

Fullseye: An Image-Inspection and 3-D Metrology Platform Based on Typed Operators and Physical Simulation

○Kazufumi Furuse[†]

[†]: Independent researcher

kazufumi@furuse.work

Abstract: We propose Fullseye, an open-source platform in which image-inspection and 3-D metrology procedures are assembled as chains of operators with declared input and output types, evaluated against ground truth generated by physical and imaging simulation, and recorded for reuse. Simulated images under varied illumination and optical conditions make it possible to determine whether processing succeeds and under which conditions it fails before constructing the apparatus. The platform comprises approximately 3,000 operators, operator search, and 218 proof-of-concept programs. We report quantitative evaluations of representative examples and distinguish their validation stages.

<Keywords> image inspection, 3-D metrology, physical simulation

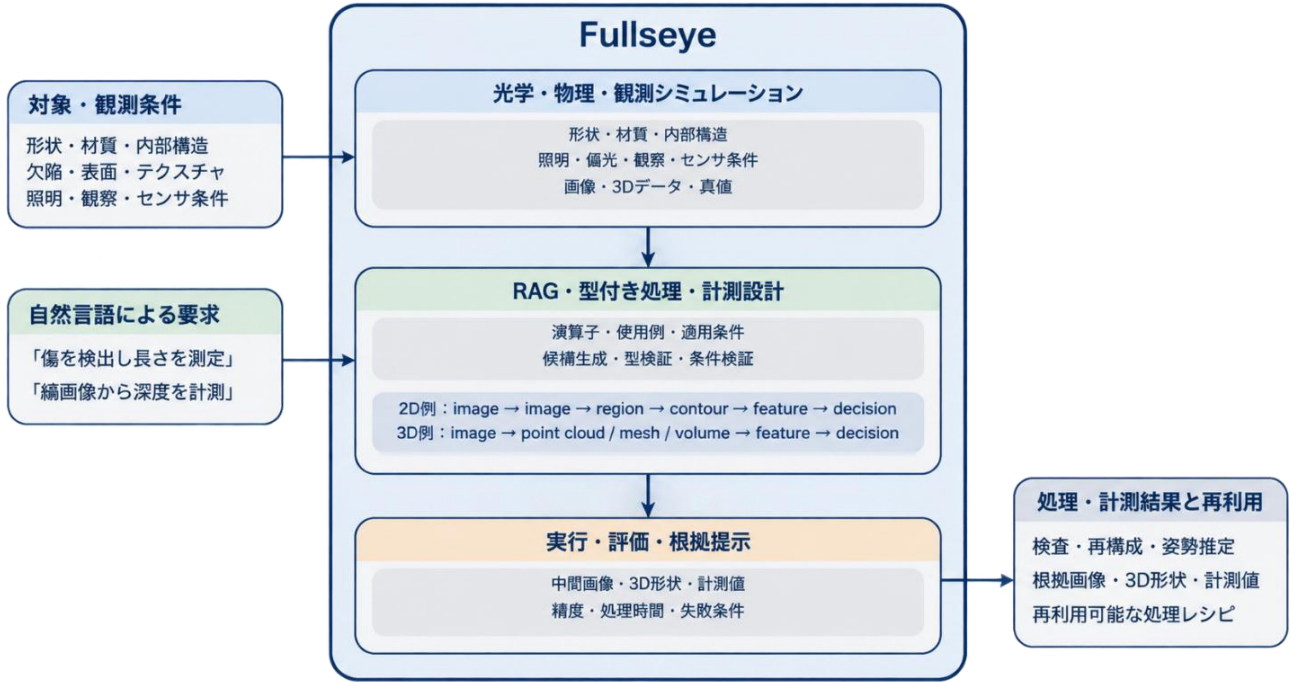


Fig. 1 Architecture of the Fullseye image-inspection and metrology pipeline

1. Introduction

In industrial vision, imaging conditions, processing procedures, and evaluation methods are often designed separately for each task, including visual inspection, 3-D metrology, optical and interferometric measurement, X-ray CT, and robotic perception. Consequently, transferring processing and evaluation knowledge developed at one site to another task with slightly different illumination or objects is not straightforward.

Moreover, ground truth is required to assess processing quality, but the exact depth, position, and dimensions of defects are often unknown in real images. Fullseye addresses this problem by virtually constructing the optical and imaging system, generating ground truth, and using it as the basis for comparison.

Fullseye[1] (Fig. 1), proposed in this paper, is a platform that connects the following procedure within one framework to address these two difficulties: limited transferability of knowledge and limited availability of

ground truth. Consider the task of detecting and measuring the length of a scratch on a glossy component surface. First, the user specifies the component geometry and material, scratch dimensions, and illumination and camera conditions to generate simulated images in which the true scratch position and length are known. Second, the user constructs a processing chain such as smoothing $\rightarrow$ binarization $\rightarrow$ region selection $\rightarrow$ contour extraction $\rightarrow$ length measurement (Listing 1 in Section 4). Incompatible combinations are rejected before execution. Third, the detection rate and length error are evaluated numerically against the simulated ground truth, and the illumination and scratch conditions that cause failure are recorded. The same procedure applies to point clouds, meshes, and volumes (3-D data), interference fringes and polarization images (optical metrology), and X-ray projections (CT).

