PHOBOSLAB   Blog   Games   Projects
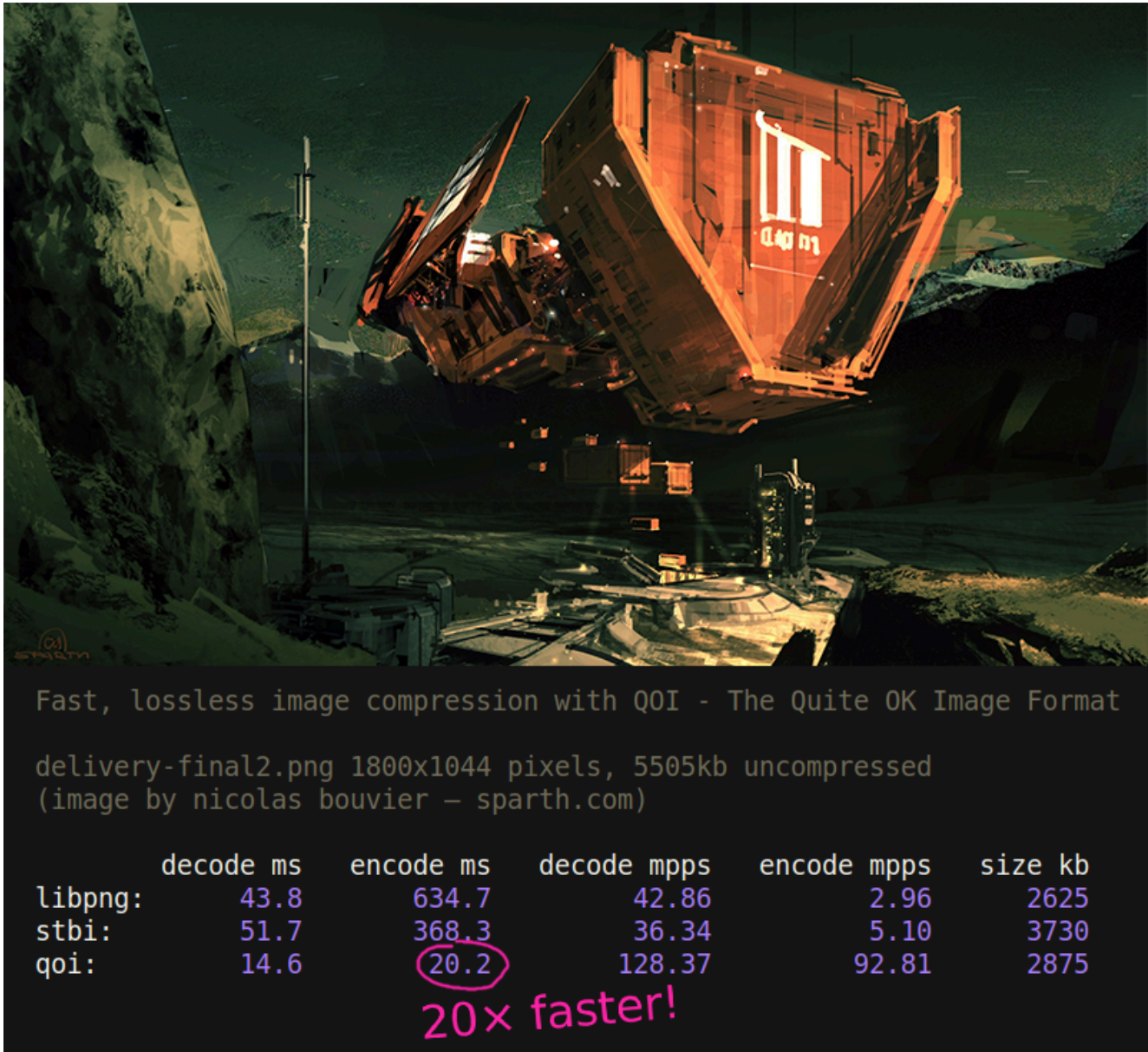
Dominic Szablewski, @phoboslab
— Wednesday, November 24th 2021

Lossless Image Compression in O(n) Time

Introducing QOI — the *Quite OK Image Format*. It losslessly compresses RGB and RGBA images to a similar size of PNG, while offering a **20x-50x** speedup in compression and **3x-4x** speedup in decompression. All single-threaded, no SIMD. It's also **stupidly simple**.

tl;dr: 300 lines of C, single header, source on github, benchmark results here.

Fast, lossless image compression with QOI - The Quite OK Image Format

delivery-final2.png 1800x1044 pixels, 5505kb uncompressed
(image by nicolas bouvier — sparth.com)

	decode ms	encode ms	decode mpps	encode mpps	size kb
libpng:	43.8	634.7	42.86	2.96	2625
stbi:	51.7	368.3	36.34	5.10	3730
qoi:	14.6	20.2	128.37	92.81	2875

20× faster!

I want to preface this article by saying that I have no idea what I'm doing. I'm not a compression guy. I barely understand how Huffman Coding and DCT works. Luckily, QOI uses neither.

