Compression Tools

As you have probably gathered by now, quite a lot of compression tools are available these days. This chapter outlines major multipurpose tools; interesting tools that are single-format or have special features for that format are covered in the chapter on that format. While the dream of "the single tool that does everything" hasn't been realized, most compressionists wind up with a few go-to tools that work well with the sources and platforms they target. Generally, the broader the range of jobs you need to handle, the greater the variety of tools you'll want to have to do them well.

I'm focusing on software-based tools for transcoding. Where products include live ingest or encoding I'll call that out, but I am not covering live-only encoders here.

What to Look for in a Compression Tool

There are a lot of factors to consider in choosing a compression tool, and different factors will be more or less important depending on one's needs.

Class

Different compression tools target different usage and pricing models, and what you need varies a lot depending on what you're trying to do. I break it down into a few general classes, as described in the following sections.

Consumer

Consumer encoding tools are aimed at the casual users doing their own content. Increasingly these are web-based, not a traditional local app. We're seeing a lot of adoption of free, open source tools for consumer transcoding, particularly for legally questionable tasks like transcoding from encrypted DVDs.

Workstation

Most of the products listed are in the workstation products, meaning that they're commercial products that run on single workstations. This is the province of "high-touch" encoding where preprocessing and codec settings can be tweaked.

Enterprise

Enterprise encoders are very expensive products that deliver a full workflow for high-volume encoding. Enterprise gets interesting as you start thinking about metrics like Total Cost of Ownership per encoded hour and encoded hours/day of throughput.

Open source

Many open source tools arguably could fit into one of the other buckets, but they tend to feel like their own things, with particular workflow assumptions and very deep configurability. Open source components like AVISynth and ffmpeg are also commonly used in workflows with commercial tools.

Price

Prices for compression tool software vary from free to above $100,000. Bear in mind that the compression tool itself might only be a minor part of the cost for a full system of computers, capture devices, video decks, codecs, and so on. And generally the most expensive part of the workflow is the salaries of the people running them.

I often see people getting stuck on trying to do a job they don't have the tools for, just to avoid buying a compression tool. But ask any carpenter or plumber how much it would cost them to replace their tools—expect a number in the five figures. Being able to reliably produce professional-grade work requires professional-grade tools.

It's time to buy a tool when its price equals how much time it'll save you times how much an hour you could make with that time.

Platforms

We all have our favorite platforms to work on, be it Mac, Windows, or Linux.

Windows has the bulk of professional compression tools, but there are good tools for some tasks everywhere, and unique strengths to each platform.

In the end, the OS is just another tool. Given the cost of compression workstations and software, it's better to buy the platform that runs the best tools for your job. Multiplatform workflows are easily achieved today, and having a workflow that mixes different platforms (like Macs for editing and Windows for compression) is common and quite straightforward these days.

The biggest difference between platforms is often the implementations of codecs (decoders and encoders) more than the basic functionality of the tools. For example, ProRes can only be authored on Macs (but can be decoded on Windows), and the VC-1 encoders available on Windows are higher-quality and much faster than those on Mac or Linux.

Compression Systems

While software determines what can be encoded, your hardware determines how fast it goes. And since compression is a batch process, throughput goes up almost linearly with performance. While email or even Photoshop may seem fine on a two-year-old laptop or three-year-old workstation, a decent $4000 workstation can easily be 4x faster than either of those.

There's a green factor here as well; a compression machine running full blast can consume hundreds of watts. Since newer chips have much better MIPS/watt, using older hardware can make your electrical bill and carbon emissions many times higher per minute of video.

The type, speed, and number of CPUs have the biggest impact on performance.

What kind?

Today only Intel and AMD make high performance computing CPUs suitable for mainstream compression apps (there are other vendors targeting low-power applications, but they're not good at compression). The current models from each vendor as this is written, Intel's "Nehalem" and AMD's "BK10," are both quite good. They've leapfrogged each other every few years, with Intel somewhat faster on the high end. But AMD does very nicely in terms of performance per dollar and watt.