The intended use cases are as follows. (a) Before constructing an apparatus, compare detection success using simulated images under varied illumination and camera conditions and narrow the candidates (Section 4.2). (b) During start-up, when few defective samples are available, evaluate processing and decision thresholds using defects of known dimensions and positions (Section 5). (c) Record processing chains, evaluation, and failure conditions as text; reevaluate changes such as another production line or degraded illumination with the same metrics; and present the basis for decisions (Listing 1). (d) Let generative AI assemble processing candidates and accept or reject them using types and ground truth (Section 5).

2. Related Work

BlenderProc2[2] and Kubric[3] are platforms for generating synthetic data with ground truth, and rendered synthetic defects have also been used to train and evaluate defect segmentation for visual inspection[4]. These studies primarily target training data, whereas this paper evaluates processing chains against ground truth for measured quantities. HALCON[5] distinguishes images, regions, and contours as types, while Halide[6] provides a declarative description of processing pipelines. Whereas these systems focus on executing processing, this work places emphasis on handling measurement-side processing such as X-ray CT, interferometry, and

polarization with the same type system and on making it callable from an LLM as open-source software. VisProg[7], ViperGPT[8], Gorilla[9], which generates calls based on API-documentation retrieval, and processing-chain generation for industrial inspection[10] use LLMs to compose visual processing. This work instead evaluates candidates using type consistency and ground truth. MVTec AD[11] and MVTec 3D-AD[12] provide real-image evaluation data. Although each component has precedents, the contribution of this work is to connect evaluation against ground truth for measured quantities, pre-execution checking and operator retrieval based on types, and regression tests of PoCs that record failure conditions in a single platform.

3. Definitions

The terms used in this paper are defined as follows.

[Operator] A unit of processing applied to an image or data, such as Gaussian smoothing, Otsu binarization, or point-cloud registration. It is implemented as a Python function and has a name, arguments, and input and output types.

[Type (data type)] The kind of data received and returned by an operator. In 2-D, the platform distinguishes image, color image, region (a set of pixels), contour (a sequence of subpixel points), and feature (a numerical value such as a dimension or shape descriptor). In 3-D, it distinguishes point clouds, meshes, and volumes. Pass/fail decisions are expressed as thresholds on features.

[Type consistency] At every stage in a processing chain, the output type of the preceding stage matches the input type of the next. For example, contour extraction (region $\rightarrow$ contour) may follow binarization (image $\rightarrow$ region), whereas smoothing that accepts an image (image $\rightarrow$ image) may not (Fig. 2). Fullseye checks a 2-D processing chain before executing its first stage and rejects mismatches; 3-D, optical, and other operators check input types on every call.

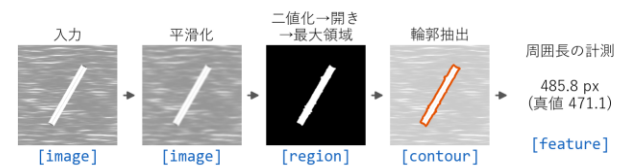


Fig. 2 Outputs and types at each processing stage (scratch on the metal surface in Fig. 3)

[Reusable processing] A processing chain saved as text as a sequence of operator names and arguments. As a rule, it returns the same result when given the same input and arguments, including random seeds, in the same version (determinism). Because it is text, it can be applied to and compared on other data and tasks and can also be searched.

[Ground truth] The correct answer used for evaluation. Here it means values known when a simulated image is generated, such as scratch position, depth, or step height, as well as analytical solutions and published values.

[PoC (Proof of Concept, demonstration program)] A program that executes, for one task, input generation, processing, comparison with ground truth or a reference, pass/fail criteria, and recording of failure conditions. Fullseye contains 218 PoCs, including subjects outside image inspection and metrology, all of which are run by regression tests.

4. Fullseye Architecture

Figure 1 shows the processing architecture of Fullseye. Given an object and observation conditions, optical, physical, and observation simulations generate images, 3-D data, and ground truth. A processing chain of typed operators then processes and measures them, leaving evaluation results and evidence—intermediate images, measured values, and failure conditions—in a reusable form.

4.1. Typed Operators and Scale

Fullseye is a Python platform that integrates 2-D image processing, 3-D geometry, point-cloud, mesh, and volume processing, optics, interference and polarization, structured light, light transport, X-ray CT, acoustics, and metrology as typed operators. It is released under the Apache-2.0 license; the source and installation commands are given at the beginning of Listing 1 (Python 3.10 or later; the basic functions run on CPU alone). The total number of operators is 3,083 in version 0.6.0 (October 2026).

As an indication of functional scope, HALCON[5], a representative industrial machine-vision operator system, is used as a comparison reference. The HALCON operator reference classifies 2,313 operators into 30 chapters, including Filters, Regions, and

Matching. For each Fullseye operator, the name of the functionally closest HALCON operator is recorded in machine-readable form. By this name mapping, 981 of the 2,313 operators (42.4%) have a corresponding Fullseye operator. This mapping indicates functional proximity and does not guarantee equivalent arguments or outputs. Of the 30 chapters, 23 contain at least one mapped operator. Most unmapped operators concern Graphics, tuple operations, system functions, file I/O, character recognition, and deep learning rather than image-processing algorithms themselves. However, 58 operators in template Matching and 44 in 3-D matching remain unmapped and are future work. Fullseye does not claim compatibility with or replacement of HALCON.

