Episode Book

EPISODE #6594

How Command-Line Image Tools Actually Compare

PUBLISHED	14 November 2026 at 15:06 UTC
DURATION	22:20
LISTEN	myweirdprompts.com/episode/command-line-image-tools-12052/
HOSTS	Corn, Herman, Sven, Hilbert
SCRIPT MODEL	z-ai/glm-5.3
VOICES	omnivoice
PIPELINE	V5.26
SOURCES	9 cited by the research (listed at the end)
BOOK GENERATED	15 November 2026 at 11:16 UTC

SYNOPSIS

Everyone quotes the rule that libvips is three to ten times faster than ImageMagick, yet a single decoder flag flips that benchmark on its head. Mapping the command-line image toolbox means matching the job to the right engine: libvips for streaming huge files, ImageMagick for obscure formats and pixel math, ExifTool for bulk metadata, MozJPEG for invisible compression gains, and GraphicsMagick as the legacy fork. Plus why shell commands still beat GUIs on scale and repeatability.

TRANSCRIPT



Corn

Ten thousand photos, one terminal window, and nobody clicking anything. That's the world Daniel wants us to tour today.



Herman

His question is basically: everyone knows ImageMagick exists, but what's the serious command-line image toolbox beyond it — libvips, GraphicsMagick, ExifTool, the little format optimisers — and what is each one actually best at. Plus the deeper bit: with GUI editors and AI image generators everywhere, why should anyone bother learning a shell command at all?



Corn

And the answer turns out not to be a list of tools. It's a map — job to engine. The map has surprises in it, including one where the famous performance number is wrong in both directions at once.



Herman

The framing I'd put on it is that the command-line image world is specialised, not general-purpose. ImageMagick is the universal solvent — it claims over two hundred formats and does things like Fourier transforms and connected-component labelling on pixels, which no other tool even attempts. libvips is the throughput engine — around three hundred operations, built to stream huge images through small amounts of memory. ExifTool is the metadata authority — the one that reads and writes the tags every camera and phone actually produces. MozJPEG is the encoder that beats JPEG. And GraphicsMagick is the frozen fork.



Corn

Which is a diplomatic way of saying the museum piece.



Herman

I was getting to that. The other thread, the one we'll pay off properly later: you'll see quoted everywhere that libvips is three to ten times faster than ImageMagick. That's true in some workloads, false in others, and there's a single ImageMagick flag that beats



Herman (continued)

libvips outright on the thumbnail test. So the real skill isn't picking the fast tool. It's knowing where the time goes.



Corn

I want to start with the throughput engine, then. Mostly because I have a suspicion about how you're going to describe it.



Herman

libvips. Right. The one-liner is vips thumbnail in dot jpg out dot jpg four hundred, and the reason that one line is the canonical example is what's hidden inside it. JPEG has this property where the image is stored as DCT coefficients, and if you shrink it at load time — decode only the coefficients you need — you get a massive saving. vips thumbnail does JPEG shrink-on-load by default, and it streams the file sequentially by default. No flags to remember. ImageMagick can do the same trick, but you have to know the flag and, as we'll see, where exactly to put it.



Corn

The design underneath is the interesting part. It's demand-driven — the image isn't fully decoded and held; operations pull small horizontal regions through the pipeline as needed.



Herman

With a thread per region, so it works across cores naturally. Eight-bit bands stay eight-bit until an operator needs more precision. Compare that with a typical distro ImageMagick build, which is Q16-HDRI — six to eight bytes per pixel held in a full-image cache before you've done anything. The structural difference explains most of the benchmark gaps we'll get to.



Corn

And the jobs where it pulls clearly ahead — batch thumbnailing, obviously, but also the big-image work.



Herman

The big-image work is where I think it's unmatched. There's a save operation, dzsave, that writes DeepZoom pyramids — the tiled structures map viewers and slide viewers want. And through OpenSlide it handles whole-slide microscopy scans, which are the pathological case: a slide can be a hundred thousand pixels on a side. Something that never fully materialises the image is the only sane way to touch those.



