MathWorks News&Notes

The Magazine for the MATLAB® and Simulink® Community

WIND POWER

Power plant control software development at Vestas Wind Systems

ALSO IN THIS ISSUE

Pragmatic Digital Transformation

Aiming Parker Solar Probe at the Sun

Cleve's Corner: Deep Learning for Trail Camera Data

Deploying Predictive Maintenance Algorithms

MathWorks®

Dr. Ayanna Howard, Chair of the School of Interactive Computing at Georgia Institute of Technology, uses miniature humanoid robots to keep children with behavioral or motor disabilities engaged during physical therapy sessions.

"The robot can provide guidance, motivation, and feedback while creating a bond with the child who is using it."

mathworks.com/robot-therapy

FEATURES

4 Pragmatic Digital Transformation

Improve processes and end products by reusing data and models throughout the product or service life cycle.

8 Equipping Student Engineers with Data Science Skills

"Data science skills will soon be essential for all engineers, whether they are applying machine learning algorithms, providing data for those algorithms, or making decisions based on the results."

10 Environment-in-the-Loop Verification of Automotive Radar IC Designs

In their "shift-left" V&V methodology, NXP Semiconductors engineers perform virtual field trials early in development, using driving scenarios based on the Euro NCAP standard.

14 Seeing the Road Ahead: The Path Toward Fully Autonomous Self-Driving Cars

Perception technologies paired with AI help drivers park, brake, and steer; detect lane boundaries; and prevent sleepy motorists from drifting off behind the wheel.

18 Developing Wind Power Plant Control Software

"We have engineers worldwide working in parallel on the same model with lots of merges. By combining Model-Based Design and continuous integration, we've shortened iterations and automated testing processes."

20 Deploying Predictive Maintenance Algorithms to the Cloud and Edge

A packaging machine example shows how to develop a predictive maintenance algorithm and deploy it in a production system with MATLAB®.

24 Aiming Parker Solar Probe at the Sun with Simulink GNC Software

GNC algorithms keep the Parker Solar Probe and its thermal protection system oriented toward the Sun as the spacecraft passes through the solar atmosphere.

28 Cleve's Corner: Experiments with Deep Learning for Trail Camera Data

Could an off-the-shelf CNN be trained to identify individual animal species that visit trail cameras? Cleve Moler wanted to find out.

34 Analyzing Satellite-Based Radar Imagery for Iceberg Surveillance

C-CORE partnered with Norwegian energy company Equinor to develop software that uses deep learning to classify targets in SAR images.

Cover image courtesy of Vestas Wind Systems A/S

QUICK READS

7 Running a Car on Sunlight

16 Third-Party Products: Power Electronics Control Design

17 Developing an Open-Source Toolkit for Radiation Therapy Planning

33 Robotics and Autonomous System Boundary-Breakers

37 Removing the Water from Underwater Images

Managing Editor
Linda Webb

Editor
Rosemary Oxenford

Art Director
Kevin Hart

Graphic Designers
Gabrielle Vendetti, Shriya Srinivas

Production Editor
Julie Cornell

Technical Writer
Jack Wilber

Digital Production Specialist
Glorimar Rivera

Printer
DS Graphics

Print Liaison
Jill Mespelli

Editorial Board
Thomas Andraczek, P.J. Boardman, Michael Carone, Stacey Gage, Michelle Hirsch, Andy May, Cleve Moler, Sameer Prabhu, Richard Rovner, Loren Shure, John Stewart, Jim Tung

Contributors and Reviewers
R. Aberg, S. Avadhanula, M. Bangert, A. Baru, R. Boldt, G. Bourdon, W. Campbell, M. Carleton, N. Chavoor, E. Cigan, R. De Rafael, S. DeLand, K. Devleker, K. Dodge, E. Donovan, G. Drayer Andrade, M. Erickson, T. Erkkinen, R. Gentile, J. Ghidella, H. Gorr, S. Gross, H. Hafren, P. Hagen Nielsen, L. Harvey, L. Heske, A. Hira, D. Hoadley, R. Holt, C. Howell, O. Jäkel, M. Jung, S. Karlapalem, B. Khusainov, L. Lemieux, T. Lennon, J. Li, K. Lorenc, P. Massano, B. McKay, S. Miller, C. Mohtadi, T. Morton, A. Nehemiah, M. Pavia, P. Pilotte, J. Pingel, T. Popham, O. Pujado, E. Putnam, S. Rao, G. Reith, A. Rosenfield, O. Saarela, J. Sanderson, K. Santhanakrishnan, G. Schraberger, E. Selbach, H. Sharma, V. Suomi, N. Terhag-Graeff, A. Turevskiy, T. Vekua, V. Viswanathan, N. Wahl, P. Wallner, P. Webb, H. Wieser, A. Willard, J. Witzburger

Subscribe
mathworks.com/subscribe

Contact Us
mathworks.com/contact

Follow us on social media @MathWorks

Read online
mathworks.com/news-notes

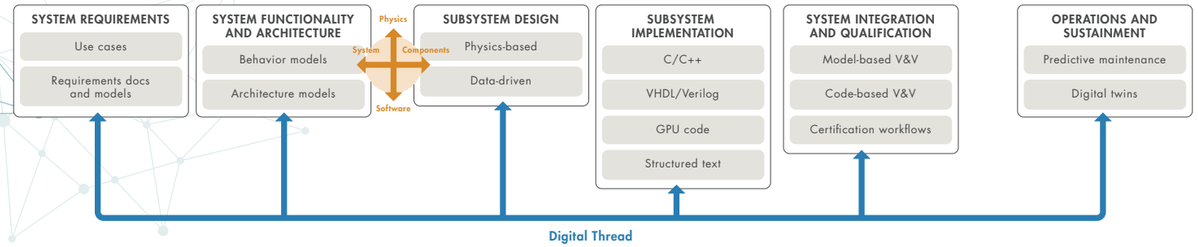
MIX From responsible sources FSC® C084277

MathWorks®

Printed on 30% post-consumer waste materials

Pragmatic Digital Transformation

By Michael Carone, Paul Pilotte, and Jim Tung, MathWorks

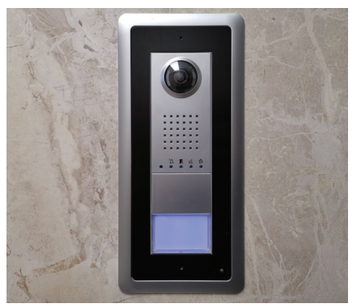


In pragmatic digital transformation, a digital thread connects your system from requirements to architecture to testing and the system in operation, opening opportunities to improve models, processes, and end products.

The doorbell started life as a simple mechanical device. It had one button with a single function: to ring a bell.

Today's version is likely to have a camera, a motion sensor, video, and a smartphone interface that can access data sent from the doorbell to the cloud. It is no longer just a doorbell; it is a complete security system.

The evolution of the doorbell is just one example of digital transformation—the use of technologies such as data analytics, connectivity, cloud computing, and AI to transform products, processes, and entire systems.



Almost every organization seems to include digital transformation in its vision and strategy, but most struggle with executing digital transformation initiatives. There are myriad reasons: the challenges of introducing new technologies and providing the workforce with relevant skills, ensuring that the company's culture and organizational structures are conducive to change, and anticipating correctly which processes need to change and how, to name a few.

To effect change, some organizations begin with proof-of-concept and pilot projects. They soon find themselves mired in a "pilot purgatory," unable to scale up by formalizing the piloted approaches and making them part of the company's standard workflows and practices. Other organizations start with large infrastructure development efforts that are difficult to execute and fail to meet the requirements of the actual projects, workflows, or products that emerge from the transformation strategies.

We have observed that organizations are often most successful with digital transformation when they adopt a *pragmatic* approach.

What Is Pragmatic Digital Transformation?

Pragmatic digital transformation does not require starting from the ground up or completely overhauling existing processes and assets. Just the reverse; its fundamental principle is *reuse*: In pragmatic digital transformation, data and models—and the engineering teams' associated skills in developing analytics, models, and simulations—are applied systematically to workflows throughout the life cycle of the product or service.

The systematic use of data can start with analytics developed specifically to get insights from experimental and research data. But it also means scaling and extending those analytics to the huge, heterogeneous sets of live and archived data, acquired from manufacturing, maintenance records, and other business processes, to enable data-driven decisions not only during research and design but also in production, operations, and maintenance.

Systematically Using Data: From Data Siloes to Data Analytics

As organizations recognize today, the challenge is not the lack of data but the crushing volumes and variety of their data—not only engineering, scientific, and field data but also business and transactional data. The diversity of data management approaches adds to the complexity: Data may be stored on-premise or in the cloud, in consolidated data lakes or separate databases, in relational databases or spreadsheets. And every datastore may have a different governance policy and access permissions.

Digital transformation begins when the accumulated knowledge and transformative potential of this data can be uncovered and applied systematically throughout the product life cycle. The core tasks are, first, to integrate data from multiple repositories; second, to develop analytics that are easy to

Using Big Data Analytics to Optimize Manufacturing Processes at GSK Consumer Healthcare

GSK Consumer Healthcare's R&D team wanted to improve manufacturing processes and increase capacity at the company's toothpaste manufacturing plants. The most cost-effective approach, they knew, would be to systematically use the historical data they had accumulated over the years. They set out to see whether they could learn from that history to make better products.

They began by focusing on process data. Accumulated across all their factories, formulations, and batches, the data amounted to terabytes, and it was housed in separate siloes, separate systems, and different formats.

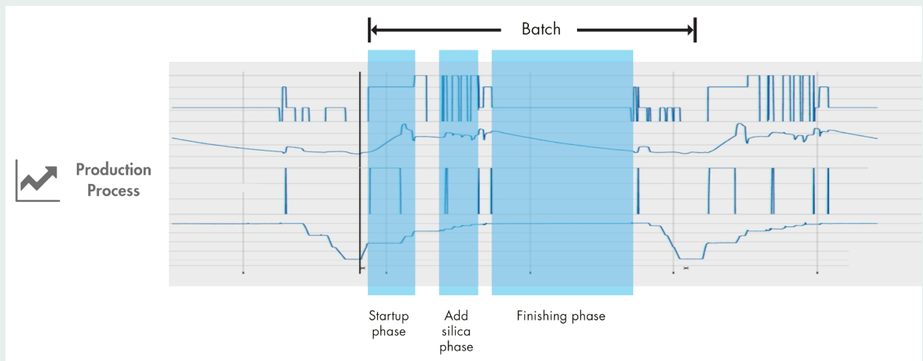
"Lying hidden within the servers and notebooks of the manufacturing communities, there exists a wealth of untapped knowledge. It is long overdue that we brush off this diligently collected process data and begin to learn the secrets it hides."

—Bob Sochon, GlaxoSmithKline

To get insights from their process data, GSK first needed to clean it by filtering out noise, filling in missing data, and removing outliers. They could then use it to compare phases from batch to batch.

The R&D team built an algorithm in MATLAB® to sort and tag the data by formulation phase (Startup, Add Silica, or Finishing), and ran this algorithm across all their process data. They built an interface in MATLAB that enables their process engineers to select and observe data by formulation combination, batch, and operator.

By linking manufacturing phases to analytical data, GSK has seen dramatic improvements in both processes and capacity—for example, vessel heating time, which used to take 30 minutes, now takes just two minutes. These improvements translate into significant business benefits: reduced time to market for new formulas, and increased output from factories previously thought to be close to full capacity.



Minimizing the Cost of Ownership with Simulation and Digital Twins at Atlas Copco

Air compressor manufacturer Atlas Copco has turned the systematic use of models and data into a collaboration platform that streamlines communications across their technical organizations and within their global sales organization.

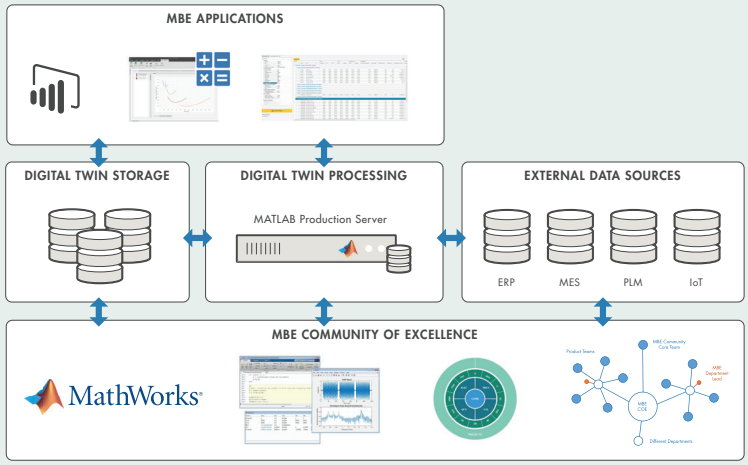
In developing their new ZR 160 VSD+ product line, Atlas Copco engineers had two priorities: reliability—if a single compressor fails, the entire production plant fails—and energy efficiency—electricity accounts for 75% of the total life cycle cost of a compressor, a considerable amount when the average compressor runs day and night for 10 years.

Not only did the team want to design an efficient product; they also wanted to *design the product efficiently*.

They implemented a framework to manage the models, the data, and variants based on a digital twin. The same models drive the configuration applications that their sales and application engineering teams use to configure and quote systems for specific customers.

With this platform they can quickly implement and deploy upgrades onto 120,000 machines that are in operation worldwide. Each machine is equipped with up to 50 sensors that continuously relay data back to the Atlas Copco data warehouse, enabling the service division to set up customer-specific predictive maintenance strategies based on real-time information on the condition of the machine.

ATLAS COPCO MODEL-BASED ENGINEERING PLATFORM



"We use a digital twin as the single source of truth and then build applications on top so that everyone has access to the same data and information."

—Carl Wouters, Atlas Copco

use and access; and third, to integrate those analytics into the workflow at the right time to enable groups throughout the organization (engineering, business-unit management, analysts, service teams, and more) to apply insights from the data to improve processes or designs.

Extending the Use of Models: From Development to the System in Operation

The systematic reuse of models is a basic principle of Model-Based Design, where models form a digital thread connecting development, design optimization, code generation, and verification and validation. This digital thread does not need to be limited to the development process; it can be extended to deployed systems in operation when design models are reused as digital twins. A digital twin—an up-to-date representation of a system or subsystem as it operates—can be used to assess the current condition of the asset, and more importantly, optimize the asset's performance or perform predictive maintenance.

Improved Processes, Deeper Insights

In pragmatic digital transformation, previously siloed data is combined and applied throughout development and deployment to improve processes and provide insights into system performance. A system model captures the high-level system behavior as well as the detailed subsystems. Those models connect to system requirements for traceability and early validation. Subsystem models can be reused to generate an implementation as software or an FPGA. And the models are reused for integration, validation, and verification, either at the model level or operating on the actual code.

By taking a pragmatic approach, organizations can reap the business benefits of digital transformation—improved quality, higher output, cost savings—while avoiding the struggles and pitfalls that deter some from embarking on, or even contemplating, a digital transformation initiative. ◆

RUNNING A CAR ON SUNLIGHT

The Lightyear One solar-powered electric car can run 450 miles on a single charge—and when the 54 square feet of solar panels covering its hood and roof are exposed to perpetual sunlight, Lightyear One could get 12,000 miles per year on solar power alone. The car's electronics convert sunlight into electricity and optimize how much energy the car draws from the battery. As a result, it uses just 160 watt-hours of electricity per mile.

Lightyear One is a startup founded by members of the Eindhoven University of Technology team that won the Bridgestone World Solar Challenge. To optimize aerodynamics, the team used lightweight materials and a small battery pack, incorporated motors into each wheel instead of relying on one large engine and a drive train, and replaced wing mirrors with cameras. In wind tunnel tests, Lightyear One broke the record for being the most aerodynamic five-seater electric car to date.

Lightyear One. Image credit: Lightyear

Equipping Student Engineers with Data Science Skills

By Thomas Popham, University of Warwick

Data science and machine learning will soon be essential skills for all engineers, whether they are applying machine learning algorithms, providing data to feed these algorithms, or making decisions based on the results. That is why, in 2018, we introduced data science as a thread through the Warwick Engineering degree, starting from the introduction of programming and simple statistical models during the first year, moving to a core data analytics module in the second year, and then offering more stream-specific modules in the third and fourth years.

Year One: Systems Modeling, Simulation, and Computation

All first-year engineering undergraduates take *ES197: Systems Modelling, Simulation, and Computation*. In this module, students learn how to use both physical and (simple) data-driven approaches for modeling engineering systems. This module also serves as an introduction to programming.

To familiarize themselves with programming and with MATLAB®, students complete lessons from the online MATLAB Fundamentals course. From an educator's perspective, this approach works really well, as it allows students to learn at their own pace and get immediate feedback on the programming exercises.

After applying the MATLAB skills they've acquired to assignments on curve fitting and deriving simple models and relationships from data, the students tackle modeling and simulation problems using examples from electrical, thermal, and translational systems.

In later assignments, students incorporate noise or other random effects into the model. For example, we have them create a simple model in MATLAB in which particles shoot up into the air and fall back down while being acted upon by random forces. The simulation produces an interesting 3D visualization (Figure 1). The entire project gives students confidence in their abilities to create their own models programmatically.

Year Two: Engineering Mathematics and Data Analytics

The second-year module *ES2C7 Engineering Mathematics and Data Analytics* focuses on solving regression, classification, and clustering problems. When I worked in industry, I saw that solving data science problems was relatively straightforward once the data was clean and in the proper format, but that is rarely the case with real-world data. With this in mind, I teach the students how to identify and remove outliers, handle missing values, and organize data in tables.

MATLAB live scripts are particularly useful during lectures because I can include formatted text and images to remind me of what I want to cover and because the output of the code appears along with the code that produced it. The Classification Learner and Regression Learner apps in Statistics and Machine Learning Toolbox™, meanwhile, make it possible to teach the broad principles of regression and classification without delving into implementation details (Figure 2).

After completing lab assignments on regression, classification, and clustering, the students work on a group project in which I ask them to imagine working for an engineering consultancy tasked with assessing the quality of manufactured steel components. The students must predict which components are most likely to fail using two datasets, one that is fairly clean and one that is messy and complicated.

Working with noisy data in a variety of file formats, including Excel®, CSV, and plain text, the students remove outliers, perform joins, and prepare the data to be used in training a model. Most groups use the Regression Learner app or implement linear regression in a MATLAB script; some try both approaches.

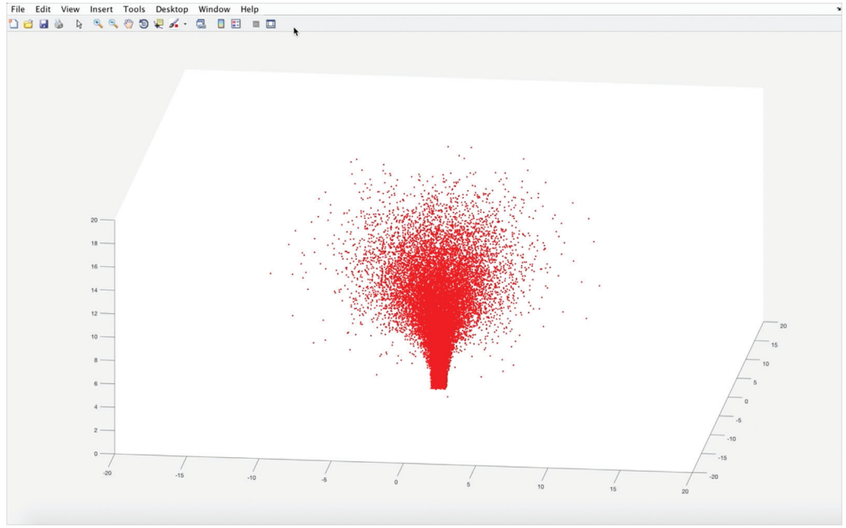


Figure 1. MATLAB 3D visualization of particles responding to random forces.

Year Three and Beyond

For students interested in exploring data science and machine learning further, Warwick offers a third-year module on intelligent system design that covers computer vision and more advanced machine learning techniques. In this module, I introduce the sense-perceive-act framework used in many autonomous control system applications. The quadcopter model in Simulink® (Figure 3) is very useful for showing this basic framework while introducing topics to be covered later in the module, such as Kalman filtering and optical flow.

Later, students develop a gesture recognition app with MATLAB that combines computer vision and machine learning. For this project, students develop a model capable of interpreting webcam images of their own hands and classifying them as one of several predetermined hand gestures. The project is particularly engaging for the students because they are working with their own data and need to think about factors such as lighting and how many different images are needed to train an accurate classifier.

Students who learn how to apply data science techniques in the context of real-world problems early in their studies are well prepared not only for advanced coursework in subsequent years but also for careers as practicing engineers. We have already received very positive feedback from our students on this approach—they have found that they are able to apply these techniques during undergraduate internships and to talk about these skills in interviews.

With device connectivity enabling companies to base their design decisions on data rather than on intuition or previous experience, engineers with a background in data analytics are very much in demand. We are confident that our graduates will be able to apply machine learning and data analytics whenever the situation requires it. ◆

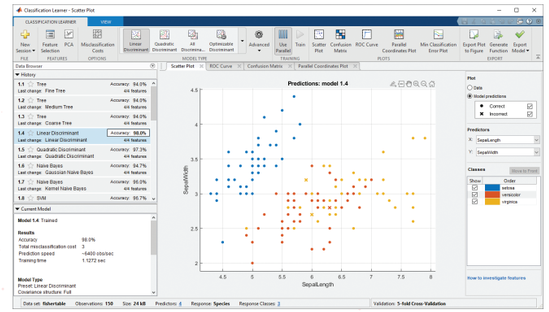


Figure 2. Classification Learner app.

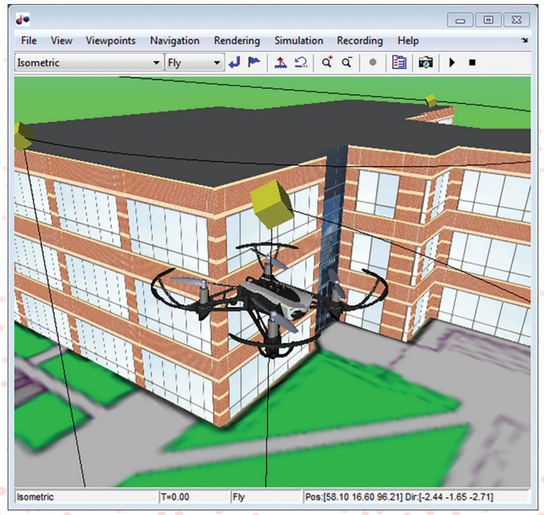
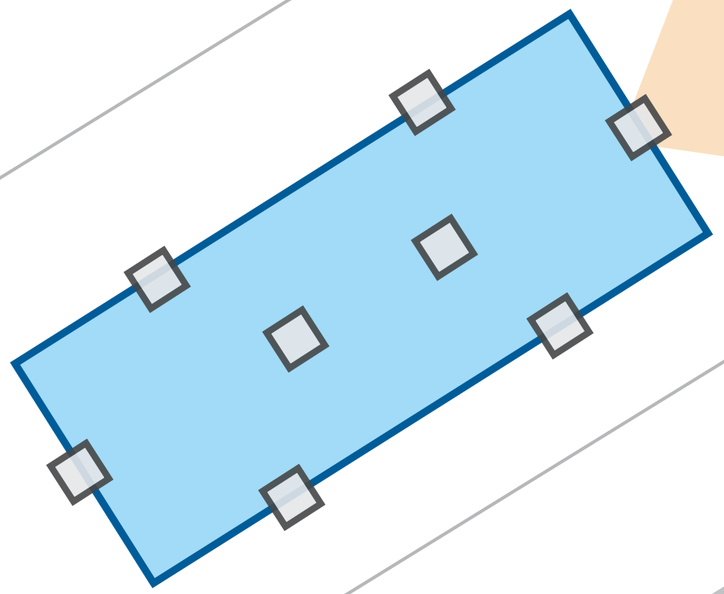


Figure 3. 3D visualization of Simulink quadcopter example.

Environment-in-the-Loop Verification of Automotive Radar IC Designs

By Sainath Karlapalem, NXP Semiconductors



At NXP Semiconductors, my team and I have developed a new methodology for verifying automotive radar integrated circuit (IC) designs. This *shift-left* methodology combines early verification of datasheet-level metrics with virtual field trials. By focusing on metrics at the specification level rather than the hardware implementation level, we ensure that the verification signoff criteria we use to evaluate a design align with those our customers are most interested in. And, by simulating on-road scenarios in virtual field trials, we enable environment-in-the-loop verification with realistic test stimuli for radar IC hardware.

Our customers, who include many tier 1 automotive suppliers, are most interested in the kinds of performance metrics captured on a datasheet, such as signal-to-noise ratio (SNR) and total harmonic distortion (THD). They are less interested in individual component test results, code coverage results, and other metrics at the hardware implementation level, although these results are a primary concern of most IC verification teams. In addition, our customers use field trials and real-world driving scenarios to evaluate complete radar systems, while IC verification teams often use test patterns that are far removed from real-world signals to evaluate individual RF, analog, and digital components (Figure 1).

The shift-left methodology that my team and I have defined and implemented aligns the processes we use to verify IC designs with the criteria our customers use to evaluate them. The on-road driving scenarios we developed for virtual field trials are based on the European New Car Assessment Programme (Euro NCAP) standard that many of our customers follow, and the functional and performance metrics we produce (for example, SNR) are the same metrics that our customers use to evaluate IC components in their own products.

Early Verification of Datasheet-Level Metrics

When verifying the digital portions of automotive radar systems in the past, my team adopted an approach based on the Universal Verification Methodology (UVM). This approach involved replicating the functionality of the design under test (DUT) with a reference model created in a high-level language. The output of the DUT was then compared with the output of the reference model for a given input test vector. The UVM tests did not capture SNR measurements and other metrics our customers were interested in, and even relatively small implementation changes, such as updating the coefficients of a finite impulse response (FIR) filter, required a corresponding change in the testbench. Keeping the testbench in sync with the implementation required considerable effort and time.

Given the drawbacks and limitations of this approach, we decided to focus our verification efforts on the functionality and performance of our design rather than on one-to-one equivalence between the implementation and reference model. Now, we develop MATLAB® algorithms that compute high-level design metrics such as SNR, THD, and power spectral density (PSD), as well as metrics for filters and other components, such as stopband attenuation and passband ripple. Using HDL Verifier™, we generate SystemVerilog DPI components from these MATLAB algorithms and integrate them into the HDL testbench for the Cadence® simulation environment (Figure 2).

Sample signal data is collected from the DUT and passed to the DPI-C function generated from our MATLAB verification code. We plot the results (Figure 3) and check them against the system requirements to ensure that the design matches the specifications.

Using DPI-C models generated from MATLAB enables us to compute functional and performance metrics at multiple interfaces in

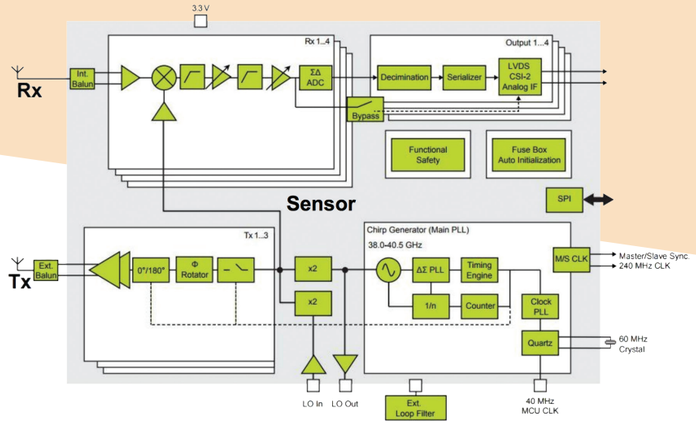


Figure 1. Automotive radar system architecture showing RF, analog, and digital subsystems.

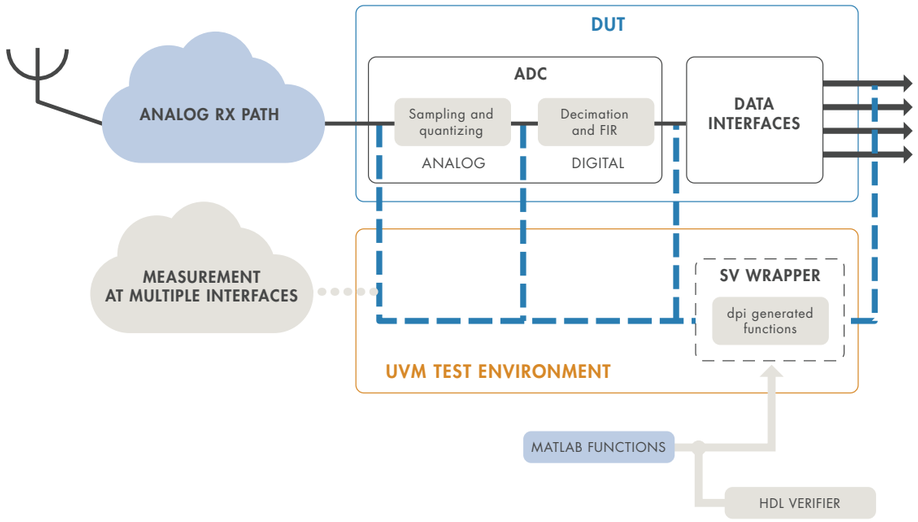


Figure 2. A test environment using MATLAB verification functions implemented in a SystemVerilog wrapper via DPI-C with HDL Verifier.

the Cadence HDL verification environment. We can decouple design implementation from verification and conduct testing at a level of abstraction more closely aligned with the metrics that interest our customers.

We can also reuse the C code generated from MATLAB to analyze the results from tests of initial silicon. For example, we collect sample data from our radar sensor IC and pass it through the same SNR calculation C functions generated from MATLAB that we used to verify our design in SystemVerilog.

Virtual Field Trials

In our transition to a metrics-driven verification approach we conduct virtual field trials using data from real-world driving scenarios. In the past, we verified the RF, analog, and digital subsystems separately, using a different set of test vectors for each subsystem. Few of these test vectors were derived from radar reflections obtained during on-road tests.

We have extended our methodology to include *environment-in-the-loop* verification. We now build driving scenarios using the Driving Scenario Designer app in Automated Driving Toolbox™ (Figure 4). Prebuilt scenarios in the app represent the Euro NCAP test protocols, our customers' benchmark for evaluating radar system performance.

Next, we build a radar sensor model with Phased Array Toolbox™. To match this model with the datasheet specifications of our actual sensor, we adjust parameters for antenna aperture, peak transmit power, the receiver noise figure, and the number of antenna elements. We also adjust parameters that affect the frequency-modulated continuous-wave (FMCW) waveform, including maximum range, chirp duration, sweep bandwidth, and sample rate. We integrate the sensor model into the driving scenario that we created earlier, virtually mounting the radar sensor on the ego vehicle (Figure 5).

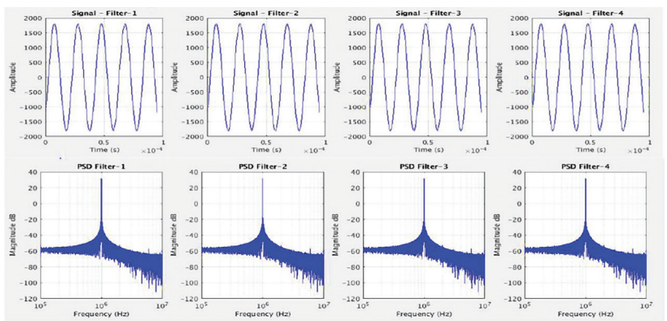


Figure 3. Sample signals (top) and power spectral density plots (bottom) computed using MATLAB.

> **The shift-left methodology that my team and I have defined and implemented aligns the processes we use to verify IC designs with the criteria our customers use to evaluate them.**
>
> – Sainath Karlapalem, NXP Semiconductors

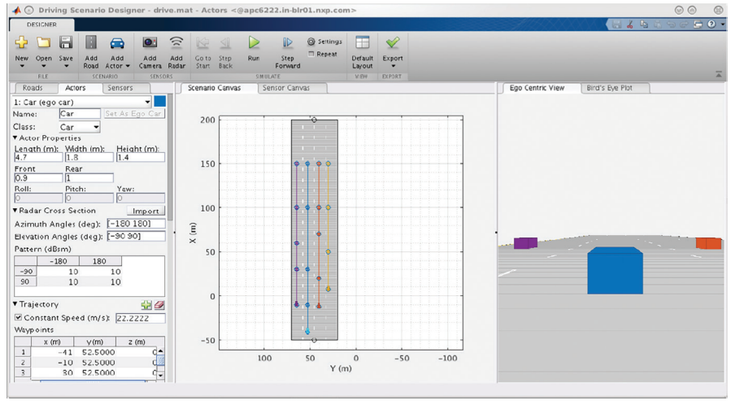


Figure 4. Driving Scenario Designer app in Automated Driving Toolbox.

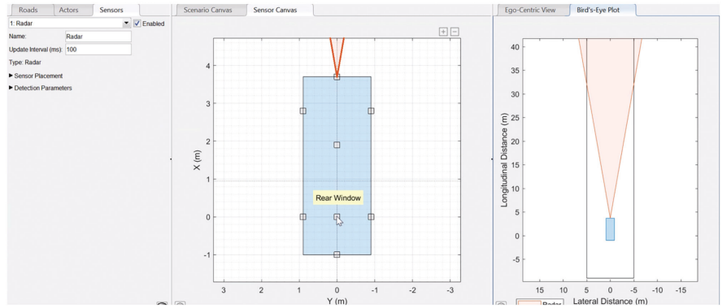


Figure 5. Interface for managing the placement of the radar sensor on the ego vehicle.

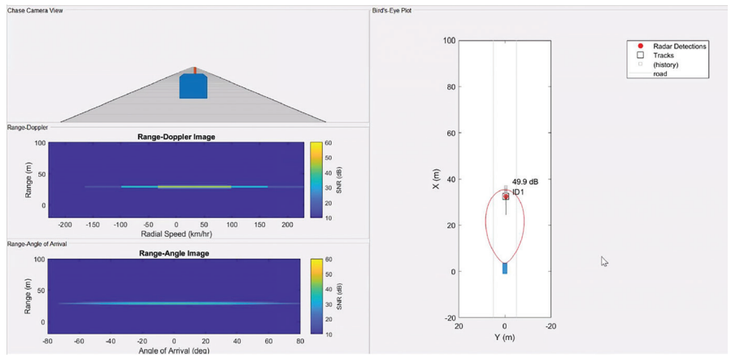


Figure 6. Chase camera view (top left) and bird's-eye plot (right) from a virtual field trial.

We then execute the driving scenario and capture the mixer output of the sensor, a signal dechirped from the radar reflections of objects in the scenario. We pass this dechirped signal through a Simulink® model of our ADC design to produce digital IQ data, which we feed into our digital baseband processing chain.

With this setup we can generate IQ data based on Euro NCAP driving scenarios and conduct virtual field trials of our digital processing chain early in the development phase—potentially a year or more before first silicon (Figure 6).

Future Work

We have extended our use of the new methodologies and workflows to next-generation radar transceivers. For these products, we will incorporate environmental effects into our scenarios so that we can see how the design performs in the presence of rain or fog, for example.

Recognizing that nothing restricts this new verification methodology to the digital components of automotive radar systems, we are looking forward to applying virtual field trials to analog components and to other applications, such as car-to-car communication systems. This article focused on verifying the digital portion of the sensor implementation, but this environment-in-the-loop approach can easily be extended to verify mixed-signal and RF designs such as the ADC in the sensor design. ◆

Many thanks to my NXP Semi team member Kaushik Vasanth for implementing our environment-in-the-loop verification methodology, and to Vidya Viswanathan of MathWorks for offering timely technical support.

SEEING THE ROAD AHEAD

THE PATH TOWARD FULLY AUTONOMOUS, SELF-DRIVING CARS

At the 1939 New York World's Fair, General Motors unveiled its vision of smart highways and self-driving cars. Although that dream has yet to materialize, advances in perception technology for autonomous cars have been dramatic, with cameras that read road and traffic signs, ultrasonics that sense nearby curbs, laser-based lidar for seeing 200 meters out or more, and radar that measures range and velocity. Paired with artificial intelligence (AI), these technologies help drivers park, back up, brake, accelerate, and steer; detect lane boundaries; and even prevent sleepy motorists from drifting off behind the wheel.

Here are three ways that engineers are advancing vehicle perception to usher in a future of fully autonomous, self-driving cars.

BEAMSTEERING RADAR

Engineers at California-based Metawave are pushing the limits of radar to recognize other autos, pedestrians, stationary surroundings, road hazards, and more in all weather conditions and at night. The company's analog radar platform, SPEKTRA™, forms a narrow beam and steers it to detect and classify objects within milliseconds. Metawave technology can see pedestrians 250 meters away and recognize vehicles 330 meters away. It can also accurately measure *angular resolution*, or small distances between two objects, enabling the radar to distinguish one object from another.

Conventional digital radar systems capture all the information at once, like a powerful flashbulb illuminating a scene. Metawave's radar works more like a laser beam able to see one specific section of space at a time. The beam rapidly sweeps the environment, detecting and classifying all the objects in the vehicle's field of view within milliseconds.

The technology gives cars self-driving features such as left-turn assist, blind-spot monitoring, automatic emergency braking, adaptive cruise control, and lane assist.

SMARTER LIDAR

Light detection and ranging (lidar) sensing systems emit thousands of pulses of light every second. These pulses bounce off surrounding objects and reflect back to the vehicle, where a computer uses each data point, or *voxel*, to reconstruct a three-dimensional image of the environment.

Because inclement weather interferes with the signal, lidar is often combined with other sensing technologies, such as cameras, radar, or ultrasonics. But that combination can produce an overwhelming amount of redundant and irrelevant information.

Engineers at AEye, based in Dublin, California, have advanced the capabilities of lidar by fusing it with a high-resolution video camera. Their system, called iDAR, for Intelligent Detection and Ranging, merges the high-resolution pixels from a digital camera with the 3D voxels of lidar. Because the laser pulses and video camera gather optical information through the same aperture, the two data streams are integrated and can be analyzed at the same time, saving time and processing power.

Unlike traditional lidar systems, which scan a scene equally across the whole environment, iDAR adjusts its light-pulsing patterns to give key areas of the scene more attention. Computer vision algorithms determine where to direct the pulses. They first analyze the camera data to detect the edges of objects and then initiate the higher-resolution lidar scans to classify, track, and predict the motion of those objects.

Image credit: Metawave Corp.

Image credit: AEye, Inc.

HEAT WAVES

Conventional lidar is extremely accurate, but snow, rain, and fog reduce its ability to tell animate from inanimate objects. Traditional radar, on the other hand, can see through the snow, is excellent at long distances, and can judge the relative speed of objects, but it cannot distinguish what those objects are. Cameras can classify as well as read traffic lights and street signs, but glare can disrupt the quality, and at night, they can only see what the headlights illuminate.

Owl AI's team fills in the gaps with 3D thermal imaging, which senses heat signatures given off by people and animals and greatly simplifies object classification. Owl AI's Thermal Ranging™ can pick up the infrared heat of a living object. It sees the object, whether it's moving or stationary, by day or night and in any weather conditions, up to 400 meters away, and can calculate the object's 3D range and velocity up to 100 meters away.

The device consists of a main lens similar to the one in a regular camera and an array of very small lenses positioned between the main lens and a detector. The array breaks the scene into a mosaic of images, each one looking at the object of interest from a different angle. An algorithm measures the subtle differences between the images to calculate how far away the object is.

ENHANCING SAFETY

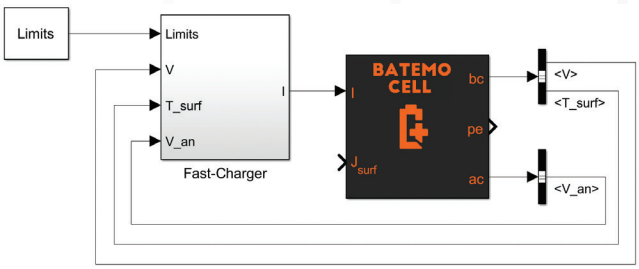
Vehicle perception technologies are key to providing a safe automated driving experience. To deliver on the promise of fully autonomous, self-driving cars, tech companies are using AI and computer vision to help vehicles see and sense their environment. And although fully autonomous cars aren't the norm yet, these companies are bringing us closer while improving the safety systems in new cars today. ◆

Control Design for Power Electronics: Batteries, Motors, and Power Conversion

The more-electrification trend in key industries has increased engineers' need for cohesive tools to design, prototype, and deploy their power electronics control systems. With Simulink®, Simscape®, and third-party products, engineers can design control algorithms graphically and then simulate them together with accurate models of batteries, motors, and power converters. They can perform rapid control prototyping (RCP) and hardware-in-the-loop (HIL) testing to verify these algorithms. They can then generate production-ready C or HDL code from the algorithms to run on MCUs, DSPs, and FPGAs.

Batemo: Batemo Cells

Batemo Cells are high-precision models of lithium-ion battery cells, including popular cells used in battery systems. The models are based on physical cell geometry and describe the inherent chemical processes even under extreme conditions where conventional models fail, including high currents, low states of charge, and extreme temperature ranges. The battery models can be incorporated into system models as Simulink blocks for real-time simulation and verification.



batemo.com

Microchip: Motor and Controller Model Libraries, MPLAB Device Blocks

Microchip provides blocksets for simulating and deploying motor and power control algorithms to PIC32 and SAM microcontrollers and dsPIC® digital signal controllers. The Motor Control Library Blockset contains Simulink blocks for motor control applications, including reference frame transforms, a proportional-integral controller, and trigonometric functions. The Motor Model Library adds a Simulink model for simulating permanent-magnet synchronous motors. For deploying control algorithms onto hardware, MPLAB® Device Blocks for Simulink provide peripheral blocks for digital and analog I/O, counters and timers, pulse width modulation motor control, and more. You can add these peripheral blocks to your Simulink models and then generate C/C++ code to run on PIC32, SAM, or dsPIC devices.

microchip.com/modeling

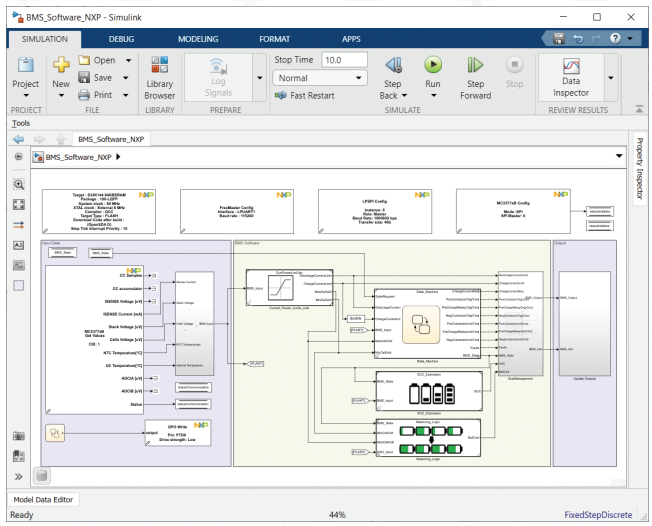
Speedgoat: Real-Time Target Machines

Speedgoat offers high-performance, real-time target machines for RCP and HIL testing of power electronics applications. Typical applications include prototyping battery management systems (BMS) for electric cars, HIL testing of motor controllers, and prototyping power converter controls. Engineers can deploy their Simulink control algorithms onto Speedgoat hardware to connect to power converters and motors. HIL testers can run deterministic Simscape-based models of electrical systems, including switching dynamics, power sources, and loads. Speedgoat offers specialized configurations for prototyping battery management systems, enabling engineers to quickly connect their hardware to complex electrical motor, battery pack, or drivetrain models.

speedgoat.ch

NXP: Model-Based Design Toolbox

NXP's Model-Based Design Toolbox is a toolchain for configuring and generating software to execute motor control algorithms on NXP processors. The toolbox provides a Simulink blockset for peripheral devices such as PWM, A/D, and CAN, as well as optimized motor control blocks for Park/Clarke transforms and digital filters. The toolbox is integrated with an Embedded Coder® target for generating and deploying code onto NXP processors and performing software-in-the-loop and processor-in-the-loop testing. The NXP Model-Based Design Toolbox supports Motor Control Blockset™ with examples that can be deployed to S32K microcontrollers.



nxp.com/mctoolbox

Learn More

Power Electronics Control Design
mathworks.com/power-electronics

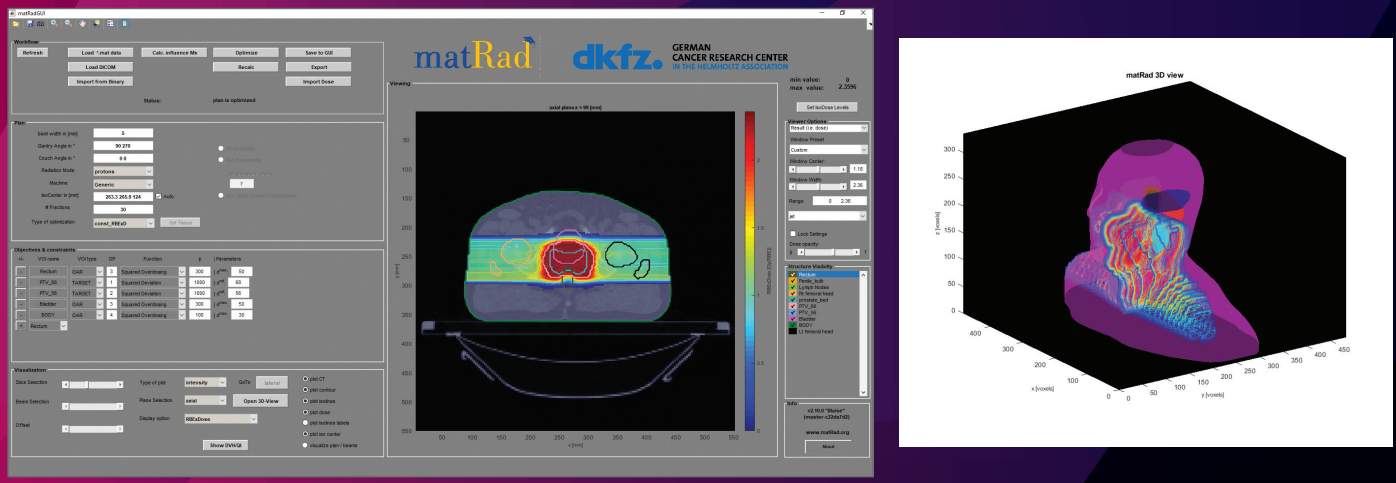
Third-Party Products and Services
mathworks.com/connections

Developing an Open-Source Toolkit for Radiation Therapy Planning

Clinicians who treat cancer patients with radiation rely on treatment planning software that optimizes the radiation dose to ensure tumor coverage while sparing surrounding tissue and organs.

Commercially developed treatment planning software is proprietary and closed source. As a result, many institutes and universities either invest significant effort in developing and maintaining their own or use open-source packages, most of which cover only a single step in treatment planning.

Researchers at the German Cancer Research Center (Deutsches Krebsforschungszentrum, or DKFZ) have developed matRad[1], a multimodality dose calculation and optimization toolkit for radiation treatment planning. Because matRad is written entirely in MATLAB®, researchers can easily modify the code to evaluate new algorithms. MATLAB excels at performing the many sparse matrix operations involved in treatment planning; as a result, matRad produces clinically accurate treatment plans as quickly and easily as its commercial counterparts.



Left: The matRad 2.10.0 interface, with workflow, plan, optimization, and visualization controls.
Right: 3D rendering of computer tomography and planned proton dose in the coronal plane of a head-and-neck cancer case.

The matRad package includes MATLAB scripts, functions, and classes that span the entire treatment planning workflow, from setting treatment parameters and optimizing the plan to visualizing and evaluating the results.

matRad remains under active development, and the team regularly accepts pull requests from researchers. For example, they recently worked with Dr. Edgardo Dörner at Pontificia Universidad Católica de Chile to incorporate a Monte Carlo photon dose calculation engine into matRad.

Since matRad was designed as a research tool, it cannot be used to treat actual patients. The dose calculations it produces, however, closely match those produced by clinically approved treatment planning systems. This level of performance opens opportunities to use matRad as an independent tool for validating treatment plans generated by other software.

[1] The current release is matRad 'Blaise' 2.10.0. *http://doi.org/10.5281/zenodo.3879616*

Developing Wind Power Plant Control Software with Model-Based Design and Continuous Integration

With more than 66,000 turbines totaling more than 100 GW of installed wind power capacity in 80 countries, Vestas Wind Systems A/S has installed more wind power than any other company. Vestas engineers use Model-Based Design with continuous integration (CI) to develop power plant control software and demonstrate compliance with grid codes to Vestas customers and grid operators.

Before adopting Model-Based Design for power plant control design, Vestas engineers used a conventional approach in which paper-based specifications and design documents developed by power engineers were handed off to software engineers, who wrote code for individual components or features manually. The power engineers ran simulations using PSCAD software, but these simulations focused on electrical power rather than software control. The simulations did not incorporate the control code, which meant there was little assurance that the PSCAD simulations reflected the system performance once the software was integrated and deployed. Vestas wanted to eliminate the potential for human error that comes with handwriting code while ensuring that its power systems simulations corresponded with the control software.

In addition, Vestas wanted to enable an engineering team that spanned five countries in Europe and Asia to work together on the same projects—and in some cases, on the same models. This geographically dispersed team needed to apply version control to models, manage frequent merges, and automate simulation-based tests.

Establishing a New Workflow

To meet these requirements, Vestas decided to use CI with Jenkins™ and to incorporate CI principles into an engineering workflow based on modeling, simulation, and code generation (Figure 1).

Today, when a grid code change is proposed or a customer requests a new feature or component, Vestas engineers create a set of formal requirements. Based on the requirements, one group develops test cases with Simulink® and Simulink Test™ that will be used to verify the new feature, while a second group models the new feature in Simulink and Stateflow®.

To create a system model for closed-loop simulations, engineers in this second group combine the control model with a Simulink model that captures the impedance and dynamic characteristics of the grid at the point of connection with the plant. They incorporate a wind turbine model developed in a proprietary tool and packaged as a DLL by another Vestas group.

> **"We can show our customers and grid operators a simulation that incorporates the actual code that will run in our power plant controller. That's what grid operators want, and it gives Vestas an advantage over competitors who still use conventional approaches."**
>
> – Per Hagen Nielsen, Vestas

After running closed-loop simulations with this system model and running checks to ensure compliance with modeling standards based on MathWorks Automotive Advisory Board (MAAB) guidelines, the engineers check the control model into a Git repository. The model check-in triggers a Jenkins job that runs the test cases developed earlier with Simulink Test, as well as additional simulation-based tests created by the power engineers and another round of modeling guideline compliance checks.

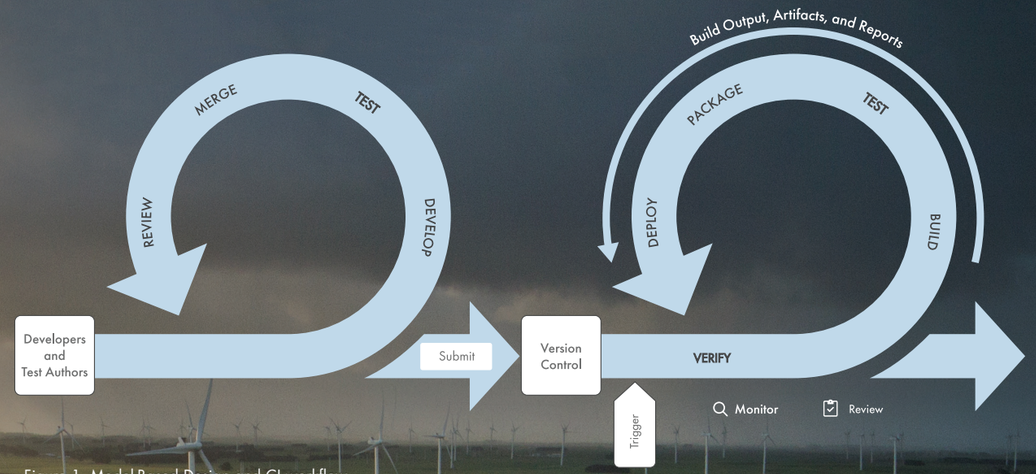


Figure 1. Model-Based Design and CI workflow.

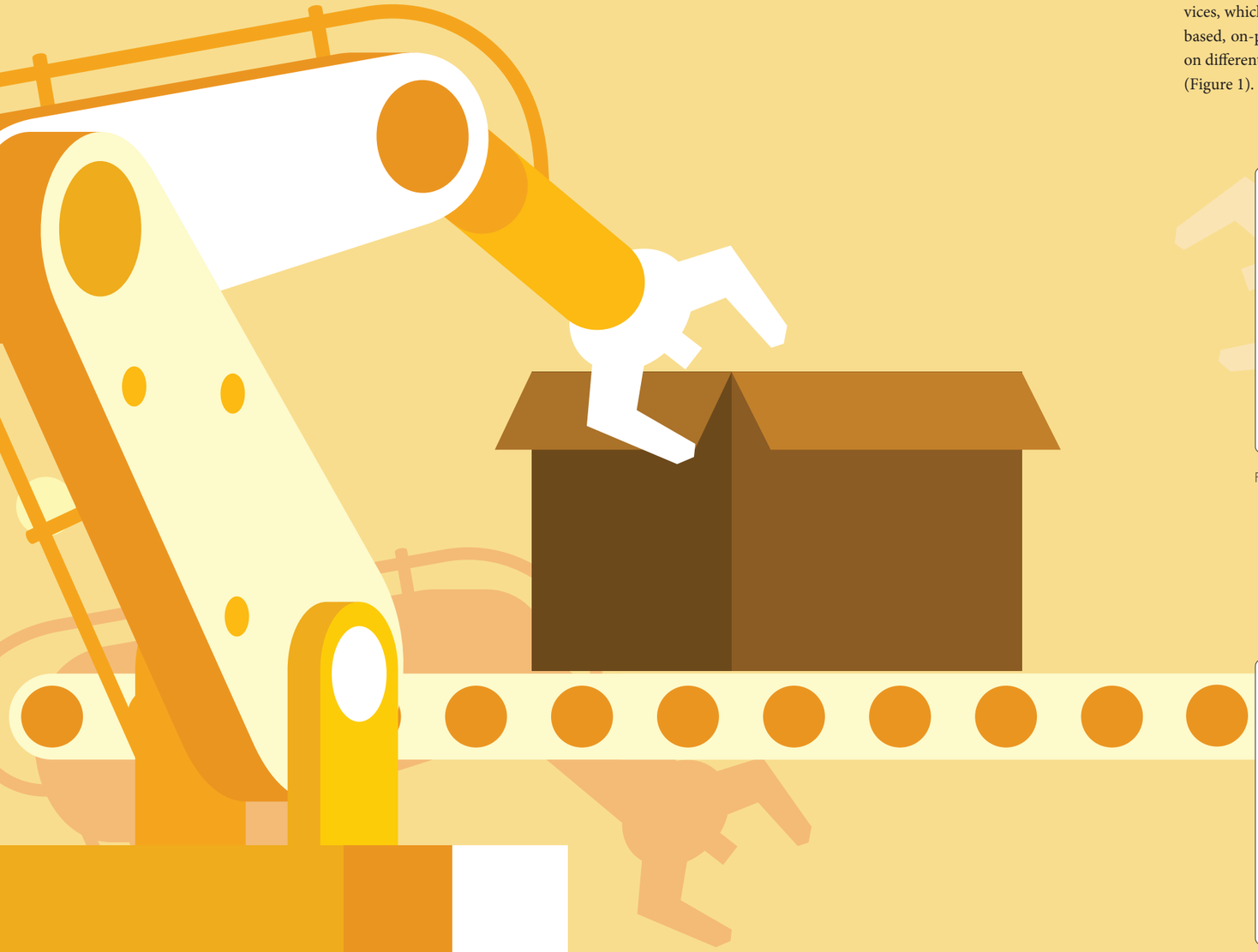
If the control model passes all tests and checks, Jenkins invokes Embedded Coder® to generate C++ code from the model. The generated C++ code is compiled into a DLL, which is then used in PSCAD to run simulations of the full plant and its control software.

Vestas uses these simulations to demonstrate to transmission system operators how the plant will perform when connected to the grid under normal conditions and in the presence of voltage drops, oscillations, and other disturbances. Finally, the generated code is tested on the target industrial control system before being deployed into production. ◆

Image courtesy of Vestas Wind Systems A/S

Deploying Predictive Maintenance Algorithms to the Cloud and Edge

By Aditya Baru, MathWorks



For organizations that manufacture or operate industrial machinery, a predictive maintenance program is key to increasing operational efficiency and reducing maintenance costs.

At the same time, however, developing and deploying predictive maintenance algorithms to any asset, whether to an aircraft, an MRI machine, a wind turbine, or an assembly line, can be challenging. Algorithm development requires not only extensive experience in machine learning techniques but also deep understanding of the system's behavior. Engineers possessing both these skills can be hard to come by. Deployment, meanwhile, involves a complex series of steps and interconnections. The algorithm must be implemented on multiple assets. Those assets will be connected to multiple edge devices, which in turn connect to an IT/OT system that may be cloud-based, on-premise, or both. Portions of a single algorithm may live on different elements of this infrastructure, adding to the complexity (Figure 1).

Using a packaging machine as an example, this article shows how to handle these complexities by developing a predictive maintenance algorithm and deploying it in a production system with MATLAB®.

