https://doi.org/10.2352/EI.2026.38.6.ISS-297

Optimizing Object Detection Performance with Complete Image System Simulations

Jonathan Brand, Eric Herman, Maria Ruiz Gonzalez, Leonard Zugby

Abstract

Optical system development requires software tools to design lenses, mechanical components, sensors, and image signal processing (ISP) pipelines. Historically, these tools are operated independently and do not provide insight into complete system performance. As a result, development teams often incur time and cost inefficiencies by designing, building, and testing hardware prototypes that either fail to meet requirements or significantly exceed them. Optical systems are therefore frequently over-designed in one or more areas—such as lens tolerances, sensor bit depth, or ISP complexity—to mitigate risk.

End-to-end simulation offers a path to eliminate these inefficiencies and accelerate time-to-market. In this work, we simulate a complete imaging system and demonstrate a method for identifying a minimally viable solution that meets the performance requirements of an object detection application. Using the imaging system simulator ImSym, we model the full imaging chain, including lens behavior, detector characteristics and noise, ISP routines, and straylight effects. These elements are combined to generate simulated images that enable validation of system performance prior to hardware fabrication.

Introduction

Products that incorporate optical systems are inherently complex and require multidisciplinary teams to bring them to market. These projects carry risks spanning requirements definition, system integration, and operational performance, all of which must be managed to ensure timely delivery and acceptable product quality.

Historically, development teams have employed systems engineering practices to manage this complexity, with the System VModel being a commonly used framework. The VModel requires teams to define requirements, build prototypes, and develop test methods to verify and validate system performance. However, building prototypes in academic and industrial environments requires significant investment of time and resources and therefore introduces substantial cost and schedule risk.

Within this framework, systems engineers communicate requirements to functional teams such as lens design, mechanical design, sensor design, and image signal processing. While intermediate design reviews occur, complete system validation typically does not take place until hardware is built, assembled, and image processing routines are tuned to the physical system. This workflow often leads teams to iterate through multiple hardware designs or to over-design components to reduce project risk, ultimately increasing manufacturing cost.

End-to-end imaging system simulation enables teams to explore broader design spaces, improve crossfunctional collaboration, and reduce time-to-market. There is a growing need for such simulation tools to support the next generation of imaging technologies. Recently, tools such as ImSym have emerged to address this need by enabling radiometrically accurate camera system simulation.

The purpose of this work is to investigate how a task-based, machine vision system for object detection can be optimized using an end-to-end imaging system simulation tool. Using ImSym, engineers can virtually evaluate lens designs, sensors, and image signal processing routines to create a digital twin prior to hardware verification. Automation through a Python interface allows rapid exploration of multiple configurations, including lens tolerances, ISP algorithms, and straylight conditions. In addition, we simulate illumination sources, sensor response and noise, and straylight artifacts to identify corner cases where system performance degrades. This enables corrective actions to be implemented in the virtual domain before hardware is built, significantly improving development efficiency and enabling first-time-right imaging system design.

Methods

Systems Engineering V-Model:

The systems-engineering V-Model provides a structured framework for developing products with complex requirements and establishes checkpoints for design reviews prior to progressing to higher-risk stages [1][2]. The V-Model, shown in Figure 1, begins with the identification of user requirements. These requirements may be qualitative or quantitative. For example, a super-resolution microscope may specify a modulation transfer function (MTF) requirement to enable visualization of features of a given size. In this work, we consider more generalized user requirements.

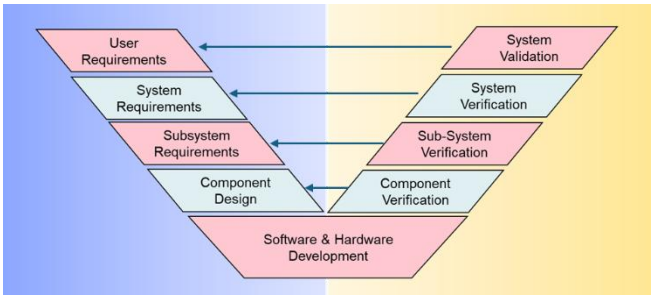


Figure 1: The Systems Engineering V-Model

The systems engineering V-Model provides a structured framework for developing products with complex requirements and establishes checkpoints for design reviews prior to progressing to higher-risk stages [1][2]. The V-Model, shown in Figure 1, begins with the identification of user requirements. These requirements may be qualitative or quantitative. For example, a super-resolution microscope may specify a modulation transfer function (MTF) requirement to enable visualization of features of a given size. In this work, we consider more generalized user requirements.