Corn

A hundred thousand pixels a side. Nothing that has to hold the whole picture survives that.



Herman

No. And the new release candidate, eight point nineteen, adds smartcrop — crop around a point of interest — and read-write of HDR colour metadata across JPEG XL, HEIF and PNG. It's an alive project.



Corn

Whereas ImageMagick, the universal solvent. I actually think its strength isn't speed at all.



Herman

It's coverage. Over two hundred formats, plus delegates that hand off to other libraries for PDF, SVG, camera RAW. When someone sends you a PSD or a DDS or a multispectral TIFF with sixty-four bands, magick is often the only thing on the machine that will even open it. And then the operations nothing else has: montage for contact sheets, composite, drawing, text annotation, Canny and Hough edge detection, the fx option for writing pixel-level mathematical expressions, perceptual hashing.



Corn

Sixty-four bands. That's not a photo, that's a satellite checking on you.



Herman

Satellite and medical imaging, exactly. Terrible for a contact sheet, indispensable for the weird file. Though the README carries a warning worth taking seriously — it advises you to set up a security policy for your environment before relying on it, because the format-handler history is long and colourful. Every extra format is extra attack surface.



Corn

ExifTool next, because I suspect it's the sleeper on this list.



Herman

It's a Perl script, effectively. One man, Phil Harvey, distribution thirteen point six three, and it needs nothing but Perl five point zero zero four — a version of Perl from the last century. And it reads and writes the tags that actually matter: EXIF, GPS, IPTC, XMP, ICC profiles, and the MakerNotes — the proprietary blocks Canon, Nikon, Sony, Fuji and the rest stash in their files. Plus it reads C2PA, the content-provenance standard, which is suddenly very relevant.



Corn

The bit that always gets me is it isn't just images.



Herman

PDF, MP4, Word documents, spreadsheets, EPUB, even torrent files and packet captures. The file-type list reads like a hoarder's manifesto. Strip the GPS coordinates from four thousand photos before you publish them — one command. Rename four thousand files by capture date — one command. That's the whole pitch. It's the bulk-operation surface.



Corn

The tool with no marketing department wins the metadata war.



Herman

It really does. Then the format-specific optimisers, and MozJPEG in particular. Three things it does: progressive encoding with a rescan optimisation you can apply losslessly to existing JPEGs through jpegtran — smaller file, not one pixel changed — trellis quantisation when converting from other formats, and quantisation tables tuned for modern high-density displays.



Corn

And its own README says the demo command-line tool isn't intended for serious use.



Herman

Which is a level of honesty you don't get from most software. The real win is that it's a drop-in, ABI-compatible replacement for libjpeg — and libvips' own install notes say, use mozjpeg if you can. So the typical path is that vips or magick was built against MozJPEG and you're getting its encoder decisions without ever typing its name.



Corn

The best optimiser on the list is the invisible one.



Herman

Last cast member, GraphicsMagick. The fork that split from ImageMagick years ago to freeze the API — stability and speed over new features. Which worked, for a while. But in libvips' benchmark table it runs the standard crop, shrink, sharpen test in about two seconds at nearly two gigabytes of peak memory — over three times slower than libvips and roughly twenty times the memory. Its raison d'être got erased by the thing it declined to race against.



Corn

And there's a research wrinkle here.



Herman

There is. As of this week, GraphicsMagick has no first-party GitHub repository — the canonical home is graphicsmagick dot org. There's a popular Node wrapper, but the project itself isn't where the rest of the ecosystem lives. When a tool's main home is a standalone dot org and the newest benchmarks show it losing badly, that's the shape of a legacy project.



Corn

Still installed everywhere, though.



Herman

Oh, everywhere. Older systems, PHP and Node stacks that picked it a decade ago. It's not going to vanish. It's just not where new work should go. Which completes the map — and now the part everyone quotes and nobody checks. The performance numbers.



