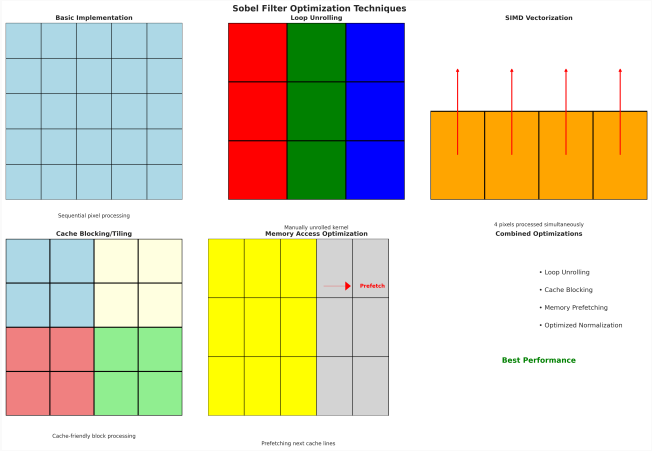
Turbo-charging the Sobel Operator

From NumPy to 2200× with modern C++

Daniel Leon

June 6, 2025

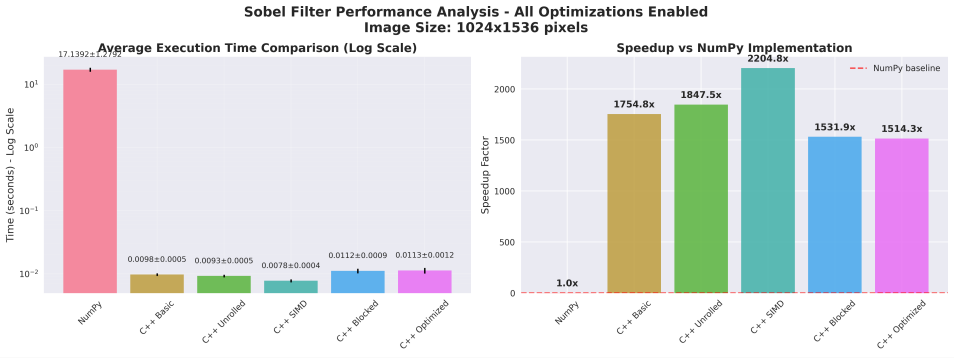
The optimisation map



Context

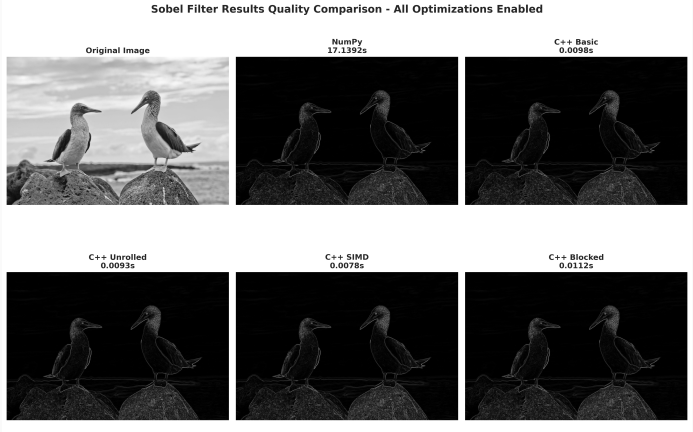
- Start: textbook kernel
- Add: loop unroll $\rightarrow$ SIMD $\rightarrow$ tiling
- 15-run mean on 1024×1536 gray frame
- i9-10885H
- GCC 13
- pybind11

Performance scoreboard



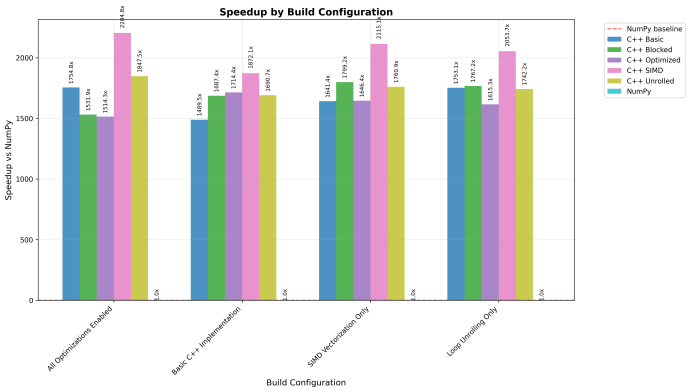
2200× faster than NumPy

Do the edges still look right?



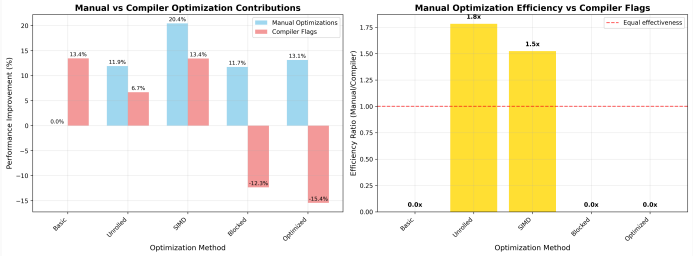
Bit-wise identical gradients — speed didn't cost accuracy.

Which combo wins where?



SIMD + unroll dominates; blocking helps on big frames. Compiler flags alone leave **15 %** on the table.

Manual vs compiler effort



- Manual tweaks still deliver **1.5–1.8× extra** over `-O3 -ffast-math`
- Best bang-for-buck: manual SIMD + `-march=native` (**+9 %**)
- Compiler flags plateau at 20 % — intrinsics still pay.

System context scaling

Test rig

- Intel i9-10885H @ 2.4 GHz
- 32 GB RAM, AVX2
- 8 cores / 16 threads

Cache blocking gets sweeter with bigger L3 caches.

Scaling hints

- Newer CPUs: $+10$–20 %
- Desktop clocks: $+15$–30 %
- AVX-512: up to $2\times$ for SIMD path
- ARM M-series: NEON close to AVX2

Get the code

github.com/dleon86/pybind11-vision-bench

Runs on Linux macOS — PRs welcome for AVX-512 or GPU ports.