HIGH ASSURANCE CONTROLLER DESIGN AND VERIFICATION OF A TWO-WHEELED INVERTED PENDULUM

Ethan Lew (elew@pdx.edu), Amanda Voegtlin, Patrick Gmerek, Justin Patterson / Dr. Marek Perkowski / Galois Inc.

OVERVIEW

Cyber-physical systems are increasingly being used to automate and enhance daily life. Robots are being installed in offices and homes, where they can guard, care, service, explore and entertain inhabitants. The intrinsic complexity present in today's robots demand that system verification must take up more of the design process; this is an observed bottleneck, with up to 80% of development costs spent on verification.

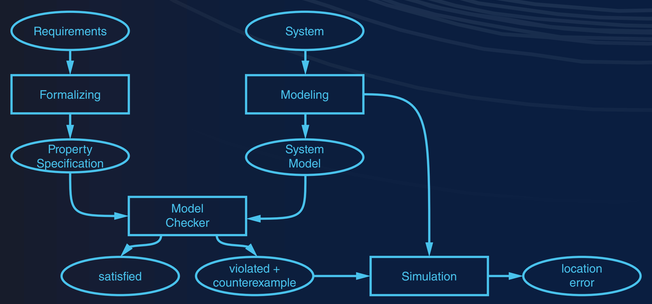
This capstone went through the design, implementation, and verification of a two-wheeled inverted pendulum (TWIP) robot. The robot contains enough instability to make it a suitable sandbox for controller design and verification. The methods used serve to illustrate how a model checker can be used to verify a hybrid system. The robot was built and shown to have improved stability from the high assurance pipeline that was used to create it.

PROCESS

A workflow was chosen such that the system properties and robot modeling could happen concurrently. This established that the design should have certain properties (a suitably large stable region). These requirements would be checked using a model checker, which will either satisfy or violate the requirements. If a requirement is violated, there are three sources of error:

model error	the model created is flawed
design error	the design is flawed
property error	the specification is invalid

A simulator was created in order to load counterexamples and determine the type of the error.



MODELING

The robot consists of two wheels, a chassis, two actuators, and a motion control unit. The kinematics of the robot were formulated by producing generalized coordinates that describe a configuration space. The robot's nonholonomy was exploited by adding a Pfaffian constraint to the kinematics, which allowed for the robot's joints to be fully expressed by the robot's overall motion.

Dynamics were derived using the Lagrange-Euler formulation. This was done by finding the Lagrange equations for the system.

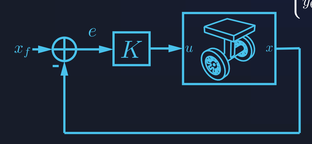
$$K = M_w v^2 + I_w \left(\frac{v}{r}\right)^2 + 2\left(M_w + \left(\frac{I_w}{r^2}\right)\right) d^2\omega^2$$

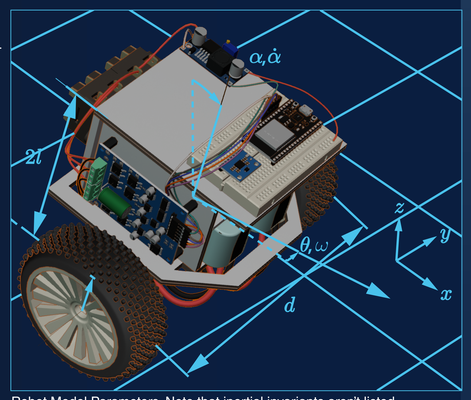
$$+ \frac{1}{2} M v^2 + \frac{1}{2} I_M \dot{\alpha}^2 + \frac{1}{2} I_p \omega^2$$

$$+ \frac{1}{2} m (v + l\cos(\alpha)\dot{\alpha})^2 + \frac{1}{2} m (-\sin(\alpha)\dot{\alpha})^2$$

$$U = mgl\,(1 - \cos(\alpha))$$

The dynamic analysis formulate a state space with five state variables of interest, three of which matter for the stability analysis: tilt angle, tilt angular rate, and forward/backward velocity. Provided that these quantities are measurable by the motion control unit, a state space controller can be made to control these values.