A 2-D processing chain is constructed, for example, as the type sequence image $\rightarrow$ image $\rightarrow$ region $\rightarrow$ contour $\rightarrow$ feature (Fig. 2). Illumination correction, smoothing, and edge enhancement are image $\rightarrow$ image; thresholding and segmentation are image $\rightarrow$ region; contour extraction is region $\rightarrow$ contour; and shape and dimensional measurements are contour $\rightarrow$ feature.

Listing 1 Execution of a processing chain and pre-execution rejection of a type-inconsistent chain (including output)

```
# https://github.com/furuse-kazufumi/fullseye
# pip install fullseye
import fullseye as fs
chain = ["gaussian", "otsu", "invert_region",
        "select_largest", "gen_contour_region_xld",
        "length_xld"]
L = fs.run_pipeline(img, chain) # perimeter 609.3 px
fs.run_pipeline(img,
               ["gaussian", "otsu", "gaussian"])
# ValueError: run_pipeline: type mismatch at
# stage 3 (gaussian): it takes 'image' but
# stage 2 (otsu) outputs 'region' -- the chain
# was refused before any stage ran
```

Listing 1 applies a 6-stage processing chain from smoothing through length measurement to a 640×480-pixel simulated image containing a dark scratch 4 px wide and 300 px long. The scratch perimeter is 609.3 px (ground truth 608 px; 300.7 px when converted to length), and processing took approximately 15 ms per image (CPU only on a typical desktop PC; median of 20 runs). A chain that places image-accepting smoothing after binarization is rejected before its first stage runs, with the mismatched stage identified.

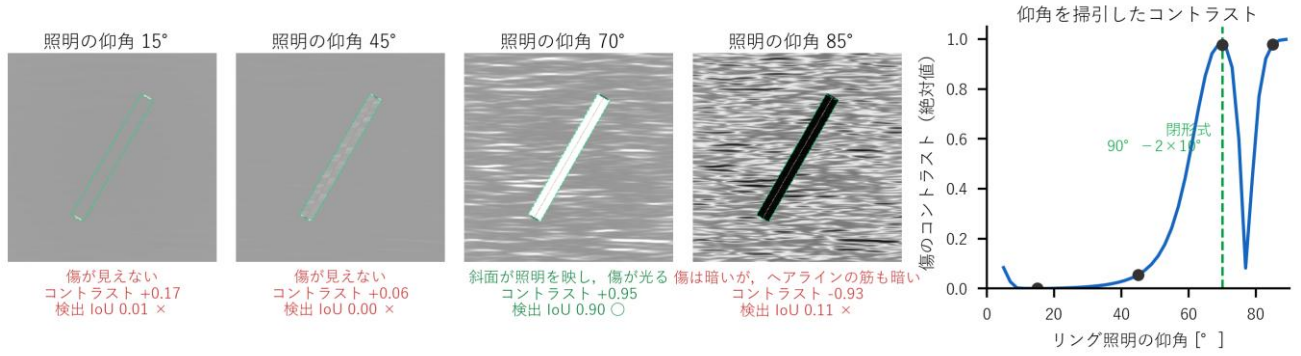


Fig. 3 Appearance of a scratch on a metal surface at different ring-light elevation angles

4.2. Physical and Imaging Simulation and Ground Truth

Object geometry, material, surface reflectance, defect geometry, illumination and observation conditions, and sensor conditions are supplied as input. Optical and imaging simulation simultaneously generate an observed image and ground truth, including depth, normals, defect regions, reflection components, object pose, and internal structure. Processing can therefore be evaluated quantitatively against correct answers that are difficult to obtain from real images alone. Systematically varying conditions reveals not only performance but also the conditions under which processing fails.

The optical operators cover every stage from light source to pixel (approximately 200 in version 0.6.0). Candidate low-angle and high-angle ring, dome, coaxial, and transmitted illumination configurations are ranked by scratch contrast and background uniformity. For materials, reflection is calculated from the complex refractive index of metals, Fresnel equations, and a microfacet BRDF. For lenses, ray tracing from surface prescriptions yields distortion and chromatic aberration; prescriptions can be optimized by damped least squares; and the diffraction-limited MTF and defocus are supported. Sensor characteristics can be obtained using the EMVA 1288 procedure, and line-scan imaging can be simulated. The platform also handles wave-optical propagation, polarization through Jones and Mueller matrices, white-light interference, light fields including compound-eye models, and photon counting (dToF). These functions permit illumination, lens, and sensor candidates to be compared in simulated images before an apparatus is built.

Figure 3 shows a V-shaped scratch with a 10° slope

on a hairline-finished metal surface, rendered while varying only the ring-light elevation angle. Pixel brightness is computed from illumination, surface orientation, and reflectance, while the true scratch position is known from the surface geometry. The scratch shines brightly near the closed-form angle $90^\circ - 2 \times 10^\circ = 70^\circ$. At 85° the scratch appears dark, but the hairline texture also darkens, causing the smoothing and binarization chain to detect the texture and fail. Detection success and failure causes can be checked against ground truth for each illumination condition. The curve at right is the absolute contrast between sloped and flat surfaces calculated by the lighting operator (lighting_sweep); its definition differs from the values in the individual images.