Our example focuses on an automotive camera system tasked with detecting and tracking vehicles in an image. From this user

requirement, system-level and subsystem-level requirements are derived before any components are designed.

Once requirements are defined, teams proceed to component design followed by hardware and software development. Verification tests are created at the component, subsystem, and system levels, with each test linked to a specific requirement. Ultimately, validation testing is performed against the original user requirements. Organizations must balance the stringency of their requirements against development risk; insufficient performance risks market rejection, while overly conservative requirements increase cost and development time.

Hardware development to support verification and validation is both costly and time-consuming. Projects are often constrained by the "triple constraint" of scope, schedule, and budget. Scope risk is closely tied to technical risk, particularly when requirements push the limits of available technology or are difficult to quantify. Budget risk arises from cost-of-goods constraints and high initial prototype costs. Schedule risk is driven by long hardware lead times, funding milestones, and limited resource availability.

For the automotive camera example considered here, the user and subsystem requirements are summarized in Table 1.

User Requirement	**Validated By:**
Detect cars in variety of lighting conditions	Less than 5% of cars frames missed
Sub-System Requirement	**Requirement**
Field of View	A diagonal half-field of view greater than 50° and less than 60°
Image Resolution	Greater than 50% Modulation at 100 cy/mm
TV Distortion	Average TV distortion less than 25%
Relative Illumination (Vignetting)	Relative illumination greater than 85%
Color Accuracy	An average $\boldsymbol{\Delta E}$ less than 5.0

Table 1 Sample Requirements for an automotive camera. Realistically this device will have many more requirements

ImSym: Imaging System Simulator

To create virtual prototypes, we used ImSym—Imaging System Simulator—to evaluate camera designs across multiple configurations. ImSym simulates camera data by computing a grid of point spread functions (PSFs) and using basis functions to model the scene spectrum, resulting in a final simulated image [3].

To demonstrate ImSym's accuracy, cameras with 35 mm and 25 mm focal length lenses were built, and their MTF was measured experimentally. The same camera systems were simulated in ImSym, and identical MTF measurements were performed on the simulated images. The results, shown in Figure 2, demonstrate strong agreement between measured and simulated data.

Sources of error include variation between as-designed and as-built performance. For commercial off-the-shelf (COTS) lenses, manufacturing and alignment tolerances are generally unknown, and as-built performance is typically worse than nominal design performance. These effects must therefore be accounted for during simulation.

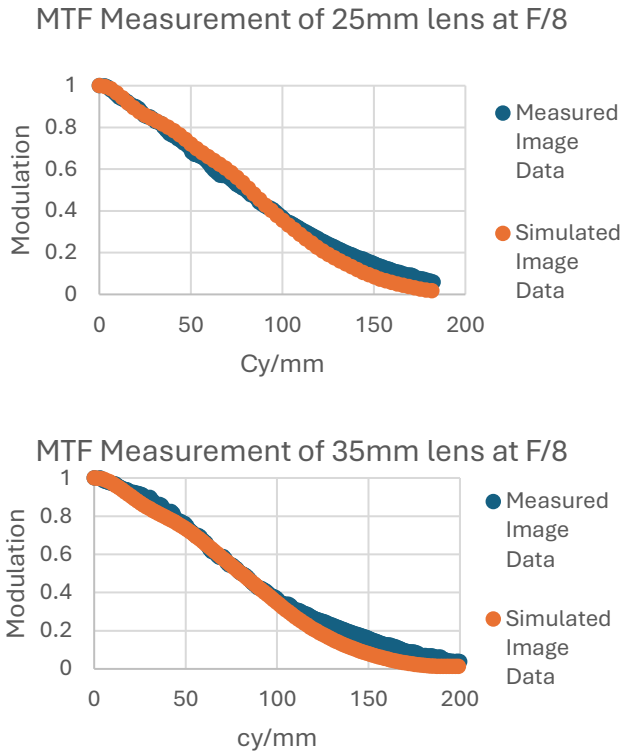


Figure 2 Validation of ImSym comparing real-world (Measured) MTF data against ImSym Simulated Data for two different lenses.

ImSym uses PSFs and other lens design data from CODE V to generate simulated images [4]. The lens selected for this study was designed for automotive applications. Figure 3 shows the lens layout, and Table 2 summarizes key design parameters.

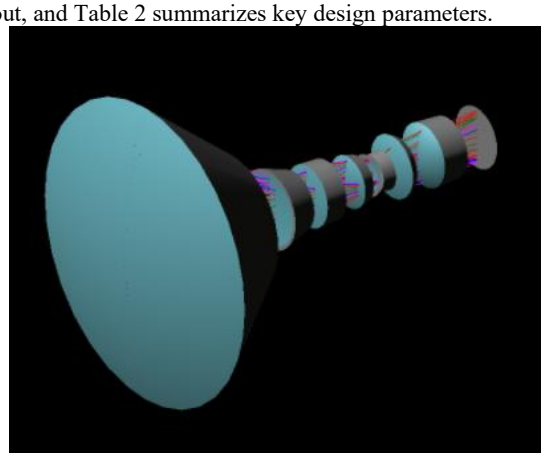


Figure 3 Initial Lens design for the camera design

Performance Specification	Value
F/#	2.5
Field of View	50°
Relative Illumination	92%

Table 2 Summary of Lens Specifications

Stray-light effects were incorporated by defining light sources in LightTools. Using ImSym's Stray Light Scanner, a point source was positioned at various locations across the camera field of view (FOV) [3][4]. Figure 4 shows a stray-light simulation for a source at infinity positioned at a 70° radial angle.

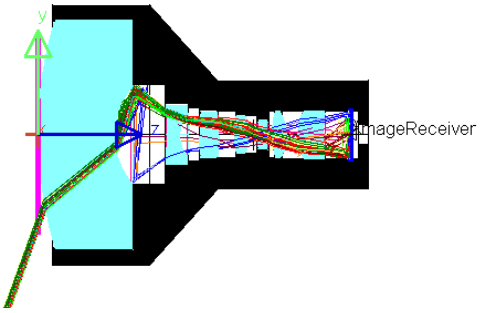


Figure 4 : 70° off axis stray light source result with 5,000,000 rays

For this study, a 780 × 520 pixel RGB sensor was selected. Sensor noise was modeled using detector noise settings, corresponding to 5,000 e^-/s dark-current mean and a 1,000 e^-/s standard deviation. This setting is recommended when starting the design of a general use camera. Users can define custom sensor parameters, including color filter arrays, quantum efficiency, spectral filters, noise characteristics, pixel size, and relative angular response.

Developing Virtual Prototypes

For verification, it is important to recognize that an as-designed system will typically outperform an as-built system. Therefore, tolerances must be incorporated to perform meaningful virtual prototyping. For this work, the following set of tolerances was assumed, as shown in Table 3. The tolerances were selected so variations in the simulation were observable. A sensitivity analysis was performed to identify the most critical parameters and determine whether tighter tolerances might be required.

Tolerance Type	Value
Surface Power	3 Fringes
Surface Irregularity	1 Fringe
Refractive Index	± 0.0005
Abbe	± 0.5 %
Lens Center Thickness	± 15 µm
Lens Wedge	8 µm TIR
Air Spacing	± 15 µm
Lens Decenter	± 10 µm
Lens Tilt	8 µm TIR

Table 3 Summary of Lens Tolerances

Once tolerances were defined, a Monte Carlo simulation was performed to analyze MTF performance at 100 cycles/mm. The design modulation at 100 cycles/mm was 79%, well above the requirement of 50%. This analysis provided an estimate of expected yield and informed subsequent design decisions. The tolerance analysis also generated a set of as-built lens instances with random perturbations, which were used for final verification.

ImSym supports simulation of two-dimensional scene objects, enabling evaluation of field of view, relative illumination, distortion, MTF, and color accuracy. For each metric, custom Python-based analysis routines were implemented. While these methods are not necessarily optimal, ImSym enables teams to evaluate different test targets and measurement approaches virtually before committing to hardware for manufacturing.