You don't want to buy anything based on an older generation, though. The current chips have a lot of optimizations that make them quite a bit more powerful per GHz; a new 2.4 GHz i7 will run circles around an old 3 GHz P4. This comes from a whole lot of factors, but two big ones are better SIMD (Single Instruction Multiple Data) instructions that let a bunch of pixels be processed with a single command, and faster memory architectures.

Whenever a new chip comes out and is cited as "5–30 percent faster" or whatever, compression is almost always on the "30 percent" side of the range. A some SSE instructions were added explicitly to make compression faster.

How fast?

Clock speed is the easiest factor to weigh, since throughput on a given generation of processor is pretty linear; a 3 GHz "Barcelona" Opteron will be 50 percent faster than a 2 GHz "Barcelona" Opteron.

Comparisons are harder between different chips, and should be benchmarked. This can apply even within the same brand; the final generation of the Opteron and Core 2 lines were a lot faster per-GHz for compression than the first generations.

Don't stress about CPU speed too much; the fastest available CPU may only be 20 percent faster than one that costs a fraction as much. There are probably more efficient places to spend your money.

How many cores?

We're in a pervasively multicore world these days. Any decent laptop or consumer desktop is dual core, and soon to be quad-core. Quad-core is really the minimum bar for even a cheap workstation these days, with 8-core pretty standard, and with 12- and 16-core coming before long.

You want to have as many cores as your tools can use. If you're using a decoder and encoder that are only single-threaded, an 8-core machine may have most cores idle. But modern codec implementations are increasingly at least 8-way threaded. And of course if you're encoding multiple files at the same time, each can use its own cores. So, for the most part it's more cost-effective to get more cores at a slightly lower clock speed than fewer at a slightly higher speed. Ideally, performance will be roughly cores × clock speed.

The big price and complexity bump comes with number of sockets a CPU can go on. Laptops are always single-socket, so they get more cores by having more cores on a package. Workstations typically have two sockets and can handle higher-power chips, so can get more cores into each socket.

Macs and desktop versions of Windows are limited to two sockets each, which will scale up to 16 core as 8-core products become possible. But for really multithreaded stuff, 4 and beyond socket server motherboards are available and are commonly used with enterprise compression products. Those require Windows Server (the new 2008 R2 supports up to 256 cores).

Note that the current Intel and AMD memory architectures postdate Windows XP. Windows Vista, 7, and Server 2008 offer a nice performance boost on multisocket systems.

32-bit vs. 64-bit

Any CPU recent enough to be a decent encoder supports 64-bit operating systems.

Mac OS X is incrementally adding more and more 64-bit support with each version, so there's not really a choice to be made there. Windows has explicitly 32-bit and 64-bit editions, so that choice needs to be made before installation.

Most content creation apps are still 32-bit (like Premiere and After Effects CS4 and the 2009 version of Final Cut Studio). However, any app that doesn't need a specific 32-bit device driver runs fine on a 64-bit OS. So while a capture driver may need to be updated for 64-bit, file-to-file transcoding should work fine.

A big advantage to a 64-bit OS is each 32-bit app can use its own 4 GB of memory. Windows 32-bit is limited to 4 GB of RAM total, with each app limited to 2 GB (3 GB if a special parameter is set). This can pay off in a big way if you're encoding a lot of files simultaneously. For example, Expression Encoder 3's 1080p Smooth Streaming encoding is more reliable and faster running on 64-bit, even though the application is 32-bit.

There are further benefits when 64-bit apps are available:

- Compiling for 64-bit typically yields at least a 10 percent performance improvement due to additional hardware registers available in 64-bit mode.
- There's no per-app memory cap.

So, all that said, 64-bit should clearly be the default for new OS installs as long as you don't have any oddball 32-bit hardware, even if all your apps are 32-bit.

Since switching to 64-bit requires a clean install, it might not be worth upgrading just for that, but it definitely should be considered for the next update/upgrade.

How much RAM?

The simple answer to "How much RAM?" is "Enough." You never want a compression machine to run out of memory and start swapping to the hard drive. That just slays compression speed; a 10x increase in encoding time wouldn't be unusual.

Memory is cheap enough these days that's there's no point in having less than 4 GB in a laptop or 32-bit Windows system, and a 64-bit workstation should have 8 GB minimum.

Storage

One nice thing about compression is that it's bandwidth-intensive, but video is read and written in big contiguous chunks. This means expensive storage solutions like 15 K RPM or flash-based drives don't have much of an impact on compression, even though they're very useful for databases and compiling software. Ultra high-performance RAID controllers aren't needed either; motherboard RAID is generally just fine and much cheaper.

These days, just get the biggest SATA drives you can with a decent cost per GB. Compression eats up a lot of storage, so make sure to leave some room in your budget for drives. A four-drive RAID is reasonable for a standalone compression workstation.

The post-CPU future?

Like many computationally intensive areas of computing, the compression world has oscillated between using general-purpose CPUs and dedicated hardware like Digital Signal Processors (DSPs) and Application Specific Integrated Circuits (ASICs) for compression.

But the line between "hardware" and "software" encoding has blurred. CPUs contain a whole lot of DSP-like functionality in SSE, and are very parallel as well. And DSPs, GPUs, and FPGAs (Field Programmable Gate Arrays) offer much more programmability than in the past.

Today, the highest quality offline encoders run on the CPU, with hardware largely relegated to real-time broadcast encoders in the professional space (and quite a few of those are CPU-based).

In comparing performance, it can be very misleading to look at just encoding times. The appropriate benchmarks are *quality@perf* and *perf@quality*: how good can it look within a given encoding time, and how fast can I make it at a given quality level? There have been plenty of hardware encoders that produced lousy quality very quickly. But if you're willing to live with lousy quality, software encoding can delivery that quickly as well.

It's not clear to me how all this is going to evolve: are the hardware encoder technologies going to get to good enough for their speed advantage to matter? Are the CPUs going to get so fast that the quality advantage of software encoding dominates?

It's also important to remember that encoding time isn't just the encoder, but decode + preprocessing + encode. Amdahl's law states that a given optimization only impacts overall performance based on the proportion of the task being accelerated. So in a case like Blu-ray encoding from uncompressed sources off fast RAID storage, there's really no decoding or preprocessing to do, and so the encoder get 90 percent of CPU time. Speeding that up by 4x would make the new speed 10 percent + 90 percent × 4 = 370 percent of the old speed.

But transcoding from AVCHD to H.264 baseline for iPod, source decode, and scaling could be 80 percent of time. So making the H.264 encdoer 4x faster would only speed compression by the new speed is 80 + 20 × 4 = 140 percent of old speed.

Hardware acceleration

There are a couple basic approaches to hardware acceleration.

ASIC/SHED full hardware encoders

The classic hardware acceleration model (which was all we had when MPEG-2 launched) is fixed-function hardware that does everything. These are generally called ASICs—Application Specific Integrated Chips. The theory is that they're highly efficient because they're designed to do what they do well, without adding a lot of complexity in making them flexible.

The fixed-function model has certainly paid off for decoders, since there's generally one right answer.

But there's a whole lot of refinement and alchemy in compression, and once your chip tapes out, you can't do any more tweaking until you make a new chip. This lack of tuning proves to be a big disadvantage, particularly with rate control. DSP encoders can do very nicely when speed is paramount, but I've not seen any that are competitive with a modern multicore workstation in quality or speed. There are USB plug-in encoding accelerators H.264 encoders like the Elgato turbo.h264 that follow this model, but since they don't accelerate source decode, they only offer perhaps a 3–4x speed improvement with a big quality/efficiency degradation. That could be fine for transcoding to Apple TV on an older laptop, but doesn't make sense for professional use.

The initial implementations of "SHED" hardware acceleration used by Windows 7 follow the fixed function model so far (although some may support firmware updates), and are all

about getting good enough quality very quickly. Since they combine decode, preprocessing, and encoding, they also do very nicely by Amdahl's law. I'd love to see what a quality-tuned implementation could do with this new API.

GPU/DSP/FPGA-assisted acceleration

The other approach, is hardware-assisted acceleration. In this model, each part of the encode is done where it's most efficient. For example, rate control decisions and CABAC could be processed on the CPU, while DCT and motion estimation are processed on hardware. Done right, this should allow the nimbleness and rapid refinement of software with much of the performance of hardware.

There are three kinds of hardware typically used in this area: programmable Digital Signal Processors (DSPs), Field Programmable Gate Arrays (FPGAs), and good old graphic cards: Graphical Processing Units (GPUs).

FPGAs are used in the Tarari Encoder Accelerator for Windows Media and VC-1 encoding; this was the first successful attempt at the hardware-assisted approach. However, the FPGAs weren't fast enough to enable the more complex algorithms possible in software, particularly in search range. The quality for rush jobs was a lot better than software, since software couldn't use the more complex modes, but it capped out at lower level.

Newer FPGAs are a lot more powerful, and can do better. Enciris has done some very impressive demos of 1080p real-time encoding on a single FGPA.

The GPU approach is quite promising, due to how much power they have and their tight integration into the computer architecture. And GPUs are getting more powerful faster than CPUs every year.

Elemental Technologies is a startup doing some promising work in this area that is shipping several products using a hybrid CPU/GPU approach. They're accelerating the whole pipeline, which should help them continue to scale. As of summer 2009, they offer about twice the performance per dollar and per watt of pure software encoders as a similar quality.

Input Format Support

Not all applications can read all file formats. Each will support one or more architectures for reading files (at a minimum QuickTime on Mac and DirectShow on Windows).

The ability to read MPEG-2 source files requires decoders with licenses from MPEG-LA. It isn't expensive, but it may be a factor in your purchasing decision. These licensing fees are why QuickTime and Windows (before Windows 7) don't support reading MPEG-2 files out of the box, and so individual compression product vendors need to sign up for support. Support for H.264 can vary as well; QuickTime can't decode interlaced H.264 natively, so that requires a separate decoder. Dolby Digital decoding also requires a license.

Things are getting better with Mac OS X 10.6, and particularly Windows 7, with much broader format support as discussed in Chapters 28 and 29.

At a minimum, you want your software to be able to directly open files produced by all editing or capture products in your production pipeline. While it's possible to transcode to a new intermediate format as an interim step, this often entails a quality hit, and is always a big waste of time and storage.

The open source encoders are a bit of an outlier here, as they typically include support for all kinds of formats without directly covering the license fees. It's not exactly clear what's due from whom in these cases.

Preprocessing

Preprocessing is arguably the hardest part of a compression tool to get right, and there aren't any tools in the market that meet all my needs.

The most critical preprocessing feature in the HD era is scaling; going from 1920 × 1080 to 320 × 176 is a common task, and a mediocre scaler can just kill quality and a slow single-threaded scaler can really hurt performance.

Deinterlacing support varies a lot among tools. A good adaptive deinterlacing algorithm can make a huge difference in output quality. One particular area of differentiation is inverse telecine. Some tools have excellent inverse telecine support, others none at all—they treat telecined footage as any other interlaced video source, losing half the source resolution and requiring higher frame rates. Others have fragile implementations that don't work well with cadence breaks. Good inverse telecine is critical for anyone who works with NTSC content originally shot on film.

Preprocessing for analog sources requires controls for tweaking image settings (especially for luma) and noise reduction.

Lastly, a good tool will intelligently support the many aspect ratios and pixel shapes of contemporary sources. 360 × 240 should *never* be a default output. And a tool gets bonus points for doing this automatically. There's already plenty of math in compression, and I profoundly appreciate a tool that'll adjust the output frame size to the input aspect ratio.

It's rare to have much in the way of audio preprocessing in a compression tool, but a Normalize filter can be very useful when working with a variety of sources with different levels.

Output Format Support

It may sound obvious, but your encoding tool should do a good job with the formats and codecs you encode to. Beyond the checklist of what formats are supported, they should be supported by good implementations with sufficient flexibility.

The general convergence around the MPEG-LA codecs of MPEG-2, VC-1, and H.264 has been a sea change in the industry. It used to be that all compression tools sat on top of the

same SDKs, and so the actual codec versions were the same. So while we've fewer codecs to support now, there are many more implementations of those in tools, with different combinations of speed, quality, and flexibility.

And output formats themselves can vary widely in requirements. A simple MPEG-4 file isn't hard to generate, but making complaint MPEG-2 transport streams for ATSC or CableLabs is very finicky work.

Speed

Compression is one of the last tasks folks do on a computer that isn't real-time, so speed matters. And speed varies a lot among different tools. Some of this may be in quality of algorithms (basic scaling modes and deinterlacing are faster, but don't look as good). But a lot of speed comes from straight optimization, like doing video processing in Y′CbCr and taking good advantage of various SSE instructions.

Multiple-processor support is critical to getting value use out of modern workstations, and many modern tools can get almost linear scaling with the number of cores with the workflows you care about.

Automation

For high-volume encoding operations, automation is required. The easiest form is batch encoding, where multiple files can be queued up to be rendered together. Some applications also support scripting through a wide variety of APIs.

On the high end, enterprise encoding systems can offer a completely automated workflow.

The new high-end feature in enterprise automation is automated QC, where the products are able to automatically find potential quality problems for later review. It's great to have the software find where audio drops out or compression artifacts get really bad instead of having to pay someone to watch every minute of every bitrate of every clip, or just spot-checking with fingers-crossed.

Live Capture Support

Some tools can capture video and compress it in real time. Others can capture video from within the app for offline compression after the capture. A few can even do live broadcasting of an incoming stream, such as real-time recompression and rebroadcast of an ATSC stream to Smooth Streaming.

Metadata

Metadata is one of those things you may not think about it until you need it, but when you do need it, it's critical.

Most tools support at least typing in the title, copyright, and other basic fields. Where tools really vary is in their ability to handle time-based metadata, particularly captions and subtitles. A single movie can have several thousand lines of captions, and so just an interface to type them in isn't nearly enough. Ideally, there will be support for importing common caption formats and being able to extract metadata from source files.

Make sure that the tool provides a good way to import the metadata you've got, and can publish it correctly for the formats you're using. This is still a developing area; while DVD tools generally will have caption support, many products focused on streaming won't. New legal requirements for accessibility are coming into force, so make sure you know what you'll be expected to deliver in this area today and in the knowable future.

Chapter marks are another important aspect of metadata. The video editor should be able to indicate where chapters are going to go into the source file, for later recompression to apply. The start of a chapter should be encoded as an I-frame starting a new Closed GOP, for optimum speed in skipping to that frame.

Felicities

I'm a big fan of not having a computer ask me questions it could figure out itself. It's great to be able to drill down, but I want tools to do the right thing by default. Some felicities in various tools that I'm fond of:

- Default the output metadata to the input file's metadata
- Default set the output aspect ratio to match the input aspect ratio
- Automatically deinterlace if I'm going from an interlaced to progressive output
- Set the output frame rate to 23.976 if I'm using inverse telecine
- Letting me copy/paste/duplicate a setting to make a variant of it
- "Don't be dumb by default"

In general, as few clicks as possible should be required to get a decent first output.

Some felicities I've been asking for forever and haven't seen in a commercial compression tool include:

- Automatic detection and cropping of letterboxing
- A nice Adobe-style levels histogram to set luma levels

The open-source tools have been very innovative for some of these advanced features, even if the actual end-to-end workflow is typically clunky.

Full Disclosure

I've been in this compression racket for way too long now (but I'm not close to done yet!), and have spent a lot of my career advising the creators of these tools either for money or because I needed certain features for my other customers or myself.

I'm going to be as objective as I can here, but that's an impossible goal, as there are plenty of features in compression tools there because I said they should be.

So, to give some context to my biases, I'll explain what my professional and personal involvement with each product has been.

Consumer Tools

Consumer software is a challenging market in general in the web era. There's no hard line between a consumer and a workstation tool; plenty of workstation tools have approachable user interfaces and sometimes even pricing, and consumer tools can have quite advanced features. The line is more about what the product is tuned for; consumer tools are much more about converting from consumer sources to consumer playback formats, and focus on ease of use much more than batch management or metadata. And since encoding is being done for a small audience on a consumer machine, the bias is towards speed more than absolute best compression efficiency.

Even a professional workflow can include consumer tools when they have unique capabilities. The axiom "Don't be dumb by default" is particularly applicable for these products.

QuickTime Player Pro (Mac and Windows)

QuickTime Player Pro was arguably the first consumer encoder, and is certainly the oldest in common use. It was born with QuickTime (QT) itself. It was only with QuickTime 3 in Apple's dark era of declining revenue and big losses that there was ever a "non-Pro" version of QuickTime Player. Since QT 3, unlocking the advanced features required a QuickTime Pro license key, which is available for $29.99. It has no effect on apps other than QuickTime Player.

There was a decade of bitterness about this, particularly about full-screen playback having been removed from the free version. Fortunately, that was added to the basic player in QT7.2.

The new QuickTime Player for QuickTime X on Snow Leopard (Mac OS X 10.6; see Figure 1) doesn't have a Pro version, but only supports output to a limited set of formats with no configuration supported:

- Save for web (iPhone, iPhone Cellular, and Computer in a reference movie)
- Movie (just a remux without recompression of the source)
- iPhone (cellular)

- iPhone
- iPod
- AppleTV
- HD 480p
- HD 720p
- Export to YouTube
- Export to iTunes
- Export to MobileMe Gallary

While these limitations may chafe old QuickTime hands, the presets are well implemented and do a good job of picking appropriate frame size and frame rates based on the source, and certainly will be much less confusing to the casual consumer than older versions of QuickTime.

QuickTime Player 7 Pro is an optional install for 10.6 to enable more flexible export and many other features not supported by the new player (still requires that paid Pro license; existing QT7 keys work). Still, I hope we'll see more formal support for third-party extensibility come back for components like Flip4Mac.

Input format support

QuickTime Player can play any file QuickTime can, which allows a lot of extensibility. The out-of-the-box formats include the following:

- .mov (of course)

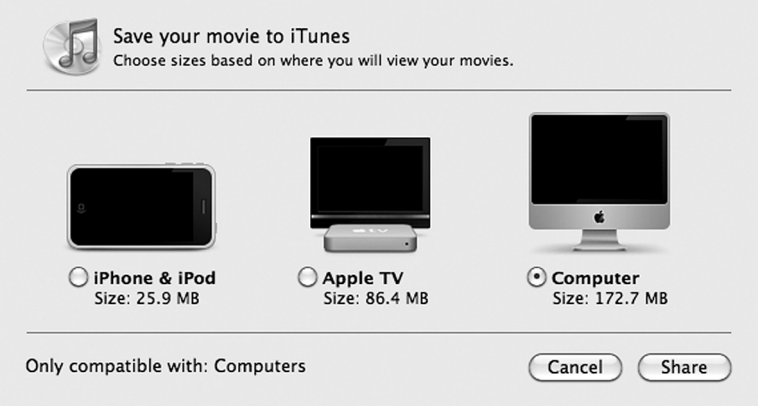


Figure 1 QuickTime Player X's iTunes export modes. It will expose only options appropriate to the frame size of the source.

- .mp4 (part 2 is Simple Profile only, H.264 is progressive up to High Profile)
- AVI (with QuickTime compatible codecs; mainly DV and uncompressed)
- Image sequences (very handy feature; can import with specified frame rate)

Beyond what's in the box, with additional codecs and components QT Player (and any other QuickTime app) can also read:

- MPEG-2 (with Apple's MPEG-2 Playback Component, 4:2:0 only, program streams and .vob, but no AC-3 decode)
- MXF, XDCAM, AVCHD, and other formats if as installed by Final Cut Pro
- WMV via Flip4Mac
- Cineform AVI or MOV via free Neo Player or any paid version

QuickTime X (on 10.6 at least) finally adds native MPEG-2 decoding.

Preprocessing

QT Pro preprocessing is quite limited (Figure 2). Some output formats like .mov and .mp4 support resizing and potentially deinterlacing; only the .mov exporter supports cropping, and even then in a very limited way. The .mov exporter supports installed QuickTime filters, but any non-trivial preprocessing should be done upstream of QT Pro.

The QTX Player's preprocessing is automatic based on source and output.

Output format support

QuickTime Player Pro supports all installed QuickTime codecs and export components. The default set includes the following:

- QuickTime .mov (whatever codecs are installed)

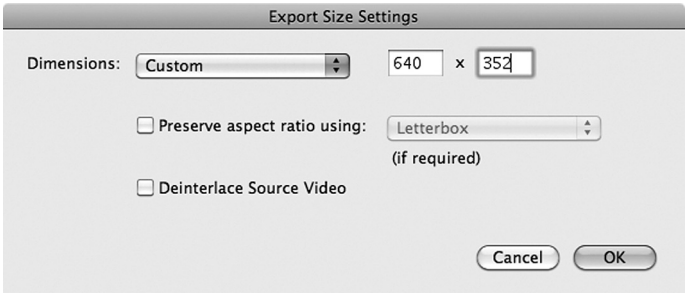


Figure 2 QuickTime Player 7's limited preprocessing settings.

- 3GPP (part 2 Simple Profile or H.264, AAC-LC, CELP, and AMR audio)
- Apple device–specific presets: AppleTV, iPod, and iPhone

One great strength of QT Pro is its ability to export without re- encoding. For example, it can do a pass-through of compatible codecs to MPEG-4, and Save As after an edit, making a new file with just the selected portions of the video.

Speed

QT Pro hands off processing to QuickTime, so that's as fast or as slow as the source and output codecs perform. Multiple encodes are run in parallel (there's no way to set serial encoding order, in fact), so you can just add additional encodes until CPU load or RAM use hits 100 percent. But they're independent, doing all decode and preprocessing each time.

Historically (and unsurprisingly) QuickTime has been notably slower on Windows than on Mac, although that gap has been closing somewhat.

Automation

QuickTime Player Pro uses AppleScript for automation on Mac and any Active Scripting language (including JavaScript and Visual Basic) on Windows; you could even control QT Pro from Excel via Visual Basic for Applications.

Live capture support

QT Pro supports capture via webcams and FireWire, but not live broadcasting. QTX added support for live screen captures. Apple's free QuickTime Broadcaster provides live RTSP encoding for QuickTime and MPEG-4.

Metadata

QT Pro can only author file-level metadata, but can pass through timecode or caption tracks if available from some source formats.

Disclosure

I've been using and enjoying QuickTime Player's authoring features for more than 15 years now, and have done presentations about and with it countless times (including Apple's WWDC and the late and lamented QuickTime Live) but haven't had any business relationships based on it.

QuickTime Player 7 Tip

Want to explore what Paleolithic codecs were like? Go into QuickTime Preferences > Advanced and check "Enable encoding using legacy codecs." You can now play with Cinepak, H.261, Sorenson Video, and the infamous Road Pizza "Video" codec. It's a great way to remind us how far we've come.

TMPGEnc Xpress (Windows)

Pegasys's TMPGEnc has been a popular consumer MPEG-2 encoder for many years. The full-featured version is now known as TMPGEnc Xpress, and handles a much broader range of formats.

Its standout features are deep MPEG-2 support and powerful preprocessing.

Input format support

TMPG mainly uses QuickTime and DirectShow for source reading, plus its own MPEG-2 decoding (including VOB and transport stream).

Preprocessing

TMPGEnc has some of the best built-in preprocessing filters, focused on fixing analog issues and other problems common in consumer sources (Figure 3). It includes a good motion-adaptive noise reducer, and an edge-enhancing Contour filter, which sharpens just edges, avoiding the noise enhancement of a typical sharpen filter. It also has very configurable inverse telecine and deinterlacing. Its audio noise reduction is a rare and welcome feature in a compression tool.

Beyond the basics, TMPGEnc has some features otherwise the province of AVISynth, like Y/C phase correction for fixing bad analog captures.

Preprocessing can be GPU accelerated using NVidia CUDA 2.0–compatible video cards.

Output format support

TMPGEnc has mainly been used for MPEG-2, particularly DVD, and uses Pegasys's own implementation for that. It includes support for custom quant matrices and manual keyframes for chapter points. See Figure 4.

TMPGEnc has broader format support, but uses the stock SDKs outside of MPEG-2:

Figure 3 TMPGEnc's Preprocessing mode. I love the magnifying glass feature.

- MPEG-2 (DVD, VideoCD, SuperVideo CD, HDV, Blu-ray, with AC-3)
- AVI (DirectShow)
- Windows Media (Format SDK)
- QuickTime (QuickTime API)
- MPEG-4 (part 2 and H.264, via MainConcept, AAC-LC/Main only)

TMPGEnc's CUDA preprocessing is fast without apparent quality loss. The "CUDA" decode depends somewhat on GPU in question—presumably it's actually using PureVideo via DXVA, not CUDA. PureVideo can produce great quality MPEG-2 decodes, but it's probably still good to test the output, particularly with an older GPU.

TMPGEnc has recently added support for the "SpursEngine" architecture, which consists of 4 of the SPE processors used in the PS3's Cell. I've not had access to SpursEngine, but

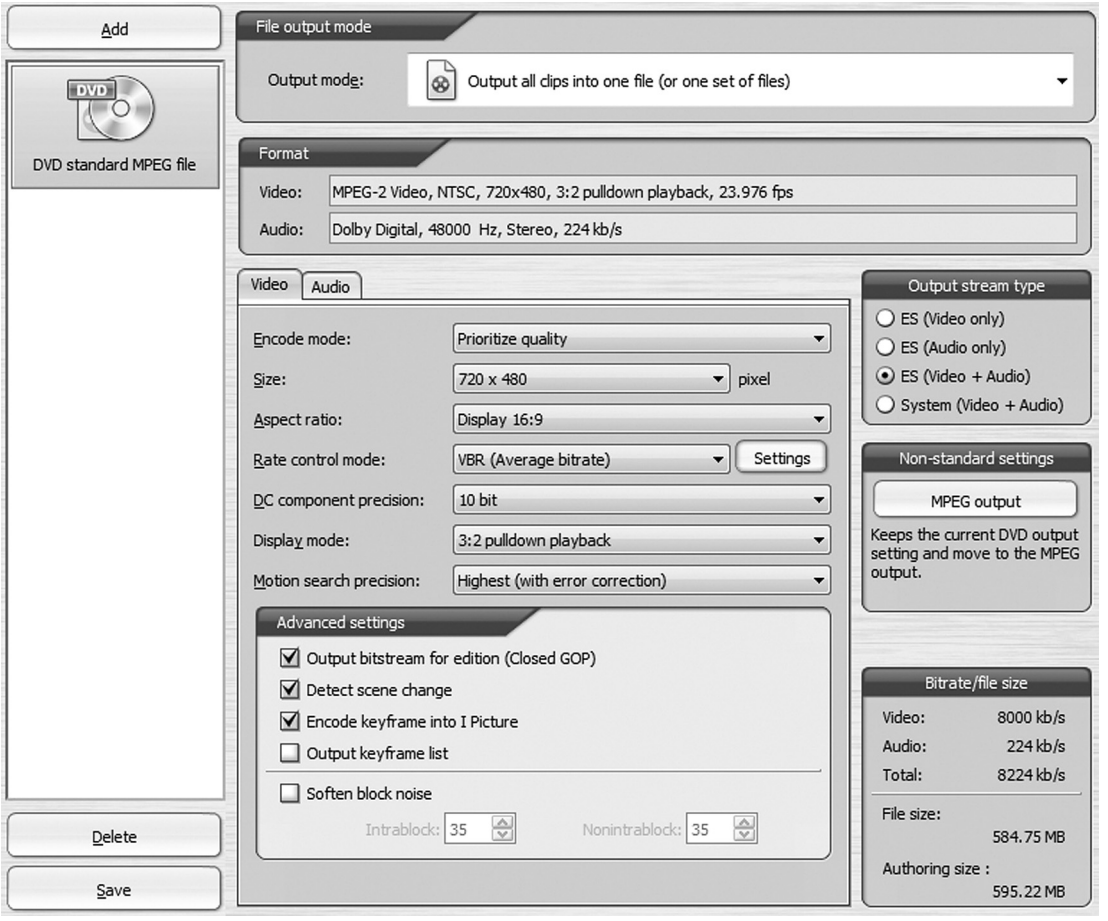


Figure 4 TMPGEnc's DVD encoding settings, its primary use.

they claim more than 4x speedup using CUDA preprocessing + SpursEngine compared to CUDA + software H.264 and nearly 6x compared to pure software encode.

Automation

TMPEG has a basic Batch queue, but nothing beyond that.

Live capture support

None.

Metadata

Just the basic header fields. There's no .scc or other caption support. That can be added later to TMPG .m2v files in a DVD-authoring app.

Disclosure

Believe it or not, