively articulated suspensions in rough terrain. *Autonomous Robots*, 14(1):5–16, 2003.

[23] USB Intel. 3.0* Radio Frequency Interference Impact on 2.4 GHz Wireless Devices. *White Paper, Apr*, 2012.

[24] Achin Jain, Matthew O'Kelly, Pratik Chaudhari, and Manfred Morari. BayesRace: Learning to race autonomously using prior experience. *arXiv preprint arXiv:2005.04755*, 2020.

[25] Azarakhsh Keipour, Mohammadreza Mousaei, and Sebastian Scherer. Automatic real-time anomaly detection for autonomous aerial vehicles. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5679–5685. IEEE, 2019.

[26] Nima Keivan and Gabe Sibley. Realtime simulation-in-the-loop control for agile ground vehicles. In *Conference Towards Autonomous Robotic Systems*, pages 276–287. Springer, 2013.

[27] Taekyung Kim, Gyuhyun Park, Kiho Kwak, Jihwan Bae, and Wonsuk Lee. Smooth model predictive path integral control without smoothing. *IEEE Robotics and Automation Letters*, 7(4):10406–10413, 2022.

[28] Jason Kong, Mark Pfeiffer, Georg Schildbach, and Francesco Borrelli. Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE intelligent vehicles symposium (IV)*, pages 1094–1099. IEEE, 2015.

[29] Hojin Lee, Taekyung Kim, Jungwi Mun, and Wonsuk Lee. Learning Terrain-Aware Kinodynamic Model for Autonomous Off-Road Rally Driving With Model Predictive Path Integral Control. *arXiv preprint arXiv:2305.00676*, 2023.

[30] Hojin Lee, Junsung Kwon, and Cheolhyeon Kwon. Learning-based Uncertainty-aware Navigation in 3D Off-Road Terrains. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10061–10068. IEEE, 2023.

[31] Alexander Lehner, Stefano Gasperini, Alvaro Marcos-Ramiro, Michael Schmidt, Mohammad-Ali Nikouei Mahani, Nassir Navab, Benjamin Busam, and Federico Tombari. 3D-VField: Adversarial augmentation of point clouds for domain generalization in 3D object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17295–17304, 2022.

[32] Lorenz Meier, Petri Tanskanen, Friedrich Fraundorfer, and Marc Pollefeys. Pixhawk: A system for autonomous flight using onboard computer vision. In *2011 ieee international conference on robotics and automation*, pages 2992–2997. IEEE, 2011.

[33] Xiangyun Meng, Nathan Hatch, Alexander Lambert, Anqi Li, Nolan Wagener, Matthew Schmittle, JoonHo Lee, Wentao Yuan, Zoey Chen, Samuel Deng, et al. TerrainNet: Visual Modeling of Complex Terrain for High-speed, Off-road Navigation. *arXiv preprint arXiv:2303.15771*, 2023.

[34] Takahiro Miki, Lorenz Wellhausen, Ruben Grandia, Fabian Jenelten, Timon Homberger, and Marco Hutter. Elevation mapping for locomotion and navigation using gpu. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2273–2280. IEEE, 2022.

[35] MIT RACECAR. The MIT RACECAR, 2016. Accessed: 2019-08-01.

[36] Peter Mitchell, Reuben O'Brien, and Minas Liarokapis. On the Development of Waterjet-Powered Robotic Speedboats: An Open-Source, Low-Cost Platform for Education and Research. In *2022 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 153–159, 2022. doi: 10.1109/SSRR56537.2022.10018662.

[37] Huy-Hung Nguyen, Duong Nguyen-Ngoc Tran, and Jae Wook Jeon. Real-Time Semantic Segmentation on Edge Devices with Nvidia Jetson AGX Xavier. In *2022 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, pages 1–4. IEEE, 2022.