4.3. Knowledge Corpus and Operator Retrieval (RAG)

Each operator is accompanied by documentation describing it, its input and output types, examples, applicable conditions, and known limitations. The documentation serves both as the operator help and as the knowledge corpus for search. Full-text retrieval using BM25[13] finds related operators from phrases such as “noise removal” and “phase shift depth.” This retrieval can be called from LLM-based coding assistance either directly or through an MCP[14] server—the protocol by which AI calls external tools—that provides operator search, access to descriptions, and processing-chain execution within a dialogue (RAG: retrieval-augmented generation). As in API-documentation-grounded generation[9], the LLM divides a request into search terms, retrieves operators, reads their descriptions and input/output types, and constructs processing candidates.

4.4. GUI and Verification Mechanisms

Figure 4 shows the Fullseye Studio interface. It applies the chain through region extraction in Fig. 2 to the image at the 70° elevation angle in Fig. 3; the 3-D view at right exaggerates height by a factor of 8. The

sample-chain selector, operator list by type (e.g., [image → image] and [region → region]), and code panel allow users to interactively construct and execute type-consistent chains and obtain Python code.

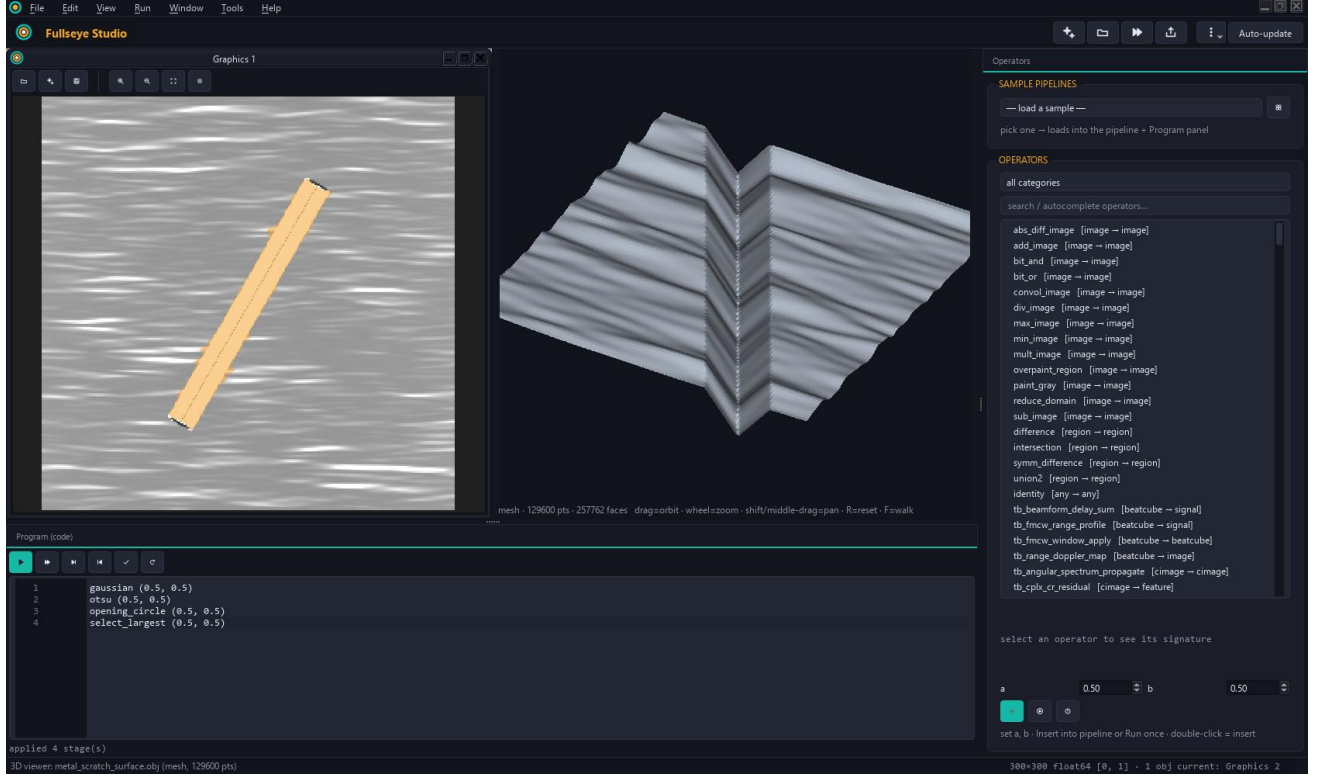


Fig. 4 Fullseye Studio interface

Approximately 1.8×10^4 regression tests across the library continuously verify agreement between declared and actual input/output types of 2-D operators and between numerical values in the documentation and execution results.

5. Demonstration with Representative PoCs

Fullseye implements PoCs (Section 3) for 2-D image inspection, 3-D reconstruction, point-cloud, mesh, and volume processing, optical, interferometric, and polarization metrology, X-ray CT, thermal and acoustic analysis, and robotic and spatial perception. They demonstrate that objects and observation conditions, typed operators, evaluation quantities, ground truth or comparison references, and result visualization can be handled within a common framework.

