STANFORD ROBOTICS

Leopard stereo depth software

Customer evaluation and source code licensing

November 2026

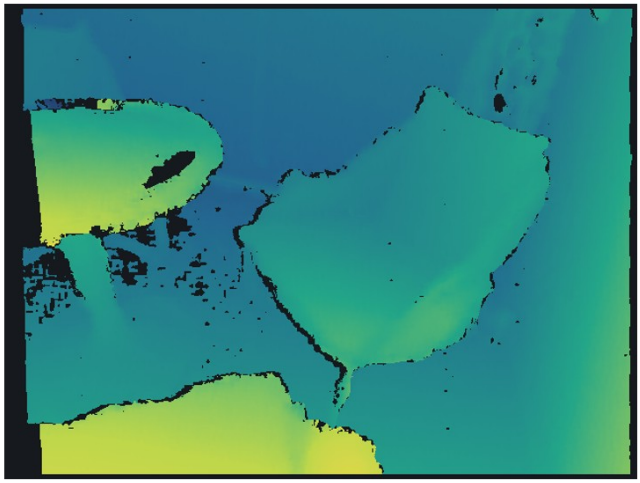
Leopard gives robotics and embedded-vision teams a configurable stereo matching engine they can inspect, integrate and tune. Our implementation combines slanted-window matching with CPU and GPU backends, brightness correction and explicit validity controls. It requires rectified stereo images and does not require a trained neural network.

Left stereo input



→

Leopard disparity output



Project sample frame0408. Local brightness correction; display scale 0-40 px (clipped), search cap 256 px. No ground truth is available for this illustration. [2]

Why evaluate Leopard

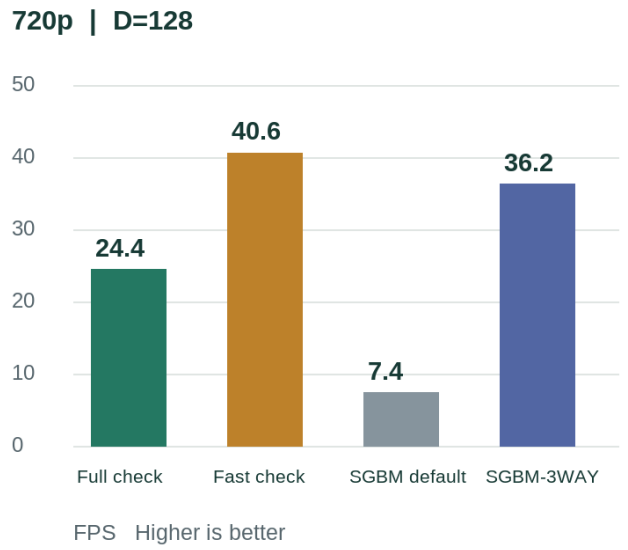
Choose the compute and quality balance. Full reverse matching provides stronger consistency checking. A faster projected check reduces latency when the application can tolerate different rejection behavior.

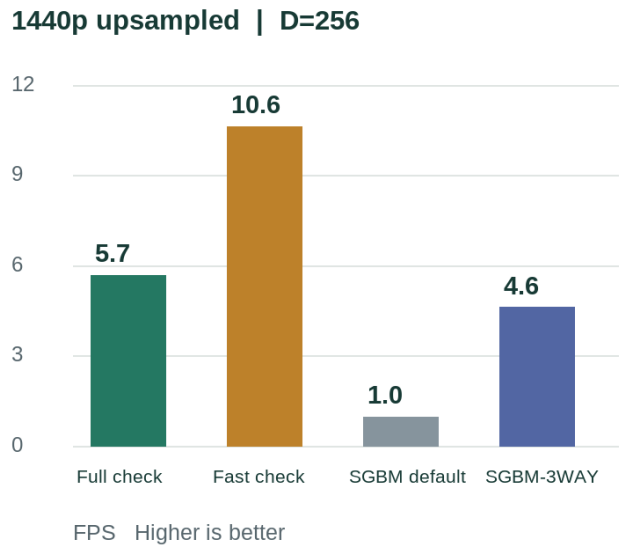
Keep control of your depth pipeline. Source access supports application-specific thresholds, camera integration and diagnostics. Invalid measurements remain explicit; optional completion identifies inferred pixels separately.

Build on measured implementations. Our latest CPU experiment reached 40.6 FPS at 720p in the fast FP32 configuration. A separate Apple Metal test reached 67.8 FPS at 1248 x 688 with brightness correction. These are different configurations and workloads, detailed on the following pages. [1,3]

CPU performance and scaling

Leopard is particularly worth evaluating when resolution and disparity search range make exhaustive matching expensive. With fixed tile, window and iteration settings, its work scales approximately with pixel count; it avoids a dense pixel-by-disparity cost volume. The crossover against SGBM depends on the input and hardware.