[38] Matthew O'Kelly, Varundev Sukhil, Houssam Abbas, Jack Harkins, Chris Kao, Yash Vardhan Pant, Rahul Mangharam, Dipshil Agarwal, Madhur Behl, Paolo Burgio, and Marko Bertogna. F1/10: An Open-Source Autonomous Cyber-Physical Platform, 2019.

[39] Yunpeng Pan, Ching-An Cheng, Kamil Saigol, Keuntaek Lee, Xinyan Yan, Evangelos Theodorou, and Byron Boots. Agile autonomous driving using end-to-end deep imitation learning. *arXiv preprint arXiv:1709.07174*, 2017.

[40] Evangelos Papadopoulos and Daniel A Rey. The force-angle measure of tipover stability margin for mobile manipulators. *Vehicle System Dynamics*, 33(1):29–48, 2000.

[41] Patrick Pfaff, Rudolph Triebel, and Wolfram Burgard. An Efficient Extension to Elevation Maps for Outdoor Terrain Mapping and Loop Closing. *The International Journal of Robotics Research*, 26(2):217–230, 2007.

[42] Mark E Pittelkau. Rotation vector in attitude estimation. *Journal of Guidance, Control, and Dynamics*, 26(6):855–860, 2003.

[43] Xingguo Qian, Chunyan Wang, and Wanzhong Zhao. Rollover prevention and path following control of integrated steering and braking systems. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 234(6):1644–1659, 2020.

[44] Fabrice Reclus and Kristen Drouard. Geofencing for fleet & freight management. In *2009 9th International Conference on Intelligent Transport Systems Telecommunications,(ITST)*, pages 353–356. IEEE, 2009.

[45] Selim Solmaz, Martin Corless, and Robert Shorten. A methodology for the design of robust rollover prevention controllers for automotive vehicles with active steering. *International Journal of Control*, 80(11):1763–1779, 2007.

[46] Siddhartha S. Srinivasa, Patrick Lancaster, Johan Michalove, Matt Schmittle, Colin Summers, Matthew Rockett, Joshua R. Smith, Sanjiban Chouhury, Christoforos Mavrogiannis, and Fereshteh Sadeghi. MuSHR: A Low-Cost, Open-Source Robotic Racecar for Education and Research. *CoRR*, abs/1908.08031, 2019.

[47] Kyle Stachowicz, Dhruv Shah, Arjun Bhorkar, Ilya Kostrikov, and Sergey Levine. FastRLAP: A System for Learning High-Speed Driving via Deep RL and Autonomous Practicing. *arXiv preprint arXiv:2304.09831*, 2023.

[48] Maximilian Stölzle, Takahiro Miki, Levin Gerdes, Martin Azkarate, and Marco Hutter. Reconstructing occluded elevation information in terrain maps with self-supervised learning. *IEEE Robotics and Automation Letters*, 7(2):1697–1704, 2022.

[49] John K Subosits and J Christian Gerdes. Impacts of model fidelity on trajectory optimization for autonomous vehicles in extreme maneuvers. *IEEE Transactions on Intelligent Vehicles*, 6(3):546–558, 2021.

[50] Tomoji Takasu and Akio Yasuda. Evaluation of RTK-GPS performance with low-cost single-frequency GPS receivers. In *Proceedings of international symposium on GPS/GNSS*, pages 852–861, 2008.

[51] Sidharth Talia, Arnav Thareja, Christoforos Mavrogiannis, Matt Schmittle, and Siddhartha S Srinivasa. PuSHR: A Multirobot System for Nonprehensile Rearrangement. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.

[52] Davide Tavernini, Matteo Massaro, Efstathios Velenis, Diomidis I Katzourakis, and Roberto Lot. Minimum time cornering: the effect of road surface and car transmission layout. *Vehicle System Dynamics*, 51(10):1533–1547, 2013.

[53] Andrew Thoesen and Hamid Marvi. Planetary surface mobility and exploration: A review. *Current Robotics Reports*, 2(3):239–249, 2021.