Many of these PoCs were created interactively. The author specifies a task and how to construct ground truth; LLM-based coding assistance consults operator

documentation and input/output types and combines classical image processing with optical and imaging simulation operators to write the processing chain and evaluation. The author compares the result with ground truth and decides whether to accept it, identifying the cause and requesting reconstruction if it fails. Type-inconsistent combinations are rejected before execution, and results are judged against ground truth, so type errors and results departing from ground truth are detected before adoption. This procedure enabled one person to accumulate 218 cross-domain PoCs.

Figure 5 shows output examples from representative PoCs. Because each panel has a different task, input, and evaluation metric, the figure indicates implementation scope and does not compare performance among PoCs. The shape in [H] is the actual Itokawa shape model[15]; fine relief and rocks are synthetic. Table 1 summarizes representative evaluation targets for each observation and measurement category.

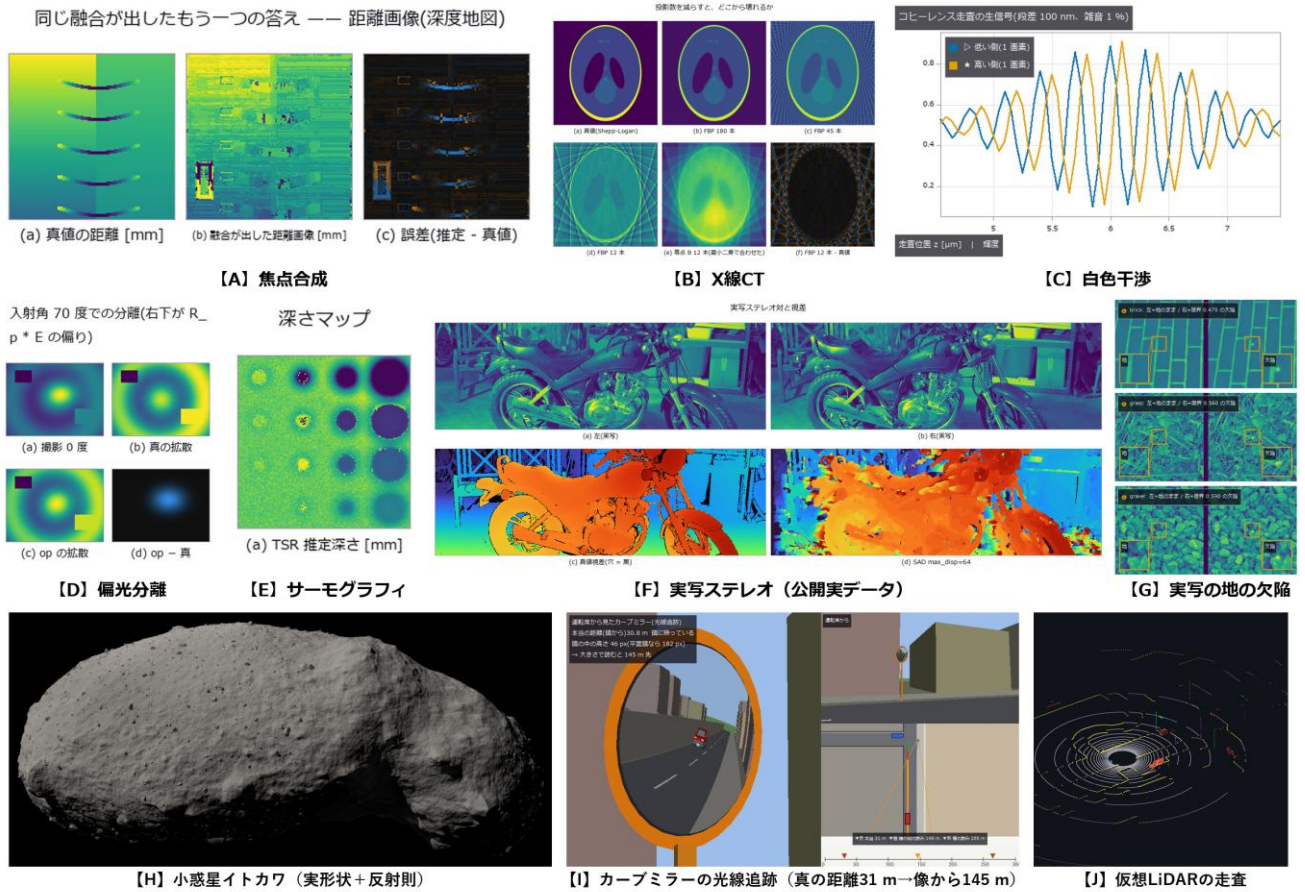


Fig. 5 Output examples from representative PoCs ([F] uses public real data, [G] is semi-synthetic, and the others are synthetic)

Table 1 PoC categories and representative evaluation targets

Category	Representative evaluation targets
Image inspection and appearance metrology	Defect candidates, dimensions, shape, texture
3-D metrology and geometric processing	Depth, point clouds, meshes, position and pose
X-ray CT and volume processing	Reconstruction, volume, voids, internal geometry
Optics, interference, and polarization	Phase, reflection, polarization properties, optical images
Thermal, acoustic, and time-series	Temperature fields, frequency characteristics, anomaly metrics
Robotic and spatial perception	Object pose, grasp candidates, safe distance

Each PoC records not only the final decision or measured value, but also input data, intermediate

images or geometry, the operator chain used, principal arguments, evaluation metrics, ground truth or comparison reference, and known limitations. This makes it possible to inspect the basis of results, rerun procedures, and reuse operators and evaluation methods for other tasks.