I was just tinkering with some ideas that I *thought* would *maybe* compress images. The result surprised me quite a bit.

Why? A Short Rant.

File formats. They suck. I absolutely loathe the usual suspects. PNG, JPEG or even worse MPEG, MOV, MP4. They burst with complexity at the seams. Every tiny aspect screams *"design by consortium"*.

A while ago I dabbled into MPEG a bit. The basic ideas for video compression in MPEG are ingenious, even more so for 1993, but the resulting file format is an abomination.

I can almost picture the meeting of the *Moving Picture **Experts** Group* where some random suit demanded there to be a way to indicate a video stream is copyrighted. And thus, the `copyright` bit flag made its way into the standard and successfully stopped movie piracy before it even began.

MPEG, an industry standard conceived 3 decades past, all patents long expired, all professional interest abandoned. Yet, the holy specification — there named *ISO/IEC 11172-2 —* is a well guarded secret, revealed only to those that fork over a cool $200 to endow the sacred work of the ISO.

Alternative open video codecs exist, but are again immensely complex. They compete with the state of the art, require huge libraries, are compute hungry and difficult to work with. Alternatives for PNG all compete on the compression ratio too, with ever increasing complexity.

There absolutely is a market for video, audio and image codecs that trade compression ratio for speed and simplicity, but no one is serving it. (Well, these guys maybe, but it's all proprietary.)

Yes, stb_image saved us all from the pains of dealing with libpng and is therefore used in countless games and apps. A while ago I aimed to do the same for video with pl_mpeg, with some success.

But with all that we learned, why did no one go back and implement a *simple*

compression scheme to compete with PNG, but without the cruft? Why did no one implement a *simple* video compression scheme similar to MPEG, but in a *sane* file format instead?

I was tinkering to do the latter: to take parts of MPEG-1 and make it easier to parse, easier to accelerate on a GPU. A *good enough* video codec.

Instead I stumbled into a solution for the former: a lossless image format that competes with PNG for some use cases. A slightly worse compression ratio, but magnitudes less complexity.

Technical Details

QOI encodes and decodes images in a single pass. It touches every pixel just once.
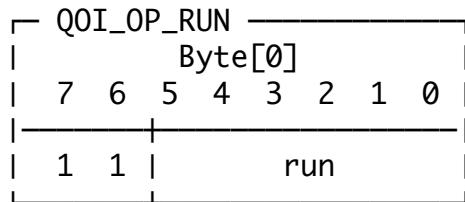
Pixels are encoded as

- a run of the previous pixel

- an index into an array of previously seen pixels

- a difference to the previous pixel value in r,g,b

- full r,g,b or r,g,b,a values

The resulting values are packed into chunks starting with a 2- or 8-bit `tag` (indicating one of those methods) followed by a number of data bits. All of these chunks (`tag` and data bits) are byte aligned, so there's no bit twiddling needed between those chunks.

The different chunk types are:

1. A run of the previous pixel

If the current pixel is exactly the same as the previous pixel, the run length is increased by 1. When a pixel is encountered that is different from the previous one, this run length is saved to the encoded data and the current pixel is packed by one of the other 3 methods.
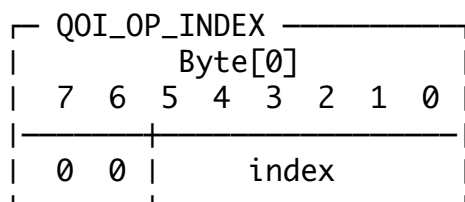
```
 ┌─ QOI_OP_RUN ─────────────┐
 │          Byte[0]         │
 │  7  6  5  4  3  2  1  0  │
 │─────────┬────────────────│
 │  1  1 │       run        │
 └───────┴──────────────────┘
2-bit tag b11
6-bit run-length repeating the previous pixel: 1..62
```

2. An index into a previously seen pixel

The encoder keeps a running array of the 64 pixels it previously encountered. When the encoder finds the current pixel still present in this array, the index into this array is saved to the stream.

To keep things `O(n)` when encoding, there's only one lookup into this array. The lookup position is determined by a "hash" of the rgba value (really just `(r * 3 + g * 5 + b * 7 + a * 11`). A linear search or some more complex bookkeeping would result in a marginally better compression ratio, but would also slow things down a bit.

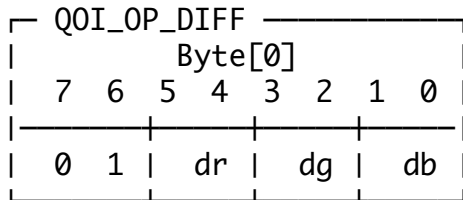```
 ┌─ QOI_OP_INDEX ───────────┐
 │   Byte[0]   │
 │ 7 6 5 4 3 2 1 0 │
 │─────────┬────────────────│
 │ 0 0 │   index   │
 └───────┴──────────────────┘
2-bit tag b00
6-bit index into the color index array: 0..63
```

### 3. The difference to the previous pixel

When the current pixel color is not too far from the previous one, the difference to the previous pixel is saved to the stream.

This comes in 2 different flavors, depending on how big the difference is. Note that this focuses on the RGB value; alpha changes are more costly.