Field of View

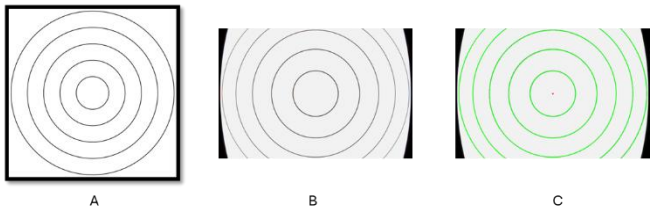


Figure 5 A) The original scene object B) The simulated Final Image in ImSym C) The result of searching for the circles in the image

Using the simulated image shown in Figure 5C, image height was measured across the sensor. With known sensor dimensions and object size, system magnification was calculated.

$$m = \frac{h_{image}}{h_{image\ object}} \quad (1)$$

$$\frac{h_{sensor\ diagonal}}{m} = h_{object\ diagonal} \quad (2)$$

$$fov = atan\left(\frac{h_{object\ diagonal}}{z_{object\ distance}}\right) \quad (3)$$

Relative Illumination

Relative illumination was computed from the same image used for the FOV calculation. It was defined as the ratio of the average signal in the image corners to the average signal in a central region.

$$Relative\ Illumination = \frac{\bar{g}_{Image\ Corners}}{\bar{g}_{center}} \quad (4)$$

Distortion

Distortion was measured using a dot-pattern target. A computer-vision algorithm quantified the difference between measured dot locations and their paraxial image heights [6][7]. Figure 6 shows the distortion target, simulated image, detected dot locations, and a magnified region.

$$D = \frac{\Delta H}{H} \cdot 100 \quad (5)$$

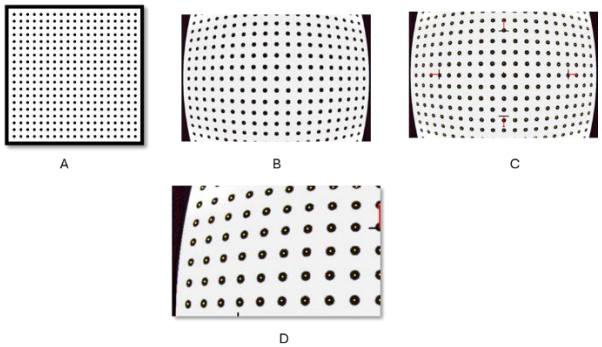
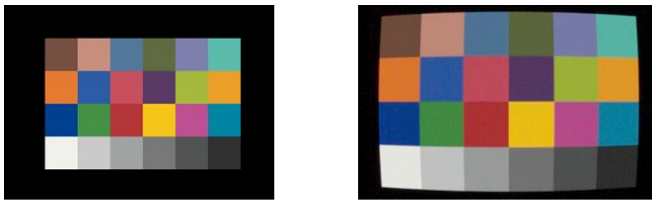


Figure 6: A) Scene Object B) Simulated image from ImSym C) Detection of dot pattern in the final image D) Close up of the dot detection

Color Accuracy

Color accuracy was evaluated using a simulated color-checker target. Manufacturer-provided color values were used as ground truth. RGB values from the simulated image were used to compute color accuracy, ΔE for each patch, using equation 6 , and the average error was reported [8]. Figure 7 shows the target and an example simulated image.



A B

Figure 7 Figure 7 A) Scene Object used for color accuracy measurement B) Example image simulation to measure color accuracy

$$\Delta E_{00} = \sqrt{\left(\frac{\Delta L'}{k_L S_L}\right)^2 + \left(\frac{\Delta C'}{k_C S_C}\right)^2 + \left(\frac{\Delta H'}{k_H S_H}\right)^2 + R_T \left(\frac{\Delta C'}{k_C S_C}\right)\left(\frac{\Delta H'}{k_H S_H}\right)} \quad (6)$$

Modulation Transfer Function

MTF was measured using a scene object containing four edges near the center of the FOV. The calculation followed the method described by Burns [9]. Figure 8 shows the MTF target, example simulation, and the computed line spread function. Figure 9 illustrates the impact of tolerance perturbations on MTF performance across simulated prototypes.

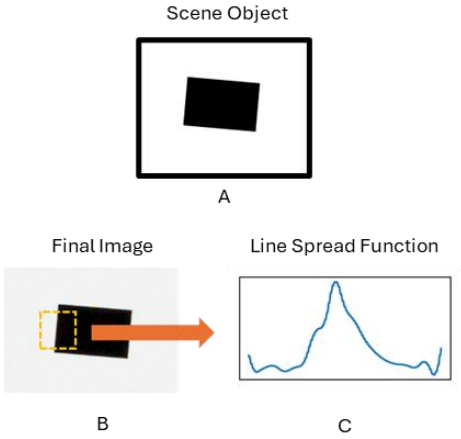


Figure 8 A) Scene Object used for MTF Measurement B) Example ROI selected for MTF Measurement C) Calculated Line Spread Function of the ROI in B)

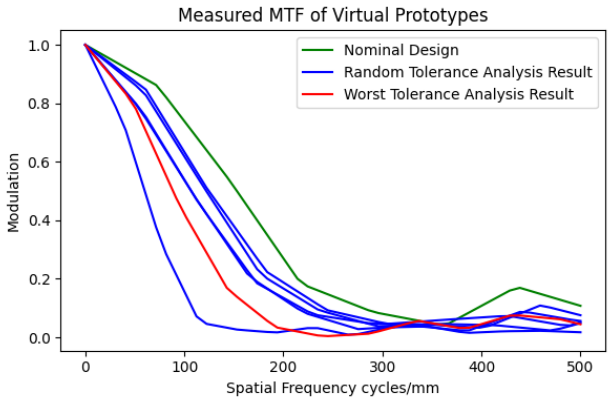


Figure 9 MTF Measurement for 5 random virtual prototypes as well as the nominal and worst-case designs

Validation Method

Validation testing was designed to represent a worst-case operating scenario. A machine-learning-based object-detection task was used to detect a vehicle at the edge of the camera FOV while a stray-light source was rotated azimuthally across the FOV.

Two detection methods were evaluated: a Haar cascade classifier and a Common Objects in Context (COCO)-based detector [10–13]. The Haar cascade uses wavelet features trained on positive and negative samples and is considered a legacy approach [10]. The COCO method relies on deep-learning models trained on 80 standardized object classes, including vehicles [11–13].

Results

Table 4 summarizes measurement results for the nominal design, worst-case configuration, and five random samples from the Monte Carlo simulation. Variation across prototypes was observed due to sensor noise and lens tolerances. These results were compared against the requirements defined in Table 1.

Based on these results, the initial design failed to meet all requirements. However, validation testing on the final simulated images revealed that acceptable task performance could still be achieved by adjusting system assumptions. Specifically, object detection was evaluated in the presence of stray light.

Using the Haar cascade, detection failures occurred when stray light obscured the vehicle. Two viable mitigation strategies were identified. First, a lower-noise sensor was simulated by reducing the dark noise by a factor of 100. While this approach increases cost of goods, it satisfies the original user requirements.

Alternatively, the COCO-based detector was evaluated using the original sensor noise settings. This method successfully detected vehicles across all simulated conditions. Figure 10 shows representative success and failure cases for the three configurations.

The percentage of missed frames was measured as the stray-light source was rotated through 180°, as shown in Figure 10. Table 5 summarizes detection performance. The Haar cascade failed in all configurations, while the COCO-based method detected vehicles in all simulated frames.

Virtual Prototype	MTF @ 100 cy/mm	ΔE_{00}
Nominal	0.9	2.41
Device 1	0.77	2.38
Device 2	0.76	2.59
Device 3	0.57	2.32
Device 4	0.56	2.35
Device 5	0.17	2.39
Worst Case	0.39	2.38

Virtual Prototype	Average TV Distortion	Diagonal Field of View	Relative Illumination
Nominal	0.1019	55.83°	89.90%
Device 1	0.127	54.9°	88.30%
Device 2	0.0604	52.39°	90.80%
Device 3	0.2202	54.44°	88.60%
Device 4	0.1004	55.15°	88.90%
Device 5	0.1154	54.01°	83.80%
Worst Case	0.1502	55.15°	89.10%

Table 4 Measurements of Nominal, five random, and the worst-case lenses through the camera simulation for MTF@100cv/mm, ΔE, average TV distortion, diagonal field of view, and relative illumination. We purposefully chose tolerances that would result in variation in MTF between units