[54] Willy Tiengo, Evandro B Costa, and Joseana M Fechine. Reducing risk of rollover in curve for heavy-duty vehicles with an agent-based advanced driver assistance system. In *2016 IEEE International Conference on Computer and Information Technology (CIT)*, pages 65–72. IEEE, 2016.

[55] Antonio Tota, Luca Dimauro, Filippo Velardocchia, Genny Paciullo, and Mauro Velardocchia. An Intelligent Predictive Algorithm for the Anti-Rollover Prevention of Heavy Vehicles for Off-Road Applications. *Machines*,

10:835, Sep 2022. ISSN 2075-1702.

[56] Vasilios Tsourapas, Damrongrit Piyabongkarn, Alexander C Williams, and Rajesh Rajamani. New method of identifying real-time predictive lateral load transfer ratio for rollover prevention systems. In *2009 American control conference*, pages 439–444. IEEE, 2009.

[57] Rodrigo Ventura and Pedro U Lima. Search and rescue robots: The civil protection teams of the future. In *2012 Third International Conference on Emerging Security Technologies*, pages 12–19. IEEE, 2012.

[58] Georg Weber, D Dettmering, and H Gebhard. Networked transport of RTCM via internet protocol (NTRIP). In *A Window on the Future of Geodesy: Proceedings of the International Association of Geodesy IAG General Assembly Sapporo, Japan June 30–July 11, 2003*, pages 60–64. Springer, 2005.

[59] Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M Rehg, Byron Boots, and Evangelos A Theodorou. Information theoretic MPC using neural network dynamics. In *30th Conference on Neural Information Processing Systems*, 2016.

[60] Rama K Yedavalli and Hsun-Hsuan Huang. Controller design for multi-body ground vehicle rollover prevention using modified LQR framework. In *Dynamic Systems and Control Conference*, volume 43352, pages 931–938, 2008.

VII. Appendix

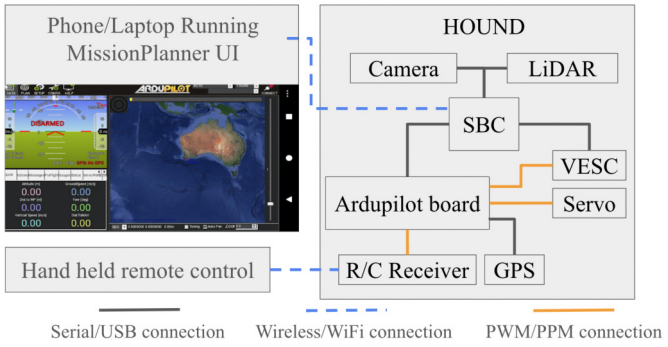


Fig. 9: Illustration describing how the components are connected to each other

A. Utilising the Ardupilot ecosystem:

The Ardupilot ecosystem provides many plug-and-play features that we take advantage of in this work. It takes care of time synchronization between itself and the SBC, as well as provides certain safety features for field operations.

As the Ardupilot board acts as a bridge between the SBC and the actuators, it allows the operator to always be in control of the final commands being sent to the hardware. For instance, the ardupilot board can "disarm" the vehicle under different circumstances by setting the motor command to "0", preventing the car from moving of its own volition or hitting the brakes if it is currently moving. The board can be, and by default is, configured to disarm the vehicle either through a two-position switch on the hand-held remote, or by the loss of radio contact with said remote. It can additionally be configured to disarm if the vehicle moves outside of a particular region, generally known as "geofencing" [44], or if the vehicle loses the GPS signal completely during autonomous operation. Note that in the case of GPS loss, operation in manual modes is still possible.