Corn

I've been waiting for this part with the enthusiasm of a sloth, which is to say, completely.



Herman

The famous source is libvips' own wiki benchmark. Ten-thousand-by-ten-thousand tiled TIFF, crop a hundred pixels off each edge, shrink ten percent, sharpen, save. Sixteen cores. On that test ImageMagick comes out seven point seven times slower. That's the number that became folklore. But libvips itself attaches caveats nobody ever copies: the test does a lot of file IO and relatively little processing, which flatters libvips, and the scripting-language tools in the table don't thread automatically, so they rank lower than they deserve.



Corn

The winner wrote the exam.



Herman

They do at least say so. The more honest picture is a production benchmark an image-CDN outfit published a few weeks ago. Same two tools, realistic workloads, median of five runs on the same sixteen-core machine. And it lands somewhere different. Thumbnailing a twenty-two-and-a-half-megapixel JPEG down to four hundred pixels wide: vips takes about a hundred and thirty milliseconds at sixty-five mebibytes, plain magick resize takes three hundred and fifty-eight at two hundred and ninety mebibytes. That looks like the folklore confirming itself. But there's a decoder flag in ImageMagick that says, only decode at this size — and if you place it correctly, magick does the same thumbnail in ninety-six milliseconds at thirty-four mebibytes. Faster and leaner than vips.



Corn

The slow tool wins when someone tells it what the job is.



Herman

And the flag is free! It's not some exotic build option. It's just that the folklore says pick the fast tool, not place the right flag. Where it inverts is the multi-step pipeline — crop, resize, sharpen, save. There libvips is three point eight times faster and uses two and a half times less peak memory, because the demand-driven design pays off every time an operation touches the whole image. And on a WebP encode of a two-and-a-half-megapixel image, they tie, dead even, because both tools spend nearly all their time inside the same libwebp encoder. The encoder is the bottleneck; the wrapper is irrelevant.



Corn

So the honest summary of three to ten times faster is: sometimes seven, sometimes four, sometimes one, occasionally zero point seven.



Herman

And both of the comparative benchmarks come from interested parties — libvips' own wiki, and a commercial image CDN with a handbook about it. Neither is lying, exactly. But when the two sources with numbers both have skin in the game, that's worth saying out loud.



Corn

There's a detail in that flag story that I think is the actual argument for the whole command-line world. The placement thing.



Herman

Yes. ImageMagick seven evaluates arguments in order. The decoder flag works if it comes before the input filename. The exact same flag placed after the filename is silently ignored — no error, no warning — and you get three hundred and sixty-six milliseconds and two hundred and ninety mebibytes instead of ninety-six and thirty-four.



Corn

Eight times the memory, for the crime of ordering words wrong.



Herman

And here's why that's an argument for CLIs rather than against. That is a bug you can only find in a script, fix in the script, and keep fixed — the fix goes in version control, it's reviewed, it's still there next year. In a GUI, that same mistake is invisible forever. You clicked resize, it resized, it took however long it took. Nobody sees that a flag got dropped.



Corn

The GUI's greatest feature and greatest cover-up are the same thing — it doesn't show you what it didn't do.



Herman

That's the repeatability argument made concrete. Now, Daniel's other question — why learn any of this when GUI editors exist, and AI image tools exist. I'd answer with four things a GUI categorically cannot do. Scale, first: ExifTool's absurd file-type list is a bulk-operation surface. Thousands of files, one command, no repeated clicking. Repeatability, second — the argument-order story. Third, precision. MozJPEG's trellis quantisation, the rescan optimisation, the decoder-size flag — these are encoder-level and decoder-level decisions no slider in any editor exposes. That flag cut peak memory by eighty-eight percent in the benchmark. No GUI gives you that knob.



Corn

And the fourth is the one I find quietly satisfying.