Same Apple M3 Pro; 12-thread budget unless noted; OpenCV 4.13.0. Default means five-direction MODE_SGBM, not factory-default parameters. Fast Leopard uses a weaker projected consistency check. All four configurations were rerun together. [1]

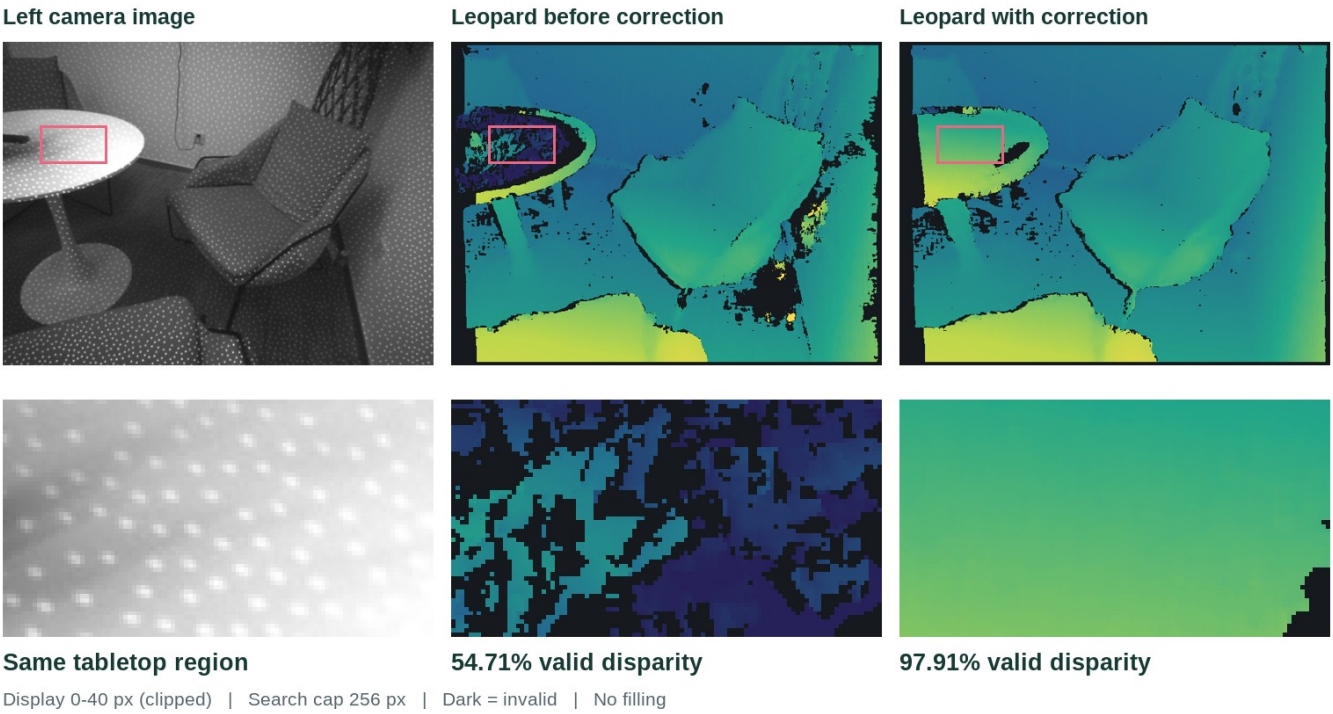
Office pair	D	Leopard full FPS	Leopard fast FPS	SGBM default FPS	SGBM 3WAY FPS
640 x 360	64	86.85	154.48	52.02	230.77
1280 x 720	128	24.41	40.56	7.37	36.25
2560 x 1440 *	256	5.67	10.62	0.97	4.61
1280 x 720 1 thread	128	4.02	7.62	7.38	10.16

At 720p, full-check Leopard achieved 3.31 times the throughput of default SGBM, and fast Leopard 5.51 times. Against the faster SGBM-3WAY, fast Leopard achieved 1.12 times at 720p and 2.30 times at upsampled 1440p. SGBM-3WAY remained faster at 360p and than full-check Leopard at 720p. The baseline mode materially changes the apparent advantage. The one-thread row shows the importance of parallel execution; an equal thread budget does not imply equal core utilization.