Packaging Machine Maintenance System

The packaging machine has several robotic arms (Figure 2, left). The arms move back and forth at high speed, moving objects onto the assembly line for packaging. They are connected to programmable logic controllers (PLCs) that communicate with a Microsoft® Azure®-based IT/OT system. This IT/OT system collects streaming data from the edge devices connected to the robotics arms, runs predictive maintenance algorithms based on this data to detect anomalies and predict when they might fail, and returns the results to dashboard tools used by engineers and operators.

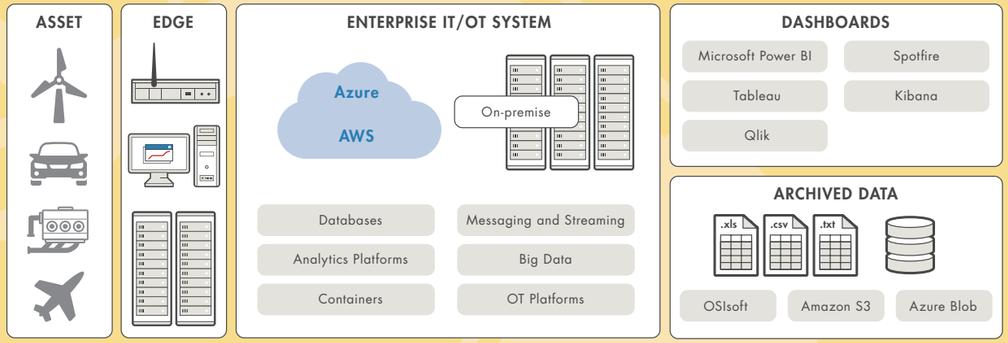


Figure 1. Components of a deployed predictive maintenance system.

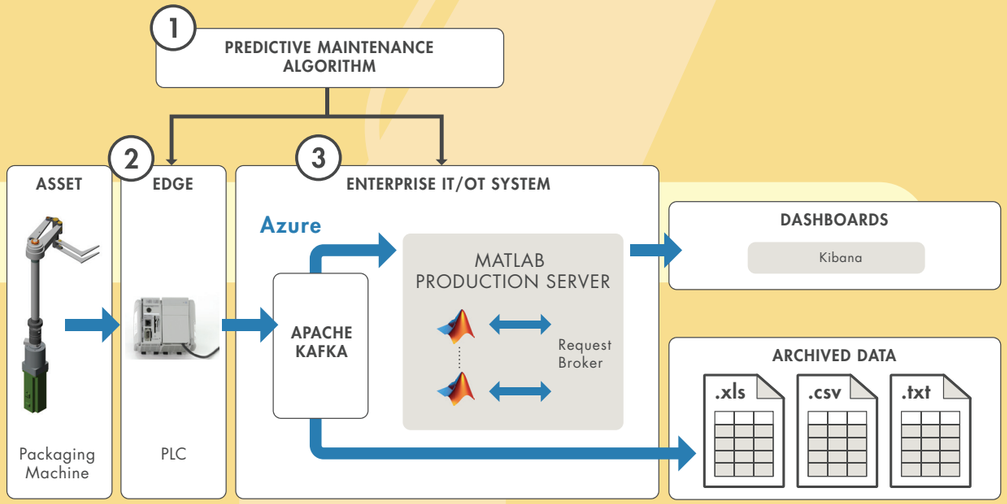


Figure 2. Packaging machine predictive maintenance system.

Figure 3. The Diagnostic Feature Designer app.

The Predictive Maintenance Algorithm

The predictive maintenance algorithm for this system has two components. The first is implemented on the edge and performs data reduction using feature extraction techniques. The second is implemented in the cloud and uses these feature values and a machine learning model to predict when a failure will occur and to estimate the machine's remaining useful life (RUL). The results of this predictive algorithm are streamed into our dashboard in near real time.

Developing the Data Reduction Algorithm

The first part of our predictive maintenance algorithm acts on the raw sensor data generated by the robotic arms. We are tracking the speed and the current drawn by the motor driving each arm.

The sensors used for machines like these can sample data at a very high rate. Storing such vast amounts of sensor data can be expensive, and analyzing this data is time-consuming, as the sheer volume makes it hard to identify regions of interest. We can solve this problem with feature extraction.

Feature extraction techniques accept streams of raw sensor data and return a smaller set of features that capture key dynamics, significantly reducing storage and transmission needs. The sensors in the robotic arm capture data at 1 KHz—that is, at 1000 samples per second. Condensing one second's worth of this data to a set of five features will reduce our data storage and transmission needs by a factor of 200.

Using the Diagnostic Feature Designer app in Predictive Maintenance Toolbox™, we import the sensor data, extract features using signal-based and dynamic-modeling techniques, and rank the features by their ability to distinguish data generated by a healthy machine from that generated by a faulty machine (Figure 3).

Once we have selected the features we want to extract, we are ready to implement and test the data reduction algorithm on the PLC that acts as our edge device. Instead of testing the algorithm on a real machine, which could damage the machine, we connect the PLC to a Simscape model of the robotic arm running on a Speedgoat real-time computer. This real-time computer can communicate with our PLC by sending and receiving data as if it were an actual machine. We begin by generating C code for the data reduction algorithm with Simulink Coder™ and deploying it to the PLC. We then deploy our packaging machine model to the Speedgoat system and perform simulations under different fault conditions to ensure that our algorithm will work correctly in a real-world environment (Figure 4).

Developing the Predictive Algorithm

We now have an edge device that reduces the amount of data being transmitted by extracting meaningful features from it. We can stream the reduced dataset into our IT/OT system using Apache™ Kafka, an open-source stream processing platform running in the Azure cloud. We will use this streaming data to estimate the RUL of the packaging machine motors.

As the condition of the motor deteriorates over time, the values of the extracted features will increase or decrease steadily, at a linear or exponential rate (Figure 5). Based on this trend, we select an exponential degradation model in Predictive Maintenance Toolbox to predict the future health of the machine.

To make this algorithm compatible with a cloud-based system, we use MATLAB Compiler SDK™ to create an executable, which we then integrate into the IT/OT system using MATLAB Production Server™ (Figure 6).

We now have machine learning algorithms that predict failures in our packaging machine using features extracted from raw data by edge devices connected to individual robotic arms, and a web-based dashboard that gives us immediate access to the results (Figure 7). ◆

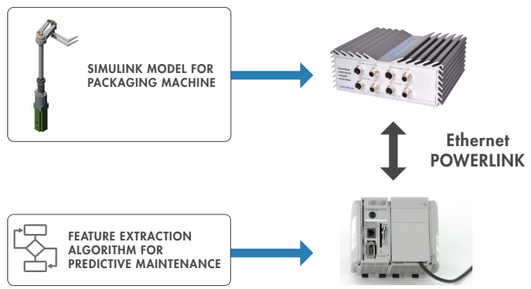


Figure 4. Deployment to PLC and testing in real time using Speedgoat hardware.

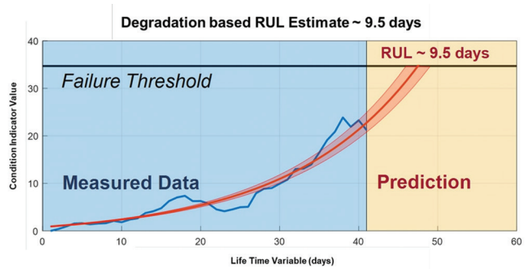


Figure 5. Sample RUL plot of streaming data.

Case Study: IMA Active

IMA Active designs and manufactures automatic processing and packaging machinery for the pharmaceuticals industry. The company wanted to develop a predictive maintenance system that would monitor the health of a tablet press production machine. The machine has critical moving parts that require precise lubrication. Too little lubricant causes stress and failure. Too much lubricant can cause leakage into the final product.

The predictive maintenance system would use data obtained from the two sensors already available on the machine and would be self-teaching, with no need for external intervention.

IMA Active engineers used Predictive Maintenance Toolbox to develop algorithms for the system. They began by extracting features from the two sensors—a total of 36.

They extracted, visualized, and ranked features from the sensor data using the Diagnostic Feature Designer app in Predictive Maintenance Toolbox. With these features they trained a fault classification model that uses machine learning techniques to estimate the health of critical moving parts in the tablet press.

The predictive maintenance system enables machine operators to optimize resource use and schedule maintenance activities according to production needs.

> "Using MATLAB tools, we managed to extract and select the best features to build a classification model. The most promising algorithm uses five features and has an accuracy of 89%."
>
> — Alessandro Ferri, IMA Active

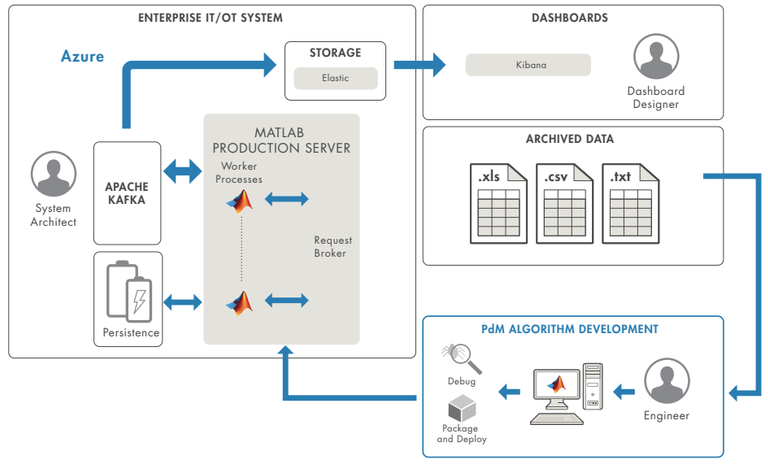


Figure 6. Cloud deployment overview.

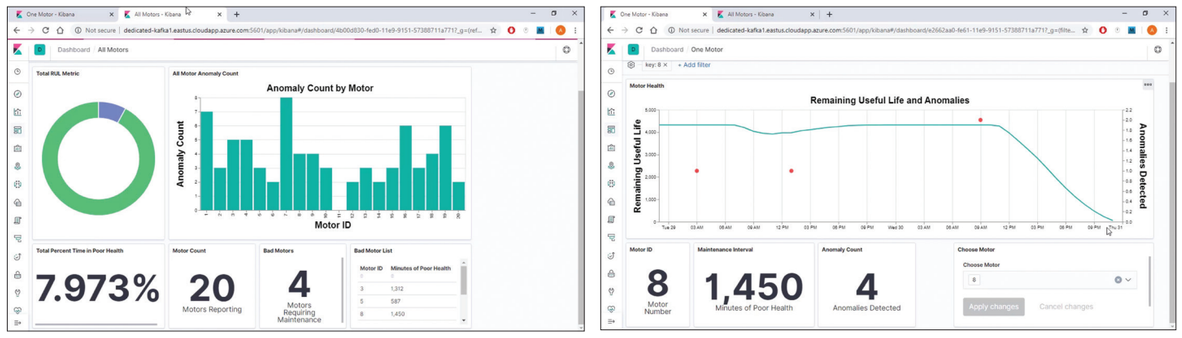


Figure 7. Final web dashboard.

Aiming Parker Solar Probe at the Sun with Simulink GNC Software

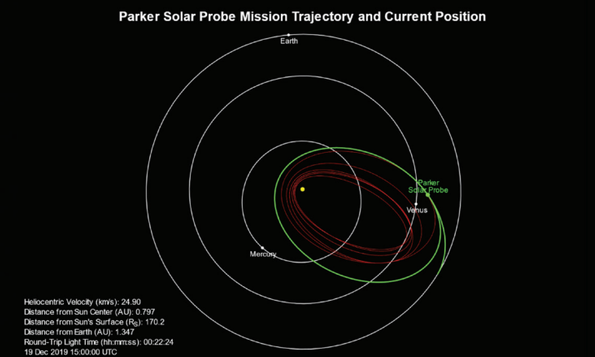
By Greg Drayer Andrade, MathWorks

Figure 1. Artist's rendition of the Parker Solar Probe approaching the Sun. Image credit: JHU APL. *parkersolarprobe.jhuapl.edu*

On Monday, November 5, 2018, Parker Solar Probe (PSP) reached its first perihelion, passing closer to the Sun's surface than any spacecraft had done before (Figure 1). Even at a top speed of about 213,200 miles per hour, it would take the spacecraft several days to pass behind the Sun and reemerge on the other side. During this time, researchers and engineers at NASA and at Johns Hopkins University Applied Physics Laboratory (JHU APL) waited anxiously for the first status beacon signal. On Wednesday, November 7, the signal was received: Parker Solar Probe was operating at status "A," with all scientific instruments running and gathering data.

Just under two weeks later, Parker Solar Probe reestablished full contact. As each subsystem was polled for its status, the APL teams' excitement grew. The science recorders had filled as expected, the spacecraft had maintained its attitude, and it was on the right trajectory. Over its nearly seven-year mission, Parker Solar Probe will orbit the Sun 24 times, coming gradually closer after each of seven Venus gravity-assist flybys until it passes within 3.83 million miles, close enough to fly through the Sun's atmosphere (Figure 2).

Confirmation that Parker Solar Probe had made first contact with the Sun was especially welcome news for the guidance, navigation, and control (GNC) team at JHU APL, which was responsible for developing the spacecraft's attitude control algorithms. Designed, implemented, and verified using Simulink®, these algorithms are mission-critical: they not only control the orientation of the spacecraft, they also keep its carbon-composite thermal protection system (TPS) pointed toward the Sun. A one- or two-degree deviation in orienting the TPS can mean the difference between a successful mission and destruction of the spacecraft.



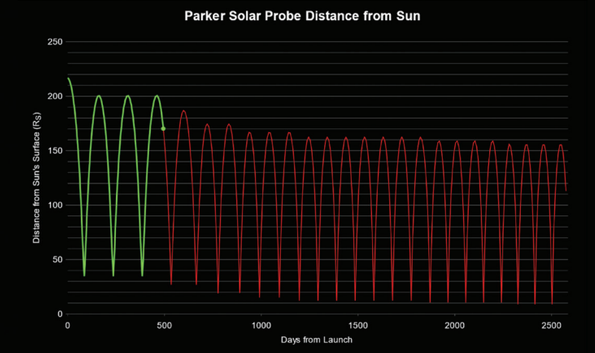


Figure 2. Graphs showing the Parker Solar Probe mission's planned path and solar approach distances. Image credit: JHU APL. *parkersolarprobe.jhuapl.edu*

Guidance and Control Design Constraints

In orbiting the Sun, Parker Solar Probe would be subjected to heat 475 times more intense than it would experience in an orbit of Earth. This meant that the attitude control system had to orient Parker Solar Probe so that it was continuously protected by the TPS.