PoC maturity is not uniform. Verification with synthetic data carrying ground truth, validation with public real images or measured data, and validation with connected physical sensors or apparatus are distinguished. Synthetic data are used to vary geometry, material, illumination, viewing direction, noise, and defect conditions systematically and investigate performance and failure conditions. Real images and measured data are used to verify reflection, backgrounds, object variability, calibration error, missing data, and imaging noise that synthetic data alone may not capture. Unverified scope is explicitly marked as a research prototype and separated from verified functions. In the version 0.6.0 maturity ledger, generated mechanically from the repository, 32 of 35 capabilities were verified on synthetic data and 3 on

public real data; none had yet been validated on hardware.

Representative quantitative evaluations are shown in Table 2. Every value in Table 2 is measured against synthetic data, using an analytical solution or ground

truth embedded during generation. Each can be reproduced by running the respective PoC (examples/poc_*.py) at tag v0.6.0. The polarization row checks the implementation against the same Fresnel equations used in synthesis.

Table 2 Quantitative evaluation of representative PoCs (with ground truth or reference; all data are synthetic)

PoC (task)	Ground truth/reference	Metric	Result
Focus stacking (shape from focus)	Known depth and all-in-focus image	All-in-focus PSNR, depth error	All-in-focus 33.69 dB (central frame 28.52 dB; best single frame 29.44 dB); overall depth RMS error 0.92 mm (constant estimate 1.17 mm), 0.47 mm with texture, and 2.20 mm without texture (worse than the region-wise best constant of 0.36 mm)
X-ray CT fidelity (Shepp-Logan[16])	Analytical total mass	Mass conservation	Relative error in total projected mass 1.3×10^{-5} ; -0.0099% after reconstruction by filtered backprojection (FBP)[17]
Step-height measurement by white-light interferometry	True step height 50–500 nm	Bias and variation	With 1% noise (40 runs, average of 144 pixels), bias ≤ 2.4 nm and standard deviation 10–14 nm (approximately 20% relative at a 50 nm step)
Specular-reflection separation by polarization	Closed-form Fresnel solution	Separation error	Agreement with closed-form $R_p \cdot E$ (0 at the Brewster angle) to within rounding error ($\leq 4.7 \times 10^{-17}$), checking the implementation
Nondestructive testing by thermography	Known defect depth	Depth error	-6 to $+2\%$ for diameters ≥ 8 mm; shortening the fitting window from 25 s to 4 s reduced the worst error among 8 small defects from $+612\%$ to -14% (the window was selected on the same data)
Compound-eye light-field measurement	Synthetic ground truth (known disparity and depth)	SNR gain (versus $\sqrt{N}$), depth error	2.25 at $N=5$ ($\sqrt{5}=2.24$), 5.33 at $N=49$ ($\sqrt{49}=7.00$); depth RMSE 0.010 px/view (reliable pixels away from discontinuities)
Coefficient of restitution and friction from high-speed video (table-tennis ball)	Published measurements[18], synthetic ground truth	Speed dependence of restitution, coefficient of friction, bounce regime	Slope -0.00572 /(km/h) (1.4% from measured -0.0058), coefficient of friction 0.250 (truth 0.25), and 15/15 agreement in sliding/rolling classification

In the thermography PoC, the initial result that shallower defects were increasingly overestimated as their diameter decreased appeared to be a physical detection limit. A control experiment showed that the cause was not physics but selection of the time window used for fitting. Shortening the window from 25 s to 4 s reduced the error of a defect from $+612\%$ to -9% , and the worst value among the 8 small defects to -14% . However, the window was selected after examining the same 8 defects; generalization of depth-dependent window selection remains unverified. Recording failure conditions and separating their causes are treated as part of evaluation.

In the PoC that models a compound-eye optical system as a light-field sensor[19, 20], the SNR gain from superposing the same object point across N views was compared with the ideal $\sqrt{N}$ for independent noise. This corresponds to reducing noise on a dark object with multiple cameras. The gain at $N=49$ remained 5.33 because integer-pixel shifts caused errors at peripheral pixels extending beyond the image; removing the outer 8 pixels yielded 7.19 ($\sqrt{49}=7.00$). Saturation from interpolation error remains for noninteger disparities.

From ball bounces recorded by a high-speed camera (1000 fps), the coefficients of restitution and friction were read from images alone and compared with

published measurements (Table 2). This can also support evaluation of bouncing and sliding during component feeding and transport.