Herman

Being the substrate. libvips is inside Mastodon, inside Rails Active Storage, inside sharp — which is what half of Node's image processing runs on — inside imgproxy, inside MediaWiki's thumbnailing. If you've ever run a Mastodon instance, you've already shipped libvips to production. The obscure CLI tool framing collapses, because the CLI is the same engine your web stack is already running. When a thumbnail bug shows up in production, vips on your laptop reproduces what production actually does. A GUI editor can't make that promise — it's a different engine entirely.



Corn

And the AI comparison — I don't think it's even a contest, it's a category error.



Herman

AI tools generate pixels. Ask one to strip GPS from four thousand JPEGs and it'll write you a poem about privacy. Nothing in that pipeline touches a file's metadata block or an encoder's quantisation table. If your job is making new pictures, the AI tool is the tool. If your job is moving, shrinking, tagging or optimising the pictures you already have, the CLI is the only tool in the room.



Corn

One more honest caveat before we get too rosy about libvips. The constant memory thing.



Herman

That's the other half of the folklore. People repeat that libvips uses constant memory regardless of image size. It doesn't. Low memory is not constant memory. It scales with image width, with each operator's window height, with your worker thread count. And the moment you rotate ninety degrees, flip vertically, or read randomly instead of sequentially, the streaming model breaks and the full decoded image lands in memory or a temp file anyway. There's also a footnote I love: libvips' own command-line tool is two point two times slower than its C API on the same test, because the CLI generates a huge amount of disc traffic. Even the fast tool's CLI is the slow version of the fast tool.



Corn

The marketing department of a benchmark table has a lot of airbrushes.



Herman

Well, and none of these tools has a marketing department. That's rather the point — you're reading source repos and a CDN handbook to find any of this out. Which is presumably why Daniel asked.



Corn

So, Sven, as our guest expert — which command-line image tool would you say keeps the best time?

S

Sven

Yes. Ze first gear, as vith any tool, must be seated before ze hands move. In my shop, a command-line image tool is a train of gears, and I am still measuring ze first one.



Herman

Could you say which gear, roughly, the first one is?

S

Sven

You ask vich is best. I cannot say. But ze libvips, it takes ze large image by small teeth, zree teeth at a time, and ze teeth are small. I counted zem once. It took me until two thousand and one.



Corn

Thorough.

S

Sven

Ze metadata, ze ExifTool, is like ze little screw behind ze faceplate. Nobody sees it. It holds ze whole clock togezzer, and if you turn it vithout care, ze dial falls off. I have seen it fall off.



Herman

Is that a caution about writing MakerNotes, or...?

S

Sven

You say ze command saves time. Ja, but vich time. Ze clock time or ze vatchmaker time. I correct your timings to ze second because ze second is vhere ze gear slips.



Corn

We'll take that under advisement.

S

Sven

And now my time runs out, and ze first gear is still only half seated. I must go. I have anozzer booking at four fifteen. Not four fourteen. Four fifteen exactly.



Corn

From the man whose release candidate is due in 2001, a note on timing.



Herman

I'll take notes on precision from anyone who's kept a booking that exact. Speaking of people with exact habits...



Hilbert

Herman, that benchmark you liked, the one from the CDN. Did they test on a clean account or one with the compositor running?



Herman

I don't think they said. Median of five runs after a warmup is all they published.



Hilbert

Mm. I ask because I ran the same thumbnail job last Tuesday, twenty-six hundred phone photos, vips against magick with the size flag in the right place, and I got nothing like their numbers. Magick was slow both ways. I spent an evening on it.



Corn

What was the difference?



Hilbert

The machine had done updates in the background. Something in there was eating a core. The timings were fine once I stopped that. But I noticed something else while I was in there. Sixty-one of the photos had the wrong date in the file name by one day. All the same day. The camera had been set wrong by whoever used it before me.



Herman

Set wrong before you took them? How do you mean?