Since the Sun is the biggest and brightest object in the solar system, at first it might seem simple to keep a spacecraft oriented toward it. In practice, however, attitude control for Parker Solar Probe is quite complex. One challenge is that near perihelion, none of the sensors that provide input data for the attitude control algorithms are actually pointed at the Sun. Instead, they are positioned behind the TPS to protect them from solar thermal radiation (Figure 3).

Two star trackers pointed away from the Sun can be used to gauge orientation from the positions of stars, but the design team had to plan for the possibility that these sensors would be unusable near perihelion. The spacecraft is equipped with two Digital Sun Sensors (DSS), but they can only be used far from the Sun. The Solar Limb Sensors (SLS) are designed for close range use, but they only detect the edge of the Sun when the spacecraft begins to rotate away from its ideal attitude. To develop a single fault-tolerant system for each segment of the orbit, it was vital to ensure that sufficient hardware was placed on the vehicle and incorporated into the control algorithms.

A second challenge was that the control algorithms had to make attitude corrections using as little electric power and thruster fuel as possible. Close to the Sun, Parker Solar Probe's solar panels remain almost entirely within the TPS shadow so that they do not melt. Extending the panels increases the pressure exerted on them, resulting in unwanted torque. Further, fuel for the spacecraft thrusters had to be used sparingly to ensure that supplies would last through the years-long mission.

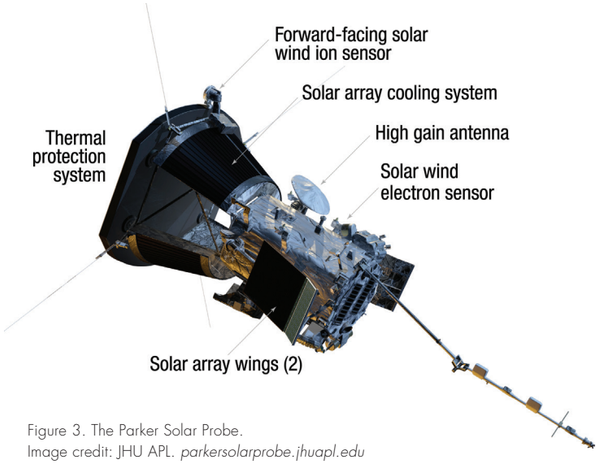


Figure 3. The Parker Solar Probe.
Image credit: JHU APL. *parkersolarprobe.jhuapl.edu*

Developing a Truth Model

The "truth" model for the spacecraft, built in MATLAB®, Simulink, and Simscape Multibody™, is essentially a plant model that captures orbital effects, physical interactions, and other spacecraft dynamics (Figure 4).

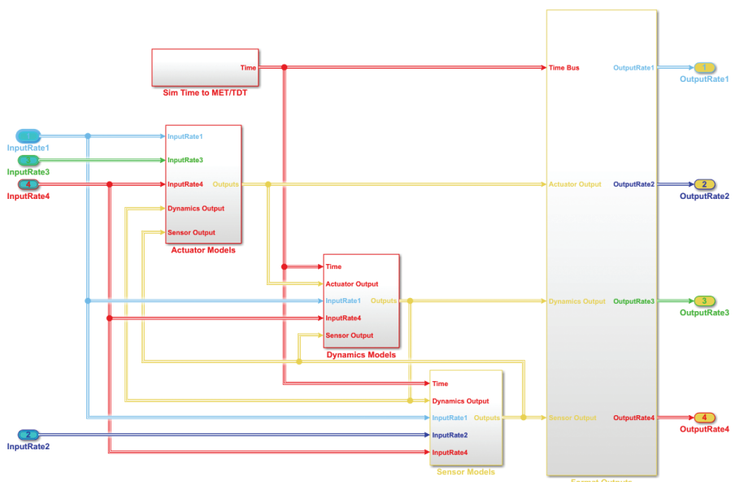


Figure 4. The Parker Solar Probe plant model, which consisted of nearly 1400 blocks and 1811 lines of MATLAB code.

Over time, many subsystems were incorporated into the model, including the battery subsystem, the thrusters, the star trackers, and the inertial measurement unit. The team also modeled the physical joints between the solar arrays and the main bus. As the model became more sophisticated, they were able to run increasingly accurate simulations. For example, they added a subsystem that models the effect of propellant slosh on spacecraft dynamics.

Developing GNC Flight Software

The initial attitude control system design did not include reaction wheels. Using only thrusters to manage momentum and make attitude corrections was one possible way to reduce weight and power consumption. To test the feasibility of this approach, the GNC team modeled several controller designs, including one with a pulse-width pulse-frequency modulator, and ran closed-loop simulations with the truth model. While the controller designs looked promising, there was no guarantee that the mission could be completed without the addition of reaction wheels. Fortunately, as the design matured, the team was able to make room for reaction wheels. This greatly simplified the overall design and allowed for improved accuracy and stability for scientific observations.

They created a system that non-propulsively manages momentum via the reaction wheels and fires thrusters to dump momentum when the wheels reach a specified momentum level. They reused much of the work from the thruster-only Simulink model in the redesigned controller. Altogether, the controller model included more than 22,000 blocks and almost 1200 lines of MATLAB code.

Because of the unforgiving environment that Parker Solar Probe would be operating in, an unprecedented number of simulations were performed. In fact, the number of formal simulations was increased by more than an order of magnitude compared with the previous mission led by JHU APL. Simulations covered both normal operating scenarios, including momentum dumps and trajectory correction maneuvers, and fault scenarios.

Most spacecraft are designed as fault-tolerant systems, but for this mission, solar conditions were more extreme than any previous spacecraft had experienced. For example, losing a star tracker on a spacecraft is considered a serious fault, but for Parker Solar Probe, it was necessary to plan for the likelihood that not just one but both star trackers could be blinded by a solar event and that additional faults could occur at the same time.

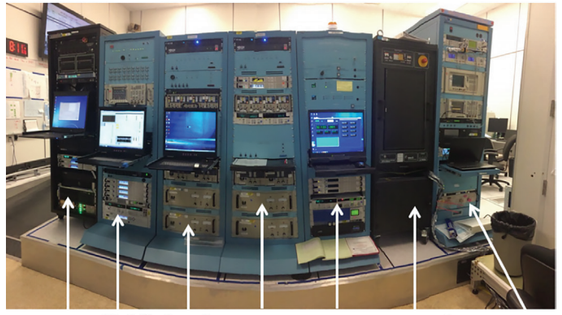


Figure 5. The JHU APL test bed.
Image credit: JHU APL. *parkersolarprobe.jhuapl.edu*

Figure 6. Parker Solar Probe launch.
Image credit: JHU APL. *parkersolarprobe.jhuapl.edu*

Code Generation and Test-Bed Verification

Initial verification of the controller design was performed via closed-loop simulations in Simulink. After generating code from the controller model using Simulink Coder™, the team ran software-in-the-loop (SIL) simulations in which the control model was replaced with the generated code.

After SIL testing and code optimization, the control design was verified on the JHU APL test bed (Figure 5). For this stage, code generated from the Simulink attitude control model was handed off to the flight software group, who incorporated it into the Parker Solar Probe flight software. The team also delivered code from the truth model to the test bed group, who integrated it with the test bed that emulates Parker Solar Probe hardware. Acceptance tests of the flight software were then conducted on the test bed. Closer to launch, more of the emulated components in the test bed were replaced with actual hardware components integrated on the spacecraft; for example, emulated reaction wheels were replaced with real ones.

Making In-Mission Adjustments

On Sunday, August 12, 2018, Parker Solar Probe was launched with a Delta IV Heavy rocket from Cape Canaveral Air Force Station, Florida (Figure 6). In addition to relaying scientific data back to Earth, the spacecraft is sending telemetry data, which the APL team analyzes and compares with simulation results in Simulink. They have already refined and calibrated their truth model based on these comparisons.

The spacecraft, including the attitude control system, was designed to operate autonomously, in part because it can take more than 15 minutes for radio signals to reach it from Earth. Nevertheless, there are three ways to make in-mission adjustments: send commands to execute preplanned maneuvers or actions, modify flight software parameters, or update the flight software itself. Since launch, the team has performed two software updates that incorporate improvements that were verified using the updated truth model.

As the mission continues, Parker Solar Probe orbits will become tighter and the time between orbits shorter. The APL team is developing MATLAB automation tools that will enable them to rapidly analyze new data from the spacecraft and respond quickly enough to make any needed changes before the next flyby. From the GNC team's perspective, the control software has been performing very well—in fact, it has far exceeded expectations. ◆

An Experiment in Deep Learning with Wild Animal Trail Camera Data

By Cleve Moler, MathWorks

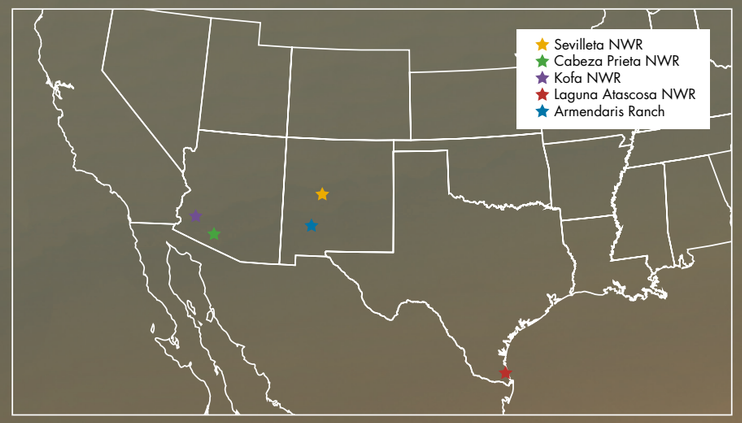


Figure 1. The five data site locations.

Trail cameras are automatically triggered by animal movements. They are used by ecologists and wildlife managers around the world to study wild animal behavior and help preserve endangered species. I want to see if MATLAB® image processing and deep learning can be used to identify individual animal species that visit trail cameras. We are going to start with off-the-shelf functionality—nothing specialized for this particular task.

My partners on this project are Heather Gorr and Jim Sanderson. Heather is a machine learning expert at MathWorks. Jim was one of my Ph.D. students at the University of New Mexico. He spent several years at Los Alamos National Laboratory writing Fortran programs for supercomputers. But an interest in nature photography evolved into a desire to switch to a career in ecology. He is now the world's leading authority on small wild cats. He is also the proprietor of CameraSweet, a software package used by investigators around the world to classify and analyze their trail camera data.

Our Data

Our study uses data collected by the United States Fish and Wildlife Service (FWS) office in Albuquerque over the past 10 years from four National Wildlife Refuges (NWRs) and one private ranch. The map in Figure 1 shows the locations of the sites.

Most of the data comes from the Sevilleta NWR, a 230,000-acre protected area in the Chihuahuan Desert in central New Mexico. Another site, also in New Mexico, is the Armendaris Ranch, 350,000 acres of private land owned by Ted Turner, the billionaire founder of CNN and former husband of actress Jane Fonda.

There is a lot of data—almost 5 million images in total. Sevilleta and three other NWRs contributed almost 4 million images that have already been classified by human experts, while the Armendaris Ranch and the Laguna Atascosa NWR in Texas contributed an additional million images that have not yet been classified.

CameraSweet has the researcher save images in a collection of folders, one folder for each camera, with subfolders for each species and for the number of animals seen in a single image. Each image file is labeled with the date and time when it was recorded.

Figure 2. A mother bear and her cub, captured by camera 28.

To read the Fish and Wildlife Service data, our MATLAB program creates a string array, **FWS**, of length 3,979,549, containing the path names of all the images in the dataset. For example,

```
k = 2680816;
example = FWS(k)

    example = "D:SNWR\Pino...
       South (28)\Bear\02\...
       2012 06 10 14 06 ...
       20.JPG"
```

The **FWS** entry for this example tells us that the data lives on a hard drive attached to my drive **D:** and that it comes from site SNWR, or Sevilleta NWR. The camera is number 28, located at Pino South. A human expert has saved this data to the CameraSweet folder for two bears.

I searched through many two-bear images, looking for a cute one, and found the one shown in Figure 2. The name of the JPG image is a time stamp for June 10, 2012, 14:06:20 hours. The command

```
imshow(example)
```

accesses the data and produces Figure 2.

Naming and Labeling Species

Researchers using CameraSweet have some flexibility in the way they name species. "Mountain Lion" and "Puma" are the same animal. There are several different spellings of "Raccoon." We have unified the names into 40 that we call *standard*. These names are the row and column headings in Figure 7.

Our MATLAB program creates a second string array, **`Labels`**, that has the standard names for the images in **`FWS`**. Using **`Labels`**, it is easy to count the number of images for each standard species. Here are the first and last three entries in this census.

```
images    percent    species

1282762   32.23      Mule Deer
690131    17.34      Pronghorn
407240    10.23      Elk
...
1909      0.05       Skunk
1659      0.04       Badger
1402      0.04       Barbary
                     Sheep
```

Two species, "Mule Deer" and "Pronghorn," together account for almost 2 million images, which is half of our data. The species "Ghost" describes the situation where something triggers the camera but there appears to be nothing in the image. Ghosts were discarded in the Sevilleta data, but the other sites provide plenty. The name "Few" is a catch-all for 10 species, including "Ocelot" and "Burro," that have fewer than 1000 images.

Overall, there is a huge disparity in the extent to which different species are represented in the data. A word cloud provides a good visualization of the species distribution (Figure 3).

The Trail Camera Images

Some of the images provide excellent portraits of the animals. Figure 4 shows four examples.

Javelina are found in Central and South America and the southwestern portions of North America. They resemble wild boar but are a distinct species. Pronghorn and coyotes are common at most of our sites. Nilgai are ubiquitous in India, where Hindus regard them as sacred. They were introduced into Texas in the 1920s. The only place they are found in our sites is Laguna Atascosa NWR.

About one-third of the images were taken at night with infrared, and appear in grayscale, like the top two examples shown in Figure 5.

The two oryx images were easily classified by human experts, even though the images show very different views. Traditional image processing techniques, which would look for features like number of legs, presence and style of antlers, and type of tail, would be baffled by the badly lit closeup on the bottom right.

There are thousands of images triggered by nonwildlife activity, including humans, cows, horses, vehicles, and domestic animals. In Figure 6, the image on the upper right has been classified as a ghost.

The subject in the lower left is obviously too close to the camera. There are faint yellow specks in the image on the lower right that could be a swarm of small flying insects. Both images are classified as "unidentified."

Training Our Deep Learning Network

Inception-v3 is a convolutional neural network (CNN) that is widely used for image processing. We will use a version of the network pretrained on more than a million images from the ImageNet database. Inception-v3 is an off-the-shelf image CNN. There is nothing in it specifically for trail cameras.