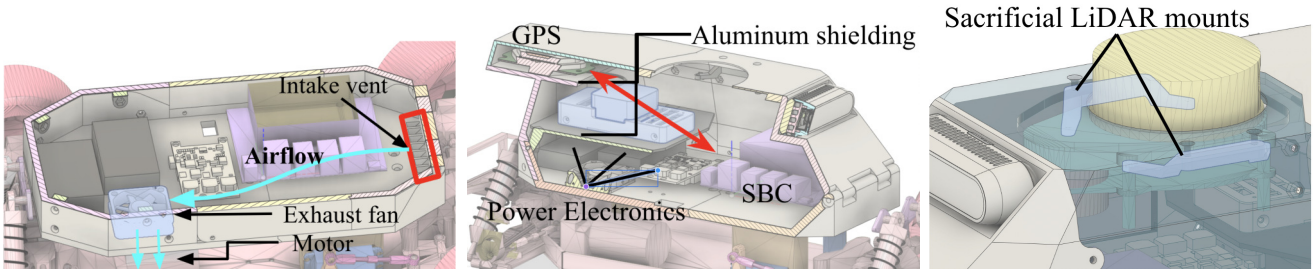
When an internet connection is available to the SBC, RTCM [50] corrections can be obtained using NTRIP [58], which can be forwarded to the GPS by the ardupilot board to obtain high-accuracy RTK positioning. During field operations, the MissionPlanner [12] UI running either on the user's laptop or phone (see Fig 9), connected to the SBC over WiFi, can be used to calibrate the IMU and compass in the ardupilot board, change configuration parameters, or reboot it. The UI also provides a convenient way of sending GPS waypoints to the SBC, which can then be used by the high-level controller, or a local planner. Additionally, the onboard buzzer on the ardupilot boards is used for producing notification sounds that let the user know the internal status of the sub-systems. For instance, the notification sounds can let the user know if the perception system has started running yet or not, or if the system has started or stopped recording data. If the user only wishes to collect data from the platform with manual driving, the perception stack and low-level controller can be configured to auto-boot, and the user can perform this task without needing to carry a laptop with them to monitor the vehicle's internal status.

B. Hardware design decisions

Heat management: To protect the system from environmental factors as well as collisions, most of the hardware is placed in an enclosed shell (see Fig 3). We use an active negative pressure cooling system, where the fan creates a slight negative pressure in the shell, and routes the air around all the components before exhaustion (see Fig 10(a)). The exhaust goes over the motor, extracting the motor's heat as well. While the exhaust air is hotter than the ambient, it is still much cooler than the motor's temperature and can provide the necessary airflow for sustained cooling. The intake vent is positioned and sized to minimize the entry of dust and dirt. While the default setup uses negative air pressure, a positive pressure system would work as well, requiring a dust filter under the fan and flipping the orientation of the fan.

Electromagnetic interference management: The SBC, devices that use USB-3.0 protocol, and power converters, all produce electromagnetic interference(**EMI**) that affects the GPS's accuracy [23]. On big platforms, the spacing between the GPS and such EMI-generating devices can be large, and so this may or may not be a concern for them. On the HOUND, we minimize the impact of the EMI on the GPS by one, maximizing the separation between the SBC and the GPS, and two, by placing two layers of Aluminum shielding between it and the power-electronics (see Fig 10(b)). The Aluminum shielding is created using Aluminum tape, readily available from any hardware store, or online.

Sacrifical LiDAR mounts: As we show in our experiments, rollover prevention is not without limitations. Eventually, the vehicle is bound to roll over. The LiDAR becomes the most



(a) Heat management using negative pressure (b) Managing electromagnetic interference. (c) Sacrificial LiDAR mounts

Fig. 10: Small enclosures create challenges for heat management 10(a) and electromagnetic interference 10(b). Sacrificial LiDAR mounts absorb the impact energy during incidental rollovers to reduce damage to the LiDAR.

susceptible sensor during a rollover as it protrudes significantly out of the body shell. If the LiDAR were to be mounted directly onto the roof, one of two things could happen, either the LiDAR survives but the roof caves in, or the roof survives but the LiDAR is damaged. The mid-section, on which the LiDAR is mounted, can take at least 10 hours to print, and the LiDAR costs at least $100. An alternate design choice would be to create a "roll cage" around the lidar, however, that would now obstruct the LiDAR's view. To avoid damage to either we use sacrificial mounts to attach the LiDAR to the roof, designed to be strong enough to survive low-speed rollovers but to absorb energy and break on high-speed impacts (see Fig 10(c)). These parts can be printed much faster and can reduce damage to the LiDAR.