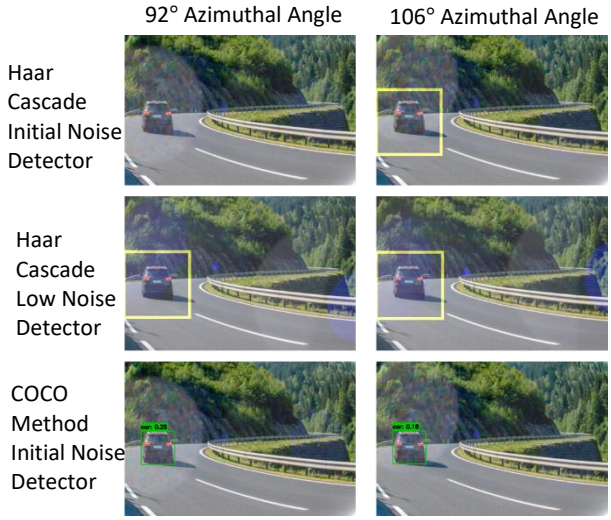


Figure 10 An example of success and failure of the Haar and COCO methods used for car detection. In the initial Haar cascade, the "Initial" noise setting was used for the sensor noise. The noise was reduced to by a factor of mode and the Haar cascade was more successful. However, we were able to keep the initial noise setting and select the COCO detection method for 100% success rate of detecting the car in all tested stray light scenarios.

Virtual Prototype	**% of frames that fail the detection task**	
	HAAR	COCO
Nominal	7.70%	0%
Device 1	13.90%	0%
Device 2	19.40%	0%
Device 3	25.00%	0%
Device 4	10.60%	0%
Device 5	13.90%	0%
Worst Case	12.80%	0%

Table 5 Summary of Haar and COCO methods ability to detect a car for all frames at the initial noise setting

User Requirement	Validated By:	Pass/Fail?
Detect cars in variety of lighting conditions	Less than 5% of cars frames missed	Pass – with an updated algorithm
Sub-System Requirement	Requirement	Pass/Fail?
Field of View	A diagonal field of view greater than 95° and less than 105°	Pass
Image Resolution	Greater than 50% Modulation at 100 lp/mm	Fail
TV Distortion	Average TV distortion < 25%	Pass
Relative Illumination	Relative illumination > 85%	Fail
Color Accuray	An average ΔE less than 5.0	Pass

Table 6 Summary of requirements' Pass/Fail status

Discussion

The combined verification and validation results enabled informed decision-making without physical prototypes. Table 6 summarizes the requirements that passed and failed.

Although the design failed the image-resolution and relative-illumination requirements, the design did pass the validation test. The availability of a more robust detection algorithm enabled two viable paths forward. There are multiple paths the team should consider before moving forward.

The MTF and distortion varied widely across the prototypes. This could result in failing gage repeatability and reproducibility studies when manufacturing at production levels. The variation in MTF could also suggest there is room for improvement in the lens design or the tolerance analysis. Exploring this gap could result in a more robust lens design.

A more robust detection algorithm was also found; however, corner cases may still exist. The team could select to use the new detection algorithm and select a higher SNR sensor.

The team may proceed with the existing hardware design while relaxing selected requirements and adopting a different detection algorithm. Without full definition of the team's objectives, we cannot make a recommendation of the best path forward. However,

we can conclude that ImSym enables these trade studies to be conducted using quantitative data prior to hardware fabrication, reducing risk and development cost. ImSym allows teams to look at their product development challenges across inputs from each subsystem of the camera system.

Conclusion

In this study, ImSym—Imaging System Simulator—was used to optimize an object-detection imaging system through end-to-end simulation. Design choices were evaluated rapidly, and their impact on system performance was assessed without building hardware. This approach is broadly applicable to optical system design and enables efficient optimization across multiple performance metrics while reducing development risk.

Author Biographies

Jonathan Brand is a Senior Staff Applications Engineer based in Denver, Colorado. He studied optics at the University of Rochester and earned his Ph.D. in Optical Sciences from the University of Arizona, where his research focused on liver fibrosis classification using MRI. Prior to joining ODE, he developed optical systems for surgical robotics and optical manufacturing applications.

Eric Herman is a Senior Systems Engineer in the Optical Solutions Group at Keysight Technologies, where he works on the Engineering Services team. His work focuses on optical design, analysis, and tolerancing. Eric earned his M.S. degree in Optical Sciences and Engineering from the Wyant College of Optical Sciences at the University of Arizona.

Maria Ruiz Gonzalez is an Applications Engineer in the Optical Solutions Group at Synopsys, supporting CODE V and ImSym. dShe earned her M.S. and Ph.D. degrees in Optical Sciences from the Wyant College of Optical Sciences at the University of Arizona, where her research focused on gamma-ray camera systems.

Leonard Zugby passed away prior to the completion of this work. He was a dedicated software engineer whose work focused on optical simulation. He made lasting contributions to tools such as CODE V, LightTools, and ImSym, enabling more accurate and efficient modeling of advanced optical systems.