Robot Model Parameters. Note that inertial invariants aren't listed.

In order to facilitate verification, the state space evolution function needs to be expressed as a system of first order differential equations. This form permits many numerical solvers to approximate the system's orbits in state space.

$$\begin{cases} \dot{y}_1 = y_2 \\ \dot{y}_2 = \frac{\cos(y_1) I_m \left(-m l y_2^2 \sin(y_1) - \frac{\tau_l}{r} - \frac{\tau_r}{r} - d_l - d_r\right) + mgl \sin(y_1)\left(M + 2M_w + m + \frac{2I_w}{r}\right)}{(l^2 m + I_M)\left(M + 2M_w + m + \frac{2I_w}{r^2}\right) - (\cos(y_1))^2 l^2 m^2} \\ \dot{y}_3 = y_4 \\ \dot{y}_4 = \frac{2d\left(\frac{\tau_l}{r} - \frac{\tau_r}{r} + d_l - d_r\right)}{I_p + 2\left(M_w + \frac{I_w}{r^2}\right) d^2} \\ \dot{y}_5 = y_6 \\ \dot{y}_6 = \frac{\left(l^2 m + I_m\right)\left(-m l y_2^2 \sin(y_1) - \frac{\tau_l}{r} - \frac{\tau_r}{r} - d_l - d_r\right) + m^2 g l^2 \sin(y_1)\cos(y_1)}{(l^2 m + I_M)\left(M + 2M_w + m + \frac{2I_w}{r^2}\right) - (\cos(y_1))^2 l^2 m^2} \end{cases}$$

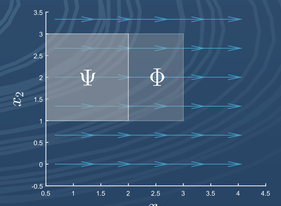
Once the dynamics were modeled for the open loop TWIP system, a linear control law can be applied. PD, PID, and LQR designs were created for the robot. Also, system linearization allowed a switching PID to be derived, changing tuning weights to incorporate locally linearized dynamics.

VERIFICATION

Model checking is an automated technique that, given a finite-state model of a system and a formal property, systematically checks whether this property holds for (a given state in) that model.

CTL-A (Computation Tree Logic for Analog)

Computation tree logic (CTL) is a branching time logic that is based on path quantifiers and temporal operators. An extension to the language, analog operators, add inequalities that can describe polytopes in a dynamical system's state space.



$\Phi = (x_{(1)} > 2) \wedge (x_{(1)} < 3) \wedge (x_{(2)} > 1) \wedge (x_{(2)} < 3)$

$\Psi = \mathsf{EF}\,(\Phi)$

Describing state space using CTL-A

CTL-A allows for a rigorous description of common verification questions, namely

Safety	Does instability exist?
Liveness	Can the system stabilize from a condition?
Fairness	How can stability repeatedly occur?
Real-time	Does stability occur fast enough?

These problems all address the stability problem in different ways. Safety addresses whether the robot is globally asymptotically stable (GAS), whereas liveness determines the region of attraction that results in a stable robot. Fairness addresses what conditions could occur where stability cannot be achieved again.

CTL is well-suited to address discrete finite automata, so a system discretization method must be created to utilize this verification technique.

$\Phi := a \mid \mathbf{z} * \mathbf{v} \mid \Phi \circ \Phi \mid \neg\Phi \mid \triangleright \diamond \Phi \mid \triangleright\, \Phi \,\mathsf{U}\, \Phi \mid \triangleright \diamond^{-1} \Phi \mid \triangleright\, \Phi \,\mathsf{U}^{-1}\, \Phi$

a	boolean state variables			
$\mathbf{z}$	**continuous state variables**			
$\mathbf{v}$	**constant real values**			
$*$	**analog operators**	$>$	$=$	**greater**
		$<$	$=$	**smaller**
$\circ$	boolean operators	$\vee$	$=$	or
		$\wedge$	$=$	and
$\neg$		$\neg$	$=$	not
$\triangleright$	path quantifiers	A	$=$	on all paths
		E	$=$	on some paths
$\diamond$	temporal operators	X	$=$	next-time
		F	$=$	eventually
		G	$=$	always
U		U	$=$	until
$^{-1}$	past	time inversion		