Fullseye operators are used not only in standalone PoCs but also by other research projects as the “eyes” of robots in physical simulation (Table 3). Each project’s physical model handles control (legs or propulsion), while Fullseye handles only perception of terrain and targets from point clouds and range images. Under conditions where vision is unnecessary—no step or target displacement—the results agree, confirming that Fullseye does not interfere with control. For a

quadruped robot, however, no better gait existed for vision to select, so adding vision produced almost no improvement. Such negative results are also recorded as boundaries of applicability. The platform also provides an entry point (`g1_walk_policy`) for executing a GPU-trained humanoid policy in physical simulation (Fig. 6). A separately trained running policy executed through this entry point traveled 50 m in 12 s in MuJoCo without falling. The improvements in Table 3 arise from project-side policies that use step height or target position estimated by Fullseye. Those projects’ code is not included in the release accompanying this paper.

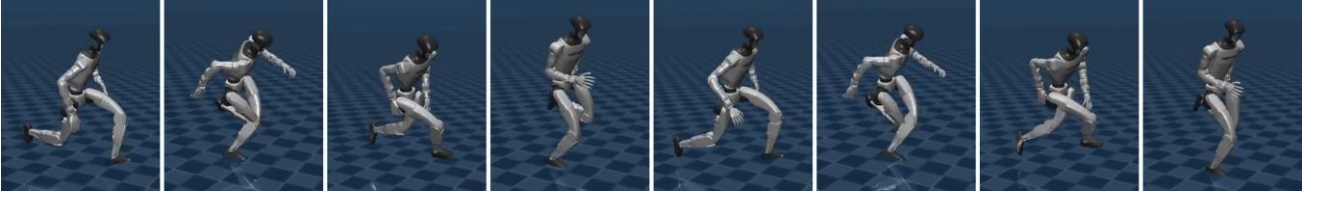


Fig. 6 Running of the G1 humanoid robot

Table 3 Examples using Fullseye for robot perception (physical simulation, compared with no vision)

Task	Role of Fullseye (operators)	Without vision	With Fullseye vision
Biped walking (planar biped model; 7 step-height conditions from 2–14 cm; distance traveled in at most 5 s)	Terrain point cloud → elevation map → step-height estimate (elevation_map, step_edges). Point cloud generated from step geometry without noise. 1 deterministic run per condition	Mean 3.35 m with steps	Mean 4.10 m (+0.75 m; no condition worsened)
Rocket landing (landing pad displaced 0.5–2.5 m)	Downward range point cloud (1 cm noise) → height threshold → centroid to estimate pad position (error ≤ 4 mm). 5 displacement conditions $\times$ 30 runs	Landing-on-pad rate 21%; landing error 1.58 m	Landing rate 67%; landing error 0.74 m

6. Discussion

The central point of Fullseye is not merely to collect multidisciplinary processing in one repository, but to associate objects and observation conditions, data representations, operators, evaluation metrics, and visualization and manage them for reuse. Type consistency supports both the safety of assembling and evaluating processing chains and the index used to find and reuse operators for a purpose.

Ground-truth synthetic data from physical simulation enable evaluation of rare defects and extreme observation conditions, but cannot fully reproduce illumination nonuniformity, aberrations, sensor noise, and specimen variability in physical systems. Performance in a real environment cannot be claimed

from synthetic evaluation alone. Validation on public real data has begun. For real photographed stereo with ground-truth disparity (Fig. 5[F], one pair from Middlebury 2014[21], downsampled to 1/4), the default setting produced a proportion of pixels with errors above 2 pixels (95.06%) worse than merely outputting a constant disparity; 15.81% remained even with semi-global matching (SGM)[22]. In semi-synthetic evaluation that inserted known defects into 3 real photographic textures (Fig. 5[G], included with scikit-image[23], CC0), detection limits were 2.03–3.47 times higher than on synthetic backgrounds with matched residual-noise levels. The limit was governed by background pattern rather than noise magnitude, a difference invisible with synthetic backgrounds alone. Validation with public measured data and physical

systems must be added in stages to clarify the platform's usefulness and range of applicability.

LLM-based composition of processing candidates improves efficiency, but acceptance should not be decided solely from language-model output. Checking types, argument ranges, determinism, performance on evaluation data, and known failure conditions before adoption can combine the efficiency of AI assistance with the reproducibility, traceability, and explainability required in industrial metrology. Usability was inspected by asking a general-purpose AI agent not involved in development (GenSpark) to use the public release; reported problems were reproduced and corrected. Operator-retrieval accuracy and the accuracy of LLM-composed processing candidates were not evaluated quantitatively in this paper.

7. Conclusion

This paper presented Fullseye, a platform for image inspection and 3-D metrology that integrates typed operators, ground-truth evaluation using physical and imaging simulation, and operator retrieval (RAG). Representative PoCs were evaluated quantitatively against ground truth or comparison references. Validation stages using synthetic and public real data were distinguished, and hardware validation was explicitly identified as future work.

Future work will broaden validation on public real images, measured data, and physical sensors and pursue applications to semiconductor packaging, precision optical components, nondestructive testing, and Physical AI. Public benchmarks such as MVTec AD[11] will also be evaluated. Figures and videos from representative PoCs are available on the page in Fig. 7 (<https://furuse.work/view2026/>). We will also expand template and 3-D matching, for which many HALCON operators remain unmapped, and investigate an extension that places compound-eye light-field imaging upstream and processing based on the insect visual-system connectome[24] downstream. This may also lead to processing for event cameras[25], which output only brightness changes at low latency.