Acknowledgements:

ImSym Validation Data was collected at the Imatest headquarters. Thank you to Jackson Knappen, Henry Koren, Ian Longton and the entire Imatest team for their assistance with collecting this data.

References

[1] Clark, “System of Systems Engineering and Family of Systems Engineering from a standards, V-Model, and Dual-V Model perspective” 2009 3rd Annual IEEE Systems Conference, 2009

[2] Wu, “An Exploratory Study of V-Model in Building ML-Enabled Software: A Systems Engineering Perspective”, CAIN: Conference on AI Engineering, 2024

[3] Keysight, “CODE V Optical Design Software” https://www.keysight.com/us/en/products/software/optical-solutions-software/optical-design-solutions/codev.html

[4] Keysight, “LightTools Illumination Design Software”, https://www.keysight.com/us/en/products/software/optical-solutions-software/optical-design-solutions/lighttools.html

[5] Keysight, “ImSym – Image System Simulator”, https://www.keysight.com/us/en/products/software/optical-solutions-software/optical-design-solutions/imsym-imaging-system-simulator.html

[6] Zhang, “A Flexible New Technique for Camera Calibration”, Microsoft Research, One Microsoft Way, Redmond, USA. December 1998

[7] EBU Tech 3249, “Measurement and analysis of the performance of film and television camera lenses,” 1995, https://tech.ebu.ch/docs/tech/tech3249.pdf

[8] Sharma, “The CIEDE2000 Color-Difference Formula: Implementation Notes, Supplementary Test Data, and Mathematical Observations” University of Rochester, Xeroz, 2004.

[9] Burns, “Slanted-Edge MTF for Digital Camera and Scanner Analysis.” Eastman Kodak Company, 2000

[10] X. Wen, H. Yuan, C. Yang, C. Song, B. Duan and H. Zhao, "Improved Haar Wavelet Feature Extraction Approaches for Vehicle Detection," 2007 IEEE Intelligent Transportation Systems Conference, Bellevue, WA, USA, 2007, pp. 1050-1053, doi: 10.1109/ITSC.2007.4357743.

[11] Han, “Vehicle Detection Method using Haar-like Feature on Real Time System”. World Academy of Science, Engineering and Technology 59 2009

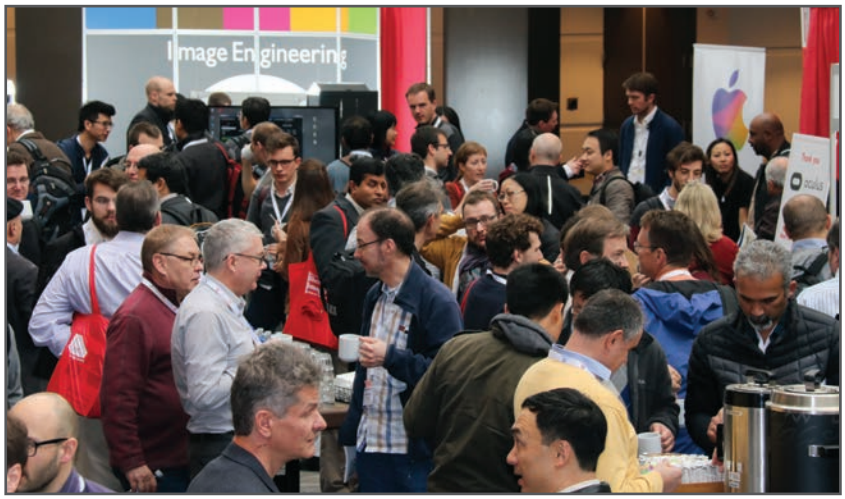
[12] Lin, *et al, “*Microsoft COCO: Common Objects in Context”. Cornell University. 2015

[13] Yıldırım,” Deep Learning-Based Object Detection for Vehicular Safety: A Comparative Study on COCO, KITTI, and Merged Datasets” International Conference on Electrical and Electronics Engineering, 2025

JOIN US AT THE NEXT EI!

electronic IMAGING

Imaging across applications . . . Where industry and academia meet!



• SHORT COURSES • EXHIBITS • DEMONSTRATION SESSION • PLENARY TALKS •
• INTERACTIVE PAPER SESSION • SPECIAL EVENTS • TECHNICAL SESSIONS •

www.electronicimaging.org

ADVANCING IMAGING SCIENCE 75 1947 IS&T 2022 imaging.org