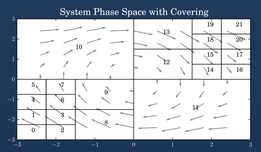
CTL-A Logic Grammar

Symbolic Dynamics

Symbolic dynamics relates a discretized dynamical system to a sequence of abstract symbols corresponding to the system's state. Transitions between states are created from a shift operation that arises from system evolution.

Symbolic encoding allows continuous time, continuous state behavior to be characterized by a labeled transition system (LTS). LT systems are commonly used by verification tools and allow analog and dynamic behavior to be verified by tools made for digital systems.

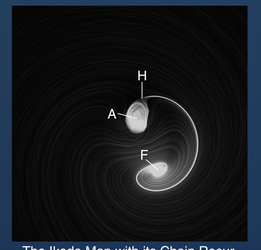
State Space Subdivision



Constructed Cyclical Directed Graph

Finding coverings in state space that capture the transitions of the chain recurrent set is non-trivial. Many approaches to Markov partitioning are available in the literature, although hyperrectangular subdivision is used frequently due to its computational simplicity and spatial data structure advantages.

Several selection criteria are available to determine which covering to subdivide. Angle deviation of a covering is used to determine attractors.



The Ikeda Map with its Chain Recurrent Set Components Labeled

Structural Graph of Ikeda Map

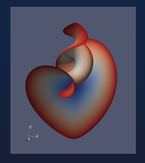
$$\begin{array}{c} \\ \infty \\ H \\ A \\ F \end{array} \begin{array}{c} \begin{array}{cccc} \infty & H & A & F \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

Structural Matrix of Ikeda Map

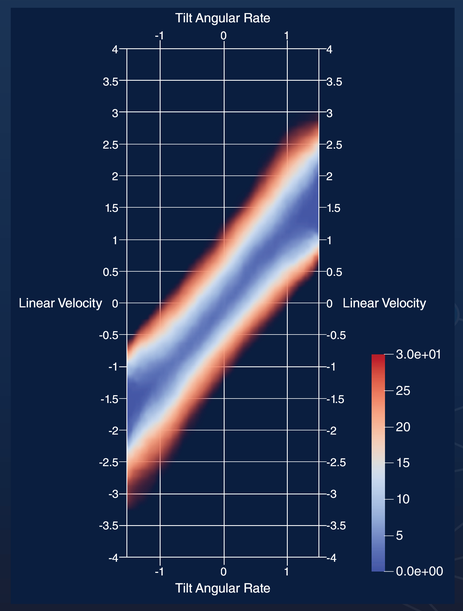
Controller Verification

Being able to associate coverings in state space to a region of attraction means that a volume can be found which contains all stable points. Assuming all other state variables to be at rest, **verification found that the robot build should be able to stabilize within 26 degrees of the equilibrium point.**

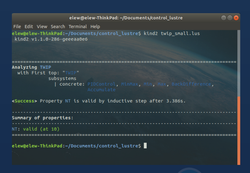
Velocity and tilt angular affect the robot's ability to stabilize. There was found a large region where the rates can be sizable, but because of their direction, are still controllable. Provided that this stable region can be approximated on the embedded implementation, the robot will have the ability to determine if it can stabilize or not, allowing it to fail gracefully should it be incapable of returning back to a stable point.



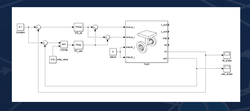
As the desired set point of a TWIP is hyperbolic, a stable manifold solver was implemented in order to see how the shape changed in response to the controller design.



The stability convergence time between linear velocity and tilt angular rate. Note that the robot has enhanced stability when the tilt angular rate and velocity are similar, able to converge quickly to the stable region.