*1440p is upsampled from 720p for timing only. Resolution and disparity range D increase together. OpenCV modes share blockSize=5, P1/P2=200/800, uniqueness=10, preFilterCap=31, disp12MaxDiff=1 and no speckle filter. Three shuffled rounds of seven warm frames; preparation, matching, filtering and output included; setup, decoding, resizing, capture, rectification and display excluded. Desktop apps remained active. [1]

Recovering bright surface detail

Exposure and local brightness differences can make corresponding stereo patches look different even when their texture remains visible. Leopard offers optional local brightness normalization before matching. On this bright tabletop example, valid disparity in the fixed table region increased from 54.71% to 97.91%, with no hole filling. [2]



The highlighted region is identical in every panel. Coverage is the fraction receiving an accepted disparity, not a ground-truth accuracy score. This example has no ground truth. [2]

Keep preprocessing on the GPU

The Metal implementation performs the selected Gaussian correction on the Apple GPU before matching. Across the separate 27-capture evaluation at 1248 x 688, moving that same correction from CPU to GPU raised median processing throughput from 42.8 to 67.8 FPS, a 1.59 times speedup. [3]

Metal pipeline at 1248 x 688	Median FPS
CPU correction plus Metal matching	42.8
Metal correction plus Metal matching	67.8
Metal correction and matching plus completion	62.8

Completion is optional and conservative: it preserves accepted measurements and tracks inferred pixels separately. Brightness normalization and completion can improve useful coverage, but may admit additional outliers. Their parameters should be selected using the application's own images and error tolerance.

GPU figures include host preparation, transfers, solve and synchronization. They are repeated-frame processing rates, not live-camera delivery rates, and are not a CPU-versus-SGBM comparison. Fifteen alternating-order warm repetitions per capture; summary is the median of per-capture FPS. [3]

Accuracy and usable coverage

A useful depth engine must balance measurement error with how much of the scene it returns. We report both. The following CPU results pool 27 real captures from Skoltech3D: three objects, three viewpoints and three lighting conditions, using RealSense D435 infrared pairs and independent scanner ground truth. [1,5]

Method	MAE px	RMSE px	Valid coverage	Correct within 1 px of all GT
Leopard FP32 full	0.402	1.022	85.0%	80.0%
Leopard FP32 fast	0.455	1.356	87.3%	81.0%
OpenCV SGBM default	0.502	1.469	98.3%	90.3%
OpenCV SGBM 3WAY	0.435	1.330	97.8%	91.1%

MAE = mean absolute error. RMSE = root mean square error. GT = ground truth. All error values are disparity pixels, not millimeters.

Full-check Leopard had lower average and RMS error among accepted pixels than either OpenCV mode. Both SGBM modes returned more pixels and a higher overall fraction within one pixel of ground truth. Default SGBM had slightly greater coverage than 3WAY, but higher error and lower correct-pixel yield on this dataset. Accepted-pixel error alone is not an equal-coverage comparison or proof of overall accuracy superiority.

Choosing a configuration

Start with FP32 and full reverse checking when rejected pixels are preferable to uncertain measurements. Compare projected checking when throughput is the priority and the application has its own validation or downstream filtering. Neither choice removes the need to measure error and coverage together.

The 16-bit fixed-point path remains an experimental option. FP32 offered a more consistent speed benefit in our tests with smaller individual-pixel changes. Reference double-precision sampling remains available for reproducibility.

Validation that supports integration

The CPU work includes 24 synthetic controls with known disparities, covering integer and fractional shifts, slanted surfaces, noise, depth steps and sparse dots across three seeds. FP32 achieved 0.01597 px pooled accepted-pixel MAE and 99.619% coverage on those controls. Synthetic results verify numerical behavior; they do not establish real-scene depth accuracy. [1]

Eleven regression tests, scalar-versus-SIMD kernel checks, memory-safety checks and thread-determinism checks passed. The unchanged reference mode matches saved results exactly on the 24 synthetic controls and 27 real captures. Experimental precision modes can change individual winners and validity masks. [1]

Real ground truth is available only on selected object surfaces at approximately 0.63-0.88 m. These are repeated captures of three scenes, not 27 independent scenes or a test at 4 m. MAE and RMSE use each method’s accepted pixels. Coverage and the within-one-pixel yield use the fixed evaluation region. No hole filling or brightness normalization is used in the CPU comparison above. [1,5]

Source code for your depth pipeline

The value of a Leopard license is access to an adaptable implementation and its evaluation tools. Your team can inspect the matching stages, choose the target backend and connect disparity and validity outputs to the rest of the perception stack. Commercial scope can be selected around the backends and platforms you need.

