Verification of Autonomous Systems Workshop ICRA 2018 - Session 2 Notes

Galen - Not fully capable - trust too much/not enough. Simulations, state space explosion. Seach based testing techniques. Collisions - V&V - hard to design objective functions, and only find one type of failure mode (the kind you're looking for). Google car example (stop signs).

Goal: How to find unknown unknown failure modes? What about metrics? Grab bag, problems with scaling to many many parameters. Need diverse and challenging scenarios. Trying to find the countours/boundaries in the performance space. Also want to make sure that the tests generated by the simulator can be run in the field.

Want to generalize within the boundaries and ID clusters in the outputs. Look for inputs that drive performance space boundary conditions - want to be focused on the areas where changes in inputs cause changes in classes of output - want lots of samples at the boundaries and only enough samples within the classes to verify the generalization.

Side effect - handles scaling and enables ID of things that have no impact. UUV example task is complex - simulator finds and focuses on rare events and explainability: "Why did it do X" "Are you going to fail?" Focus on continuous inputs to thresholds rather than binary thresholds themselves (works best with raw data rather than derived triggers).

Hierarchical search trees - search if all; natural clusters. Variance drops as proximity to raw data increases.

Could results be reproduced in the field? Even with wildly different environmental conditions, were still able to demonstrate presence of predicted performance boundary! No relationship between the modeled/simulated environment and the actual environment (different in most possible ways), but still reproduced test.

Signe - I gave this talk!

Zhe - Trying to be certain about what is intended and how to do it - how to design, plan, and solve AI problems - ensure end result is tested. Use correct-by-construction approach: Design -> Formalize -> Formal Model (verification) -> Refinement -> Implementation. Correct, secure, zero-bugs (testing only verifies presence of bugs, not absence). Using Isabelle (language for formal design). Prove properties: information flow security. Path from high level to low level verification. Matching. Code generation. Connect logic to computation. Not for industry (time, effort requirements high). PAT verification tool - service -> MOOS.

Panel:

Transferability? Sim -> HW and -> Domain randomization for robustness. Revisit process? No one SW tool. Verification in design loop? Check all environments. No

contour map.