We choose a random sample of 1400 from each of our 40 species and designate 70% (980) as training images and 30% (420) as validation images.

We let the training run overnight on a Linux® machine with a GPU (Xeon® E5-1650v4 processor, 3.5 GHz, 6 HT cores, 64 GB RAM, and a 12 GB NVIDIA® Titan XP GPU).

The resulting *confusion chart* (Figure 7) is a 40-by-40 matrix A where a(k,j) is the number of times an observation in the k-th true class was predicted to be in the j-th class. If the classification worked perfectly, the matrix would be diagonal. In this experiment, the values on the diagonal would all be 420. Nearness to diagonal is a measure of accuracy:

```
accuracy = sum(diag(A))/...
    sum(A,'all') = 0.8686
```

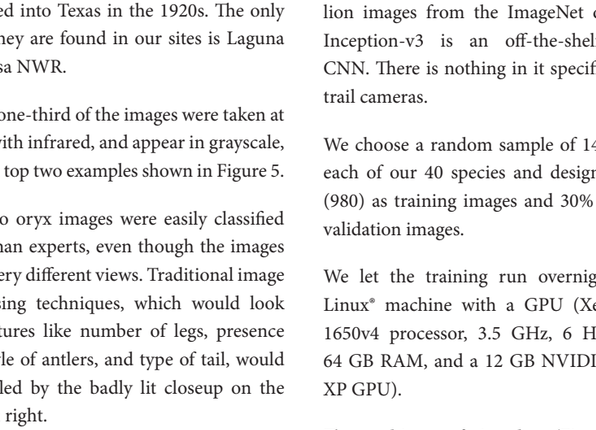


Figure 3. Word cloud showing relative distribution of species.

Figure 4. Example trail camera images. Clockwise from bottom left: coyote, javelina, pronghorn, and nilgai.

Figure 5. Top: two grayscale infrared images. Bottom: two full-color images of an oryx.

Figure 6. Images triggered by nonwildlife activity. Top: A human (left) and a "ghost" (right). Bottom: "unidentified" images.

Many of the large off-diagonal elements are not surprising. The smallest diagonal value, 216, is for birds. The row labeled Bird shows that the predicted class is often some other species. The next smallest diagonal element, 270, is for unknown. There is confusion between unknown and other species. Coyotes, with a diagonal value of 297, do not fare well, perhaps because they often appear in ambiguous images. Deer and nilgai, with diagonal values of 358 and 352, have good overall accuracy but are confused with each other.

On the other hand, the animals that are correctly classified the most often include the Barbary sheep, whose diagonal value is 405. Eagles, horses, and pronghorn are correctly classified 390 or more times. The bighorn sheep has a 386.

We now have a CNN trained to classify trail camera images. How does it perform at a new location with images that have never been classified?

We sample every tenth image from the Armendaris Ranch, a total of 18,242. The CNN classify function found 39 different species.

Almost half of the classifications—8708—were for bighorn sheep. Only 454 were for mule deer. This surprised me at first, because it meant that the network was predicting that Armendaris has almost 20 times as many bighorn sheep as mule deer, while Sevilleta, less than 100 miles to the north, has the opposite: 40 times as many mule deer as bighorn sheep.

But this result doesn't surprise Jim Sanderson. The mountains on Armendaris are the natural habitat for the sheep. The ranch manages the bighorn sheep in the same way that it manages its buffalo herds. Hunting bighorn rams is an important source of income for the ranch.

The CNN classification labels 93 images as pumas. This appears to be an overestimate. Examination of the 93 images reveals only 10 or 11 of the elusive animals.

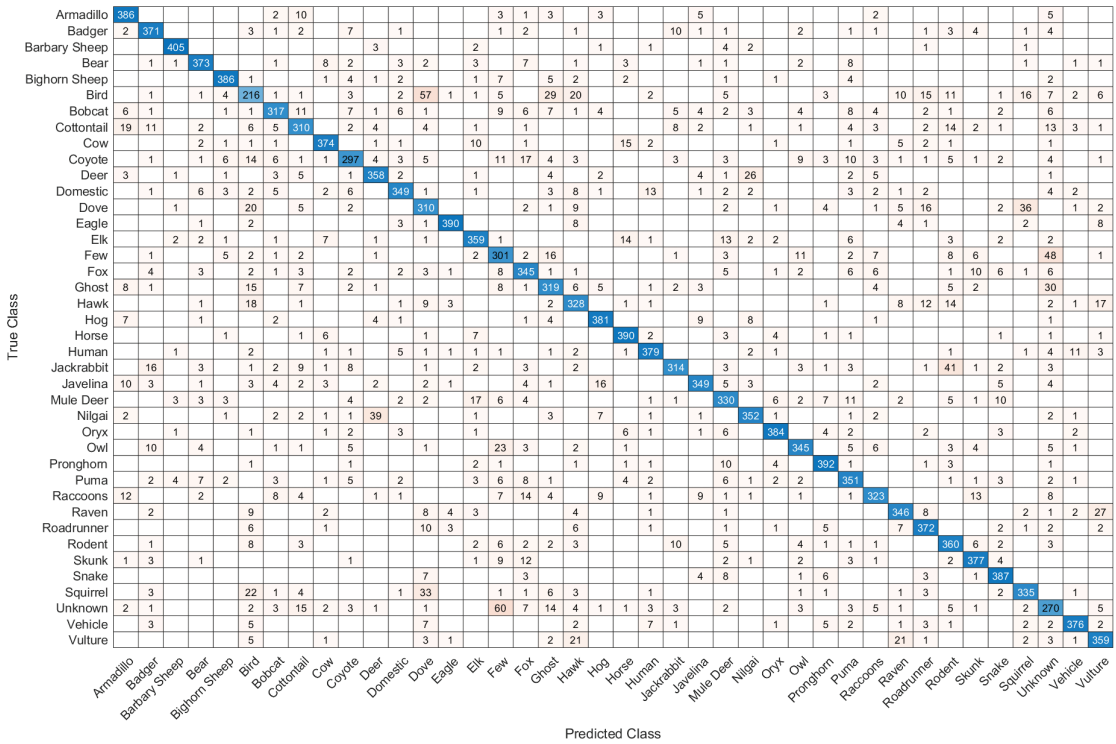


Figure 7. Confusion matrix, used to check the accuracy of the classifier.

Figure 8. Classification of previously unclassified images from the Armendaris Ranch. The upper images are classified by the CNN as bighorn sheep and puma, evidently correctly. The lower images are also classified as bighorn sheep and puma, apparently incorrectly.

All four images in Figure 8 are from Armendaris. The upper two are correctly classified by the CNN as bighorn sheep and puma, respectively. But the lower two receive the same classifications; that is clearly incorrect.

Conclusion

Overall, this deep learning classifier is more successful than I would have predicted. Even in its current state, it may be useful to investigators. Any further development of our CNN should focus on designing features specific to trail camera image identification.

One thing is clear—the more data the better. The 5 million images collected by the Fish and Wildlife Service were essential for training a network to this level of accuracy.

Acknowledgements

Thanks to Jim Sanderson, to Grant Harris at the Albuquerque FWS, and to Heather Gorr, Johanna Pingel, and the *MathWorks News & Notes* editorial team. ◆

ROBOTICS AND AUTONOMOUS SYSTEM BOUNDARY-BREAKERS

Enterprising startups and researchers envision robotic systems that can sense touch, enable telerobotic surgery, speak several languages, and more—and they are close to making that vision a reality.

Igor, a self-balancing two-wheeled robot built using the HEBI Robotics platform. Image credit: HEBI Robotics.

Performing Heart Surgery on a Patient Located Miles Away

The interventional cockpit, comprising a radiation shield, monitor, and control console. Image credit: Corindus.

Patients suffering an acute heart attack or stroke need immediate treatment in a hospital. Unfortunately, not all hospitals have physicians trained in percutaneous coronary intervention (PCI) or neurovascular intervention (NVI) procedures, and not all patients have ready access to a critical care unit.

Corindus has developed a robotic platform that makes it possible for surgeons to perform PCI and NVI on patients located hundreds or even thousands of miles away. Dr. Tejas Patel recently used the CorPath platform to perform the first long-distance telerobotic-assisted PCIs, completing five successful procedures over two days at the Apex Heart Institute in Ahmedabad, India.

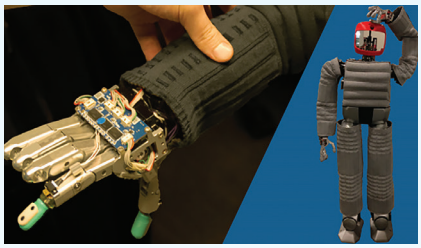
> *Every second matters for stroke victims, just as it does for those who suffer heart attacks. Our ability to treat patients wherever they are with our remote robotic protocol is the wave of the future.*
> Nicholas Kottenstette, Corindus

Giving Robots a Sense of Touch

Drexel University's Expressive and Creative Interactions Technology (ExCITe) Center equipped their in-house robot, Hubo, with padded, flexible outerwear to protect it from fall and collision damage.

Capacitive touch sensors in the sleeves give Hubo a sense of touch. The sensors are essentially circuits knitted with conductive yarn that acts as the wiring. When a person's skin presses the bare conductive yarn, Hubo identifies the location and pressure of the touch, enabling it to distinguish, say, a gentle tap on the shoulder from an aggressive push.

> *If you want a robot around your house who can take out the garbage or do the dishes, it needs to interact with people on a much more natural level than we're capable of right now. Sensing in all forms, but particularly touch, is a huge part of that.*
> Dr. Youngmoo Kim, Drexel University



Hubo's touch-sensitive sleeve (left) and full-body protective clothing (right). Image credit: Drexel University.

Building a Complex Robotics System in Days

HEBI Robotics hardware and software tools enable academic and industrial roboticists to build professional-grade, customized robots in weeks, or even days. HEBI kits provide prebuilt robotic systems, including a six-degrees-of-freedom robotic arm, a self-balancing two-wheeled robot, and a hexapod, as well as MATLAB scripts to control a single actuator or the assembled robot. After mastering the basics with a kit, researchers can quickly develop their own control applications by extending the example code or using it as a template.

A Robot That Walks, Talks, Dances, and Speaks 20 Languages

NAO is a two-foot-tall, humanoid robot designed by Aldebaran Robotics to interact in a natural way with humans. The robot can adapt to its environment; recognize shapes, objects, and even people; and speak clearly in more than 20 languages. NAO has been used in care homes and hotels, where it can greet residents, provide companionship, and perform tasks such as check-in and check-out. NAO also helps children with special needs. For example, therapists use it to teach social skills to autistic children, who are both reassured and engaged by the robot's playful yet predictable behavior.

> *The robot can provide guidance, motivation, and feedback, while creating a bond with the child who is using it.*
> Dr. Yu-Ping Chen, Georgia State University

Automating the Analysis of Satellite-Based Radar Imagery for Iceberg Surveillance with Deep Learning

By Kelley Dodge and Carl Howell, C-CORE

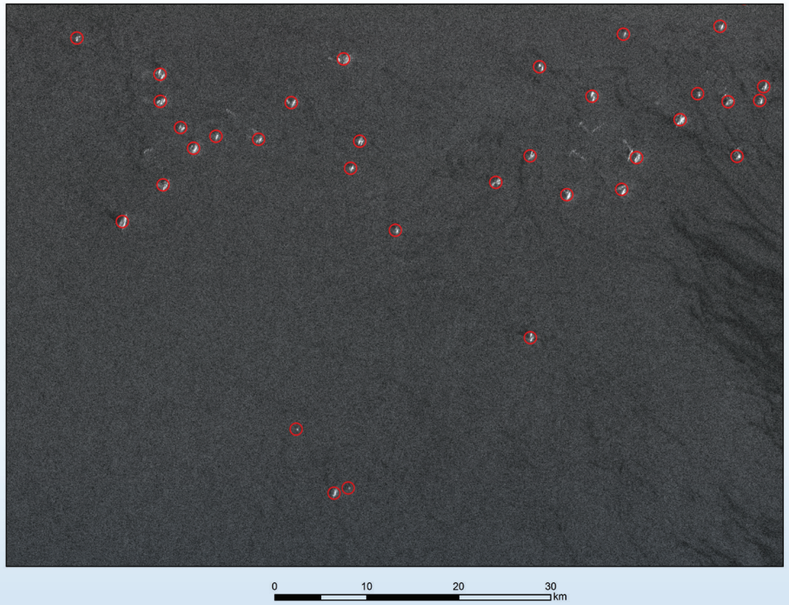


Figure 1. SAR image taken near Greenland, with targets circled.

On average, some 500 icebergs enter the Newfoundland and Labrador offshore area each year, posing potential threats to shipping and marine operations. In the 1990s, companies began using satellite synthetic aperture radar (SAR) to monitor icebergs and sea ice. SAR is well suited to the task because it can capture images from large swaths of ocean both day and night, as well as through clouds, fog, and other adverse weather conditions.

Analysis of SAR images involves identifying targets (clusters of high-intensity pixels) in the image and categorizing them as either icebergs or ships (Figure 1). Even for highly trained specialists it can take hours to analyze a handful of frames, particularly when the targets are difficult to discern.

Our team at C-CORE has partnered with the Norwegian energy company Equinor to develop new automated software that uses deep learning to classify targets in SAR images. We decided to harness the expertise of the worldwide community of AI researchers by hosting a Kaggle competition. We studied the best ideas from the competition, implemented them with convolutional neural networks (CNNs) in MATLAB®, and then built software that could be used operationally.

Challenges in Iceberg Identification

The resolution of a SAR image depends upon how much area the image covers: images that focus on relatively small regions have higher resolution than those that cover wide swaths of ocean and are therefore easier to classify (Figures 2 and 3). In practice, to extract the greatest amount of useful information from a dataset, we must work with images at all levels of resolution, even images with targets only a few pixels wide.

Before we began using deep learning, we used quadratic discriminant analysis for iceberg classification, but this involved segmenting the images to separate the target pixels from the background ocean pixels. Image segmentation was a challenge because ocean conditions vary widely, and the visual clutter caused by poor conditions made it difficult to define the contours of each target. With CNNs, there is no need to distinguish a target from its background, since the algorithms are trained on complete SAR *chips*, fixed-dimension images containing a single target.

The Kaggle Competition

Our Kaggle competition presented participants with a simple challenge: develop an algorithm capable of automatically classifying the target in a SAR image chip as either a ship or an iceberg. The dataset for the competition included 5000 images extracted from multichannel SAR data collected by the Sentinel-1 satellite along the coast of Labrador and Newfoundland (Figure 4). Our competition proved to be the most popular image-based competition ever hosted on Kaggle, with 3343 teams contributing more than 47,000 submissions.