C. Low Level Controller

The Low-Level Controller is responsible for sending the motor control and steering control commands to the Ardupilot board, where they are converted to appropriate pulse-width-modulated signals. It operates in two modes – manual and autonomous, where the RPS is "ON" by default in both. In the manual mode, the low-level controller listens to control commands from the user's hand-held remote control (9). In the autonomous mode, it listens to the control commands from the high-level controller, however, the user's throttle input corresponds to the wheel speed limit. For instance, if the operator visually infers that a collision is about to occur, and the vehicle is in autonomous mode, the user can reduce the speed limit to prevent the collision. Alternatively, the user can slowly increase the speed limit as they gain confidence in their custom controller. This speed limit is also published on a separate ROS topic should the high-level controller need it. For instance, in our implementation of the MPPI, we change the sampling range according to the speed limit imposed by the user, such that wheel speeds above this limit are not sampled.

wheelspeed control: The wheel speed control takes as input the wheel speed target V_w^* and produces a duty cycle signal for the motor using a closed-loop PI controller with a feed-forward term. Here, the current wheel speed V_w is obtained from the VESC.

$$D = \frac{E_i}{E_n} K_f V_w^* + K_p(V_w^* - V_w) + K_i \int_0^t (V_w^* - V_w)\, d\tau \quad (8)$$

Where D refers to the motor duty cycle, E_i, E_n represent the instantaneous and nominal battery voltage, K_f, K_p, K_i refer to the feedforward gain, proportional gain, and integral gain. The feed-forward term's gain is obtained by measuring the ratio of the duty cycle to wheel speed on a flat tarmac surface for a small range of speeds, and the gains for the proportional and integral terms are tuned such that changes in slope or surface can be dealt with in real-time.

Note on wheel speed measurement: To simplify the sensing apparatus, we assume that all the wheels rotate at the same wheel speed. We find that this assumption works well for high-speed driving ($> 2m/s$) on uneven, smooth terrain, but begins to break down when dealing with extremely rough terrain, such as in rock-crawling, due to the use of open differentials in the vehicle's drivetrain [49]. The work by Datar et al. [14] addresses this by having separate measurements for each wheel, however, it comes at the cost of additional build complexity.

Why the RPS only adjusts steering angle: In III-D, we only adjust the steering angle for two reasons. First, the rate at which lateral acceleration can be changed (lateral jerk) by the steering is much higher than what is achieved by changing the vehicle's velocity. Consider the following example; when turning at a speed of $3.0m/s$ with a friction coefficient of 1.0, the steering angle would be ≈ 15 degrees. The steering can move from $+15$ to -15 degrees in 0.1 seconds, based on the servo specifications. This results in a lateral jerk of $\approx 200m/s^3$. On the other hand, when turning, only a small fraction of the grip is available for changing speed. For the sake of the argument, assume that this is not the case, and all of $9.81m/s^2$ of longitudinal acceleration is available. Within 0.1 seconds, the effective lateral jerk would still be $\approx 50m/s^3$, which is 4x lower than what would be obtained from the steering.

Second, slowing down while turning can momentarily increase the lateral acceleration due to a phenomenon generally known as "lift-off oversteer" [7]. Intuitively, reducing the speed –done to prevent a rollover due to excess lateral acceleration– moves more weight onto the front tires almost instantly. This momentarily reduces the turning radius, without significantly reducing the vehicle's velocity. This increase in lateral acceleration then results in a rollover. The exact extent to which this happens, and therefore preventing it, would require knowledge of the vehicle's body frame velocity as well as the tire parameters, which we chose not to depend on.