Hilbert

I bought it second hand from a man named Dubinsky who runs a repair counter on Agrippas. He'd reset everything before selling it, except the clock. He said the clock was not his department. I set it right when I noticed, four photos in. The rest I fixed with ExifTool by matching the file numbers to the frame before.



Corn

The frame before being one you took with a camera that had the right time.



Hilbert

Same street, same minute. It held up. Dubinsky still owes me for the strap. He says straps are accessories, not repairs. I say a camera without a strap is a repair waiting to happen.



Herman

The frame-before trick is clever, I'll grant you — that's essentially manual provenance reconstruction, the thing ExifTool automates when the tags are intact in the first place.



Hilbert

The tags were intact. They were just wrong. That's worse.



Corn

Wrong and confident. The metadata equivalent of a clean benchmark from an interested party.



Herman

Which, funnily enough, is the thought the whole episode leaves me with. If the famous benchmark number is a lie of omission in both directions — seven times here, four there, a tie over there, and the slow tool winning when one flag is placed right — then the way we talk about tool performance generally is broken. Strip of workload, precision and flags, every number misleads capacity planning.



Corn

And the provenance point is the one that outlives the benchmarks. The least glamorous tool on the list — a Perl script that needs a Perl from the nineties — is the one that answers where a file actually came from, in a world where the pixels themselves are increasingly manufactured. C2PA provenance, MakerNotes, the lot.



Herman

And the most invisible one, MozJPEG, is the one making encoder decisions no editor will ever show you. The CLI isn't the retro option. It's the layer everything else is standing on — your GUI's export button, your website's thumbnails, all of it resting on libraries these commands expose.



Corn

One last thing to sit with: the next time somebody quotes you a performance multiple, ask where the time actually goes. The answer is usually one flag, one placement, or one encoder nobody mentioned.



Herman

Speaking of things nobody mentions — our producer is Hilbert Flumingtop, and he does mention, off air, at length, the straps.



Corn

If you want more of this, try episode nineteen, AI Images; episode seventeen, Cloud Render Superpowers; and episode twenty, Architectural AI. This has been My Weird Prompts. If today's map was useful, leave us a review — it's how other people with weird questions find the show.



Herman

We'll be back soon. See you then.



Corn

See you.

SOURCES

What the research for this episode read before the script was written. Primary sources first.

1. **libvips repo, README, releases v8.19.0-rc1 (2026-11-01), v8.18.7 (2026-09-26), v8.18.6 (2026-08-25); retrieved 2026-11-14. [primary]**
   https://github.com/libvips/libvips

2. **libvips wiki benchmark table, last run 2025-08-12; retrieved 2026-11-14. [primary]**
   https://github.com/libvips/libvips/wiki/Speed-and-memory-use

3. **ImageMagick repo, README, release 7.1.2-35 (2026-10-31); retrieved 2026-11-14. [primary]**
   https://github.com/ImageMagick/ImageMagick

4. **ExifTool repo, README, file-type and metadata-format tables, distribution 13.63; retrieved 2026-11-14. [primary]**
   https://github.com/exiftool/exiftool

5. **MozJPEG repo, README, release v4.1.1 (2022-08-15); retrieved 2026-11-14. [primary]**
   https://github.com/mozilla/mozjpeg

6. **libvips org page (pinned repos, star counts); retrieved 2026-11-14. [primary]**
   https://github.com/libvips

7. **WebP Cloud VIPS Handbook, Benchmarking, ImageMagick comparison, and production observability, measurements dated 2026-10-15; retrieved 2026-11-14.**
   https://vips.webp.se/en/production/benchmarking/

8. **VIPS Handbook index, states it is maintained by WebP Cloud and is not official libvips documentation; retrieved 2026-11-14.**
   https://vips.webp.se/

9. **Node GraphicsMagick wrapper, 6,970 stars, last push 2025-02-25 (surfaced via GitHub search for GraphicsMagick); retrieved 2026-11-14.**
   https://github.com/aheckmann/gm