Fig. 7 Index page for figures and videos from representative PoCs

Acknowledgments Generative AI (Claude) was used for part of the PoC implementation, literature review, and manuscript editing. The author verified the reported numerical values and content against execution results and original sources.

References

- [1] 古瀬和文 [Kazufumi Furuse]: Fullseye, <https://github.com/furuse-kazufumi/fullseye>, PyPI fullseye 0.6.0 (tag v0.6.0), 2026.
- [2] M. Denninger, et al.: BlenderProc2: A Procedural Pipeline for Photorealistic Rendering, J. Open Source Software, Vol.8, No.82, 4901, 2023.
- [3] K. Greff, et al.: Kubric: A Scalable Dataset Generator, Proc. CVPR 2022, pp.3749-3761, 2022.
- [4] J. Fulir, L. Bosnar, H. Hagen and P. Gospodnetić: Synthetic Data for Defect Segmentation on Complex Metal Surfaces, Proc. CVPR Workshops 2023, pp.4424-4434, 2023.
- [5] MVTec Software GmbH: HALCON Operator Reference, <https://www.mvtec.com/doc/halcon/>, accessed September 11, 2026.
- [6] J. Ragan-Kelley, et al.: Halide: A Language and Compiler for Optimizing Parallelism, Locality, and Recomputation in Image Processing Pipelines, Proc. PLDI 2013, pp.519-530, 2013.
- [7] T. Gupta and A. Kembhavi: Visual Programming: Compositional Visual Reasoning Without Training, Proc. CVPR 2023, pp.14953-14962, 2023.
- [8] D. Surís, S. Menon and C. Vondrick: ViperGPT: Visual Inference via Python Execution for Reasoning, Proc. ICCV 2023, pp.11888-11898, 2023.
- [9] S. G. Patil, T. Zhang, X. Wang and J. E. Gonzalez: Gorilla: Large Language Model Connected with Massive APIs, Proc. NeurIPS 2024, pp.126544-126565, 2024.
- [10] A. Martikkala, A. Daareyni, A. Queguineur and I. Flores Ituarte: LLM-assisted Workflow Generation for Low-cost Visual Inspection and Image Analysis, IET Conference Proceedings, Vol.2025, No.28, pp.54-61, 2025.
- [11] P. Bergmann, M. Fauser, D. Sattlegger and C. Steger: MVTec AD — A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection, Proc. CVPR 2019, pp.9584-9592, 2019.
- [12] P. Bergmann, X. Jin, D. Sattlegger and C. Steger: The

MVTec 3D-AD Dataset for Unsupervised 3D Anomaly Detection and Localization, Proc. VISAPP 2022, pp.202-213, 2022.

- [13] S. Robertson and H. Zaragoza: The Probabilistic Relevance Framework: BM25 and Beyond, Foundations and Trends in Information Retrieval, Vol.3, No.4, pp.333-389, 2009.
- [14] Model Context Protocol: Specification, <https://modelcontextprotocol.io/specification>, accessed October 12, 2026.
- [15] R. Gaskell, et al.: Gaskell Itokawa Shape Model V1.1, NASA Planetary Data System, doi:10.26033/3b2j-yy57, 2007.
- [16] L. A. Shepp and B. F. Logan: The Fourier Reconstruction of a Head Section, IEEE Trans. Nuclear Science, Vol.21, No.3, pp.21-43, 1974.
- [17] A. C. Kak and M. Slaney: Principles of Computerized Tomographic Imaging, IEEE Press, 1988.
- [18] Y. Inaba, et al.: Effect of Changing Table Tennis Ball Material from Celluloid to Plastic on the Post-Collision Ball Trajectory, J. Human Kinetics, Vol.55, pp.29-38, 2017.
- [19] M. F. Land and D.-E. Nilsson: Animal Eyes, 2nd ed., Oxford University Press, 2012.
- [20] R. Ng, et al.: Light Field Photography with a Hand-held Plenoptic Camera, Stanford Tech. Report CSTR 2005-02, 2005.
- [21] D. Scharstein, et al.: High-Resolution Stereo Datasets with Subpixel-Accurate Ground Truth, Proc. GCPR 2014, LNCS 8753, pp.31-42, 2014.
- [22] H. Hirschmüller: Stereo Processing by Semiglobal Matching and Mutual Information, IEEE Trans. Pattern Analysis and Machine Intelligence, Vol.30, No.2, pp.328-341, 2008.
- [23] S. van der Walt, et al.: scikit-image: image processing in Python, PeerJ, Vol.2, e453, 2014.
- [24] J. K. Lappalainen, et al.: Connectome-constrained networks predict neural activity across the fly visual system, Nature, Vol.634, pp.1132-1140, 2024.
- [25] G. Gallego, et al.: Event-Based Vision: A Survey, IEEE Trans. Pattern Analysis and Machine Intelligence, Vol.44, No.1, pp.154-180, 2022.

Kazufumi Furuse: Independent researcher developing Fullseye, an open-source image-inspection platform spanning image processing, 3-D metrology, and physical simulation. His research interests include validation of image processing with ground-truth evaluation, explainable automatic composition of processing, and metrology inspired by biological vision. <https://furuse.work>