The best-performing entries all used deep learning. Their models shared many common characteristics and layers, including convolution, rectified linear unit (ReLU), max pooling, and softmax layers. In addition, the top entries all used ensembles to boost prediction accuracy from about 92% to 97%.

Building Our Deep Learning Model with MATLAB

Using the top Kaggle entries as a starting point, we developed our own deep learning model with MATLAB and Deep Learning Toolbox™. We began by modifying a simple classifier provided in Deep Learning Toolbox. Within a few days we had a network that worked well.

To optimize network performance, we tested different combinations of parameter values, varying, for example, the number of nodes in each layer, the filter size used in the convolution layer, the pool size used in the max pooling layer, and so on. We wrote a MATLAB script that automatically built, trained, and tested 10,000 different CNNs, with values for these parameters randomly generated within reasonable limits and constraints.

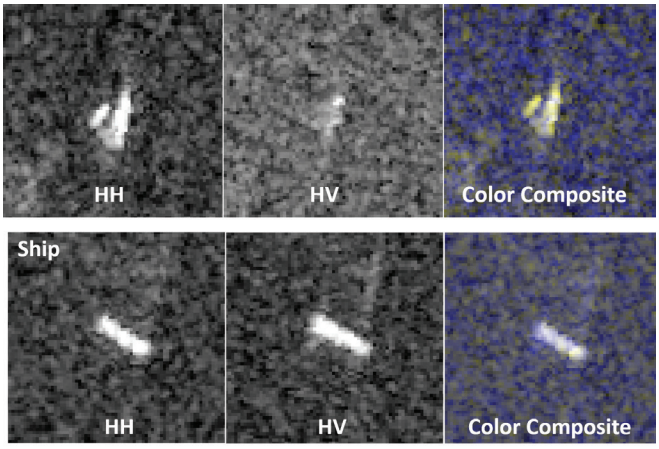


Figure 2. Color composite images of an easily classified iceberg (top right) and ship (bottom right) created from multiple polarization channels (labeled HH and HV).

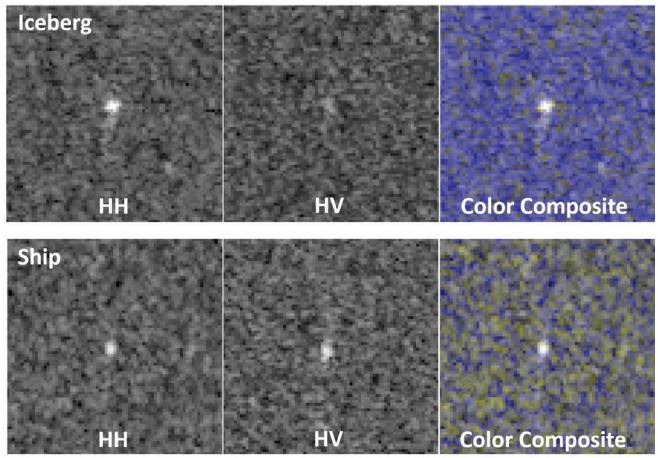


Figure 3. Color composite images of iceberg (top right) and ship (bottom right) that are difficult to classify.

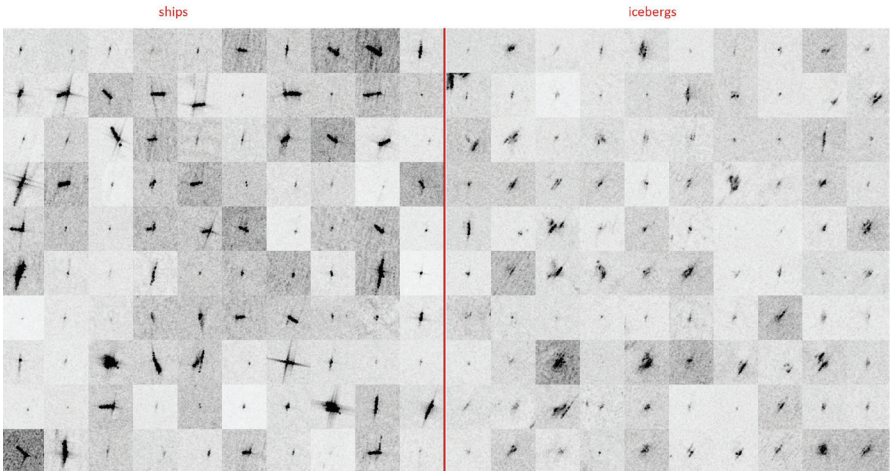


Figure 4. SAR data collected by the Sentinel-1 satellite along the coast of Newfoundland and Labrador. Adblu. "Statoil/C-CORE Iceberg Classifier Challenge: Ship or iceberg, can you decide from space?" Kaggle, 8 November 2017, *https://www.kaggle.com/c/statoil-iceberg-classifier-challenge/discussion/42108*.

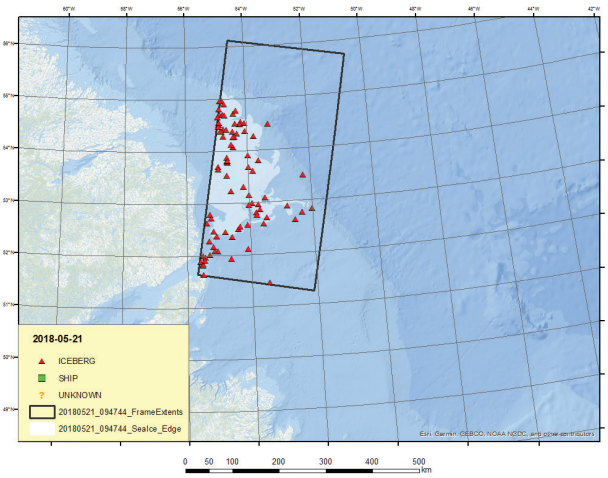


Figure 5. Iceberg locations plotted on a map.

We performed a simple greedy search on the results to find the seven highest-performing CNNs and used them to create an ensemble. Like the ensembles used by the Kaggle competition winners, our ensemble increased overall accuracy by almost 5%.

By working in MATLAB, within two weeks we went from knowing little about the implementation of CNN classifiers to producing a solution that performed well enough to be employed operationally.

Integrating the Classifier into a Complete System

Target discrimination is one step in a multistep process for iceberg identification. The process also involves land masking, to eliminate false detections caused by on-land objects, and integration with geographic information system software, to produce maps showing the locations of icebergs and ships (Figure 5).

When RADARSAT Constellation Mission satellites begin generating data this year, we will have access to even more SAR images—far too many for analysis via manual visual inspection. Software systems that incorporate deep learning algorithms like the ones we developed in MATLAB will enable C-CORE to make the most of this data by processing it accurately, quickly, and automatically. ◆

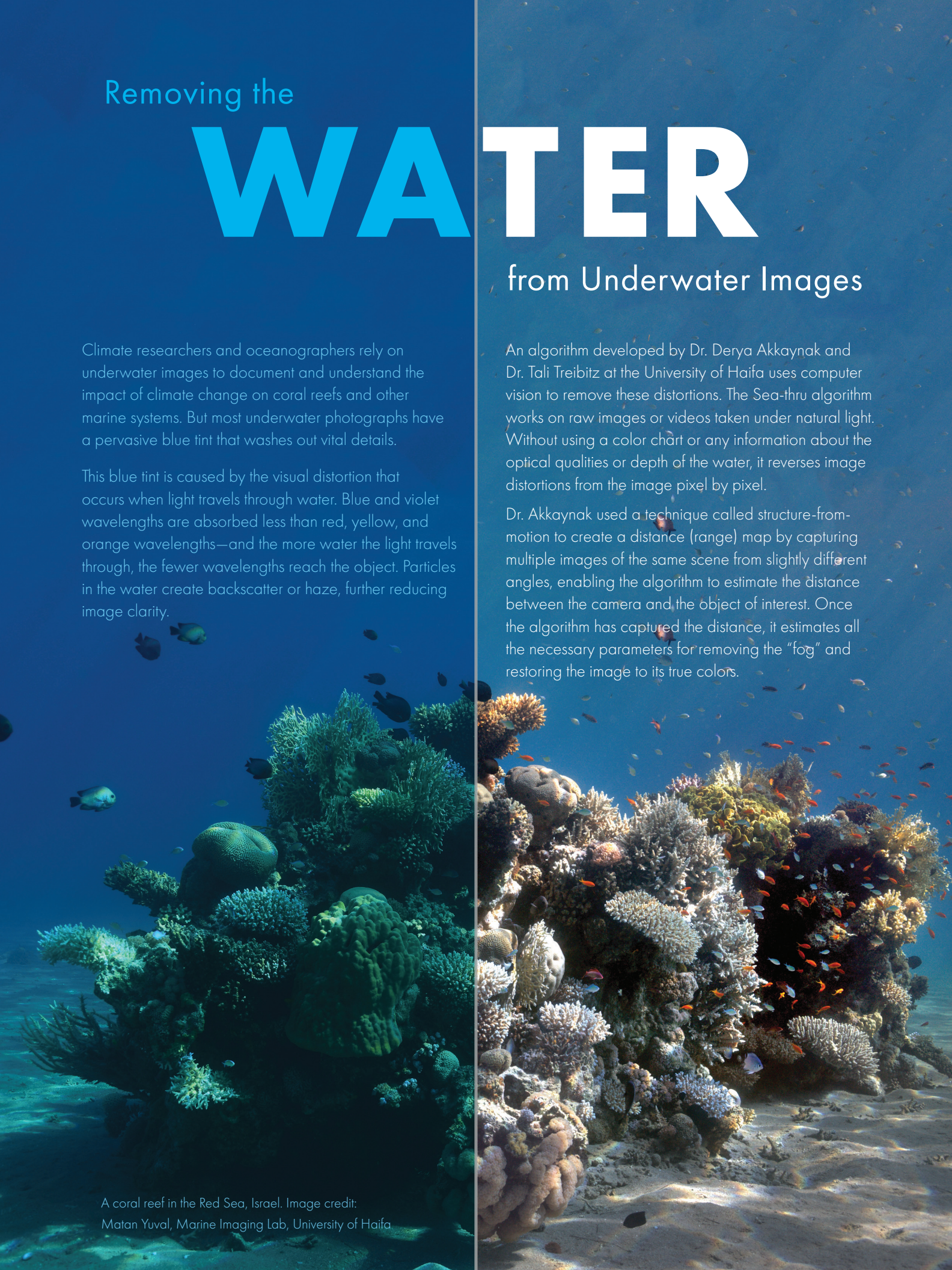
Removing the WATER from Underwater Images

Climate researchers and oceanographers rely on underwater images to document and understand the impact of climate change on coral reefs and other marine systems. But most underwater photographs have a pervasive blue tint that washes out vital details.

This blue tint is caused by the visual distortion that occurs when light travels through water. Blue and violet wavelengths are absorbed less than red, yellow, and orange wavelengths—and the more water the light travels through, the fewer wavelengths reach the object. Particles in the water create backscatter or haze, further reducing image clarity.

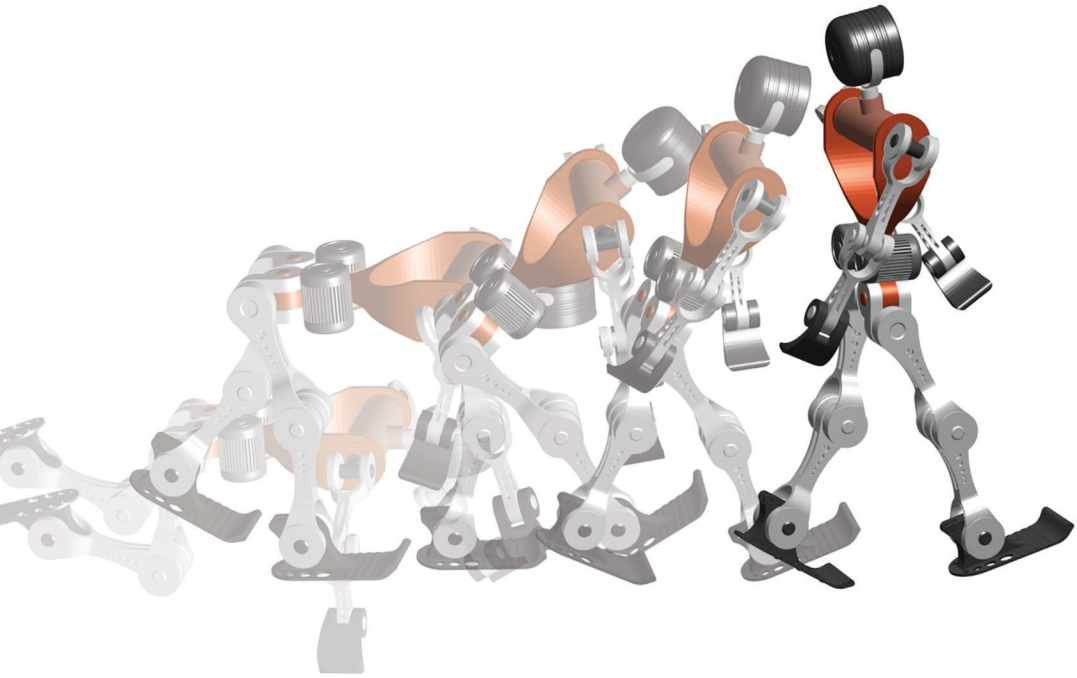
An algorithm developed by Dr. Derya Akkaynak and Dr. Tali Treibitz at the University of Haifa uses computer vision to remove these distortions. The Sea-thru algorithm works on raw images or videos taken under natural light. Without using a color chart or any information about the optical qualities or depth of the water, it reverses image distortions from the image pixel by pixel.

Dr. Akkaynak used a technique called structure-from-motion to create a distance (range) map by capturing multiple images of the same scene from slightly different angles, enabling the algorithm to estimate the distance between the camera and the object of interest. Once the algorithm has captured the distance, it estimates all the necessary parameters for removing the "fog" and restoring the image to its true colors.



A coral reef in the Red Sea, Israel. Image credit: Matan Yuval, Marine Imaging Lab, University of Haifa

MathWorks®



MATLAB SPEAKS REINFORCEMENT LEARNING

With MATLAB® and Simulink,® you can apply reinforcement learning to robotics, autonomous driving, and other systems. Quickly implement controllers and decision-making algorithms, create deep neural network policies, and model the environment and reward signals. Use reference examples and tutorials to get started right away.

mathworks.com/reinforcementlearning

93188v00 10/20