A SMT-based automatic model checker called Kind2 was used to verify the system. As Kind2 doesn't support transcendental functions, the system was partitioned into coverings and a linear system was assigned to each one. This allowed the checker to approximate a nonlinear system during controller verification, incorporating all dynamics.

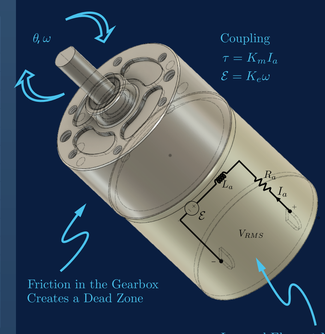


MATLAB/Simulink was utilized as a design, simulation, and verification tool. The robot dynamics were added to a Simulink block and the controller design workflow can be performed after linearizing the feedback path. Also, a metric temporal logic based verification tool called S-TaLiRo was explored, but the results weren't trustworthy.

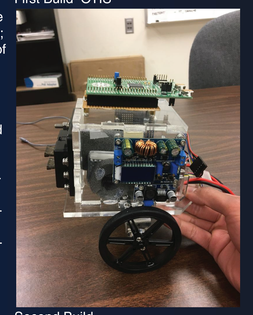
BUILD

In order to accurately portray the model used in the design and verification, a built robot needed an accurate motor controller as well as an attitude and heading reference system (AHRS). This was done by modeling the DC motors as a linear state space model with the root-mean-square voltage as input and torque as the output. An extended Kalman filter was implemented to provide sensor fusion to the 9DoF system used, being magnetic, angular rate and gravity (MARG) sensors. At first, a complementary filter was used, but was found to be too magnetically sensitive.

First Build "OTIS"



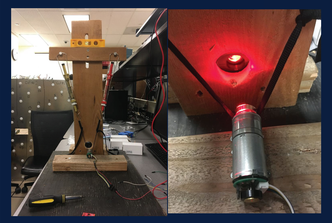
The first prototype, followed the philosophy of rapid-prototyping; it was constructed quickly out of illustration board and used components that were already on-hand. This robot used an ESP32 as the processor and the controller was implemented in C. It also included a Bluetooth module to provide wireless telemetry and remote control. The robot's motor had an appreciable deadzone that produced a limit cycle in state space when using a linear controller. This necessitated the need to create a dead zone compensator to remove it.



Second Build

Our second robot was made from acrylic and was modeled in AutoDesk Fusion360. The rough dimensions of the chassis were derived from the first robot. The specific dimensions for the mounting holes of the Buck/Boost converters, motor driver, and fusebox were determined by reviewing the part datasheet (where applicable) or by measuring the components. Once the modeling in Fusion360 was complete, the panels were exported for laser cutting. To facilitate easy assembly and disassembly, the edges of the chassis panels were joined by means of fingerjoints and hot glue. This provided a sturdy-yet-reversible bond between the panels.

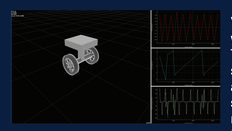
VALIDATION



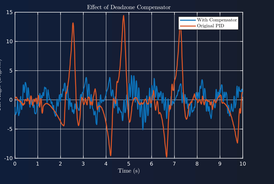
Validation was performed chiefly with comparisons of telemetry; the output of the simulator model and the output of the actual hardware, observing changes in those outputs as adjustments to the controller were made.

Using the telemetry dashboard, the PID weights were tuned using the Ziegler-Nichols method initially. Next, better PID weights were determined by the robot's dynamics model.

A Prony Brake was used to measure the torque of a Pololu DC motor. We validated the linear relationship between torque and current, along with the back electromotive force (EMF) and angular speed. Thus, we were able to calculate the motor's torque coefficient and EMF coefficient.



Validation revealed that a large dead zone was present on the motor of the first build. This dead zone was integrated into the simulation, which allowed for the design and verification of a dead zone compensator. Telemetry data shows over a 60% reduction in steady state oscillation amplitude from this compensator.