Implementation	Available capability	Measured target
C++17 CPU with Python adapter	Persistent buffers and workers; reference, FP32 and fixed16 options	Apple M3 Pro CPU
Apple Metal	GPU matching, local brightness normalization and optional completion	Apple silicon macOS
CUDA with Colab notebook	Independent GPU port and reproducible examples	NVIDIA Tesla T4
Python and Numba reference	Readable reference, diagnostics and evaluation tooling	Desktop CPU

Backends have different wrappers and performance characteristics; feature and numerical parity should be verified for the selected release. CPU timings in this report are not CUDA or Metal timings. Other processors require target-specific evaluation. [1,3,6]

A practical integration boundary

Input: equally sized, rectified left and right grayscale images, with a disparity range suited to the camera. Calibration and rectification must be handled before matching; ordinary single-camera video is not sufficient.

Output: floating-point disparity in pixels, an explicit validity mask, and optional matching costs and plane diagnostics. Invalid disparity is distinct from a valid zero. Optional completion distinguishes measured, inferred and unresolved pixels.

Metric depth: convert valid positive disparity using the calibrated focal length and stereo baseline. The lens, baseline, resolution, texture and exposure determine the practical depth envelope; there is no camera-independent millimeter accuracy guarantee.

Why license the implementation

Retain the ability to tune difficult lighting and rejection behavior instead of treating depth output as an opaque result. Reuse prepared processing objects for repeated frames, inspect stage timings, and reproduce comparisons using saved configurations and raw timing samples.

Source-level integration also allows your team to adapt memory handling and platform interfaces to its own product. Existing tests and reference outputs provide a starting point for verifying those changes. Delivery scope, integration work and support terms should be agreed for the target application.

Evaluate Leopard on your camera

The next step is a measured evaluation on representative customer footage. We propose comparing Leopard configurations with your current stereo matcher on the same inputs and hardware, then agreeing a source-code licensing scope based on the result.

Define the application

Provide calibrated stereo pairs, image dimensions, required working distance, target processor and latency budget. Include difficult cases such as motion, low texture, glare, exposure mismatch and depth boundaries, together with ground truth where available.

Agree the acceptance criteria

Measure end-to-end latency and its variation, accepted-pixel error, invalid-pixel rate, correct-pixel yield and temporal stability. Select the CPU or GPU backend, compare full and projected checking, and test normalization and completion as separate options. Keep camera capture and downstream processing costs visible.

Choose the commercial scope

Discuss source-code licensing for the selected backends, target platforms and deployment requirements. Identify integration deliverables, documentation, support and acceptance tests before agreeing commercial terms. Pricing and license terms are available through direct discussion.

Contact Leo Pang at Stanford Robotics

leo.pang@gmail.com

Evidence and attribution

[1] Stanford Robotics, matched CPU comparison, 6 November 2026. Apple M3 Pro; OpenCV 4.13.0 default MODE_SGBM and SGBM-3WAY, identical non-mode parameters and thread budgets. Four configurations rerun with 21 warm samples each and 27 native GT captures. Original mixed-precision verification supplies the 24 synthetic controls and regression evidence. Raw samples and configurations are retained.

[2] Stanford Robotics, frame0408 bright-tabletop evaluation. Project stereo pair at 640 x 480, disparity cap 256; fixed ROI x=55:155, y=125:180. Original and Metal-normalized maps share one disparity scale. No ground truth and no hole filling for the coverage comparison.

[3] Stanford Robotics, Gaussian brightness normalization on Metal. Apple M3 Pro; 27 captures at 1248 x 688, disparity cap 128. Same normalization family and matching thresholds in CPU-preprocessing and GPU-preprocessing comparisons. More coverage can come with worse RMS error; this experiment accelerates the chosen correction rather than eliminating that tradeoff.

[4] Tankovich et al., SOS Stereo Matching in O(1) with Slanted Support Windows, IROS 2018. Leopard is an independent implementation with engineering extensions; it is not Google’s source code. The paper’s GPU performance is not used as a Leopard product claim.

SOS publication at Google Research

[5] Voynov et al., Multi-sensor large-scale dataset for multi-view 3D reconstruction, CVPR 2023. Skoltech3D provides the RealSense stereo pairs and independent scanner reference for the real-capture evaluation. The benchmark is a close-range subset, not a factory RealSense accuracy-specification test.

Skoltech3D dataset and publication

[6] Stanford Robotics, Leopard CUDA benchmark and self-contained Colab notebook. Independent CUDA port tested on NVIDIA Tesla T4. Its hardware, wrapper and timing scope differ from the CPU and Metal comparisons in this report.