Unsupervised testing of autonomy software

Sean Hickey, Nickolas Vlahopoulos

University of Michigan

ABSTRACT

The verification and validation of autonomous ground vehicle software represent a significant portion of lifecycle cost, driven by system complexity and operation in unstructured environments. Conventional testing strategies rely on scripted scenarios and physical trials that may not uncover rare yet critical failures that require precise combinations of inputs to emerge. The autonomy software must be robust to adversarial attacks, communication disruptions, and component-level hardware faults inherent to their operational environments. This work presents a system-agnostic new capability for unsupervised robustness testing of autonomy software. Its performance is demonstrated on an Army-related simulation platform. There are four main functionalities to the testing capability presented here (injection, termination, automation, and a test case generation driven by a genetic algorithm (GA)). The injection process intercepts data, alters it, and injects the modified data back into the autonomy software. The termination module monitors the type of termination that is encountered (successful or not). It also tracks the type of fault when one is generated. A GA jointly searches the scenario and fault-injection spaces, emulating real-world failures and adversarial attacks. This is done in an automated iterative process that requires minimal manual effort (unsupervised operation). Faults are injected via ROS topic remapping or UDP packet manipulation without modifying the autonomy stack. Each injection is defined by a set of parameters selected by the GA. The algorithm prioritizes maximizing fault severity, maximizing test case diversity, and maintaining an even distribution of fault outcomes. An Error Severity Score combines individual scores associated with a vehicle's operation (e.g., maximizing path deviation, overspeed, jerk, etc.). Each one is scored against a baseline run with no fault injection. Scoring against the baseline lets the algorithm distinguish a clean completion from a mission that is barely completed. The outcome of the unsupervised testing is a replicable set of cases, each generating a failure mode under different data manipulations. The effectiveness of the approach is demonstrated on the Robotics Technology Kernel (RTK) autonomy stack within the ProjectGL simulator, applied to a generic but representative leader-follower platoon in which the leader follows a prescribed path and the follower runs RTK. Three GPS attack models are considered: denial-of-service attacks on GPS packets, a constant offset to the reported position, and a drifting position error that accumulates over time. Their start time, duration, and magnitude parameters collectively span the search space. The algorithm evolves the selected data manipulations that drive the follower vehicle into multiple distinct failure modes (large convoy gap, timeout, drifting off the path, and degraded completion). In the final generation, GPS offsets as small as 1.1 meters combined with drift rates of 0.1 meters per second were sufficient to trigger failure. Such drift-induced failures resemble behaviors observed during testing under GPS position errors. The new process discovers many

more conditions that cause failures than randomly injecting modifications into the data stream. The new methodology identifies unique input combinations that give autonomy engineers concrete information for building more robust systems, reducing the cost of field validation.