CONCLUSION

The primary accomplishment of this team was implementing a way to use tools designed for digital system verification to verify hybrid systems. This was accomplished by discretizing the continuous time, continuous state behavior using symbolic encoding. The results extracted from symbolic methods closely aligned with brute force simulations. A working TWIP robot was built to illustrate how this verification methodology can be incorporated into a real system build. The workflow involved deriving an accurate model of the system's kinematic and dynamics. Then, the model's equations were converted into forms that various verification and analysis tools could understand—system of first order differential equations and finite discrete automata. A model checker was able to check if a controller design complied to system requirements. Finally, validation was able to affirm that the robot build was consistent with the model and met the formally specified properties.

ACKNOWLEDGEMENTS / REFERENCES

Dr Marek Perkowski and the Maseeh College of Engineering and Computer Science, for overseeing the project. Michal Podraedsky and Matt Clark at Galois Inc, for giving us this project and advising us along the way. Bert Dechant for his advice on robotic design. Donald and Jenna Bell for their insight on inverted pendulum control.

Li, Z., Yang, C., & Fan, L. (2012). *Advanced control of wheeled inverted pendulum systems*. Springer Science & Business Media.

Drechsler, R. (Ed.). (2004). *Advanced formal verification (Vol. 122)*. Norwell: Kluwer Academic Publishers.

Zecevic, A., & Siljak, D. D. (2010). *Control of complex systems: Structural constraints and uncertainty*. Springer Science & Business Media.

Lynch, K. M., & Park, F. C. (2017). *Modern Robotics*. Cambridge University Press.

Strogatz, S., Friedman, M., Mallinckrodt, A. J., & McKay, S. (1994). *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. *Computers in Physics*, 8(5), 532-532.

Osipenko, G. (2006). *Dynamical systems, graphs, and algorithms*. Springer.

Baier, C., & Katoen, J. P. (2008). *Principles of model checking*. MIT press.

Kunkel, P., & Mehrmann, V. (2006). *Differential-algebraic equations: analysis and numerical solution (Vol. 2)*. European Mathematical Society.

Grasser, F., D'arrigo, A., Colombi, S., & Rufer, A. C. (2002). JOE: a mobile, inverted pendulum. *IEEE Transactions on industrial electronics*, 49(1), 107-114.

Ha, Y. S. (1996). Trajectory tracking control for navigation of the inverse pendulum type self-contained mobile robot. *Robotics and autonomous systems*, 17(1-2), 65-80.

Delnitz, M., Froyland, G., & Junge, O. (2001). The algorithms behind GAIO—Set oriented numerical methods for dynamical systems. *In Ergodic theory, analysis, and efficient simulation of dynamical systems* (pp. 145-174). Springer, Berlin, Heidelberg.

Delnitz, M., & Hohmann, A. (1997). A subdivision algorithm for the computation of unstable manifolds and global attractors. *Numerische Mathematik*, 75(3), 293-317.

Goher, K. M., Tokhi, M. O., & Siddique, N. H. (2014). Dynamic modeling and control of a two wheeled robotic vehicle with a virtual payload. *ARPN Journal of Engineering and Applied Sciences*, 9(3), 7-41.

Nawawi, S. W., Ahmad, M. N., & Osman, J. H. S. (2009). Real-time control of a two-wheeled inverted pendulum mobile robot. *World Academy of Science, Engineering and Technology*, 39, 214-220.

England, J. P., Krauskopf, B., & Osinga, H. M. (2004). Computing two-dimensional stable manifolds and stable sets of planar maps without the inverse. *SIAM Journal on Applied Dynamical Systems*, 3(2), 161-190.

Diebel, J. (2006). Representing attitude: *Euler angles, unit quaternions, and rotation vectors*. Matrix, 58(15-16), 1-35.

Mirzaei, F. M., & Roumeliotis, S. I. (2008). A Kalman filter-based algorithm for IMU-camera calibration: Observability analysis and performance evaluation. *IEEE transactions on robotics*, 24(5), 1143-1156.