```
┌─ QOI_OP_DIFF ──────────┐
│   Byte[0]   │
│ 7 6 5 4 3 2 1 0 │
│─────┬─────┬─────┬──────│
│ 0 1 │ dr │ dg │ db │
└─────┴─────┴─────┴──────┘
```
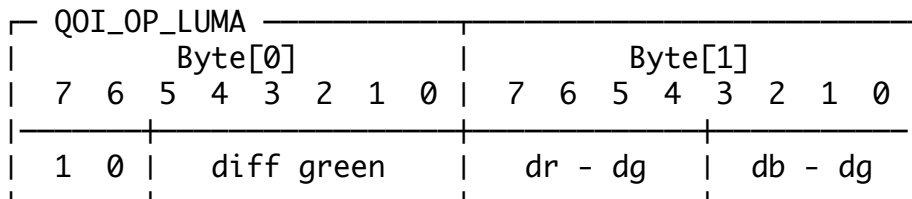```
2-bit tag b01
2-bit   red channel difference from the previous pixel -2..1
2-bit green channel difference from the previous pixel -2..1
2-bit  blue channel difference from the previous pixel -2..1
```

```
┌─ QOI_OP_LUMA ────────────┬──────────────────────────┐
│          Byte[0]         │          Byte[1]         │
│  7  6  5  4  3  2  1  0  │  7  6  5  4  3  2  1  0  │
│─────┬────────────────────┼────────────┬─────────────│
│  1  0 │    diff green     │   dr - dg   │  db - dg  │
└─────┴────────────────────┴────────────┴─────────────┘
```
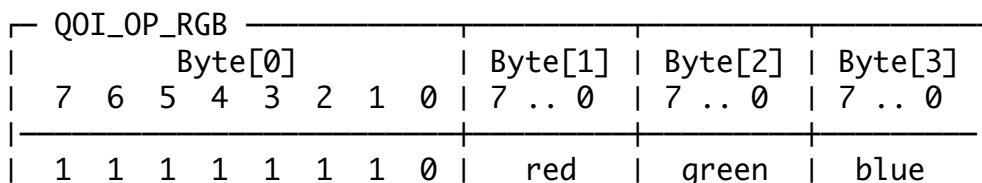
```
2-bit tag b10
6-bit green channel difference from the previous pixel -32..31
4-bit   red channel difference minus green channel difference -8..7
4-bit  blue channel difference minus green channel difference -8..7
```

4. Full rgb/rgba values

If all previous methods fail, the rgb or rgba values are saved to the stream as full bytes.

```
┌─ QOI_OP_RGB ──────────────┬─────────┬─────────┬─────────┐
│          Byte[0]         │ Byte[1] │ Byte[2] │ Byte[3] │
│  7  6  5  4  3  2  1  0  │ 7 .. 0  │ 7 .. 0  │ 7 .. 0  │
│─────────────────────────┼─────────┼─────────┼─────────│
│  1  1  1  1  1  1  1  0  │   red   │  green  │  blue   │
```

```
8-bit tag b11111110
8-bit   red channel value
8-bit green channel value
8-bit  blue channel value

  ┌─ QOI_OP_RGBA ─────────────────────────────────────────────────────────┐
  |          Byte[0]          | Byte[1] | Byte[2] | Byte[3] | Byte[4] |
  | 7  6  5  4  3  2  1  0 | 7 .. 0  | 7 .. 0  | 7 .. 0  | 7 .. 0  |
  |───────────────────────────|─────────|─────────|─────────|─────────|
  | 1  1  1  1  1  1  1  1 |   red   |  green  |  blue   |  alpha  |
  └───────────────────────────────────────────────────────────────────────┘

8-bit tag b11111111
8-bit   red channel value
8-bit green channel value
8-bit  blue channel value
8-bit alpha channel value
```

That's it.

If you have a minute, please read through the qoi.h source.

Onward

Seriously, I'm dumbfounded. BMP and TIFF have run-length-encoding and then GIF comes around with LZW. But there's nothing in between. Why? I found the space between RLE and LZW to be large enough to spend many days on. And there's a lot more to explore.

Working on QOI was a lot of fun. I had a "test runner" with some sample images lying around. Seeing how every change I made affected the compression ratio was quite exciting.

With some more work, QOI could serve as the basis for a lossless video codec, suitable for screencasts and the like.

SIMD acceleration for QOI would also be cool but (from my very limited

knowledge about some SIMD instructions on ARM), the format doesn't seem to be well suited for it. Maybe someone with a bit more experience can shed some light?

I'm also quite hyped to explore the even larger space of a *simple*, lossy image compression format. Many texture compression schemes have very exciting ideas, yet there's nothing that competes with JPEG but with *less* complexity.

© 2025 Dominic Szablewski – Imprint – powered by Pagenode (2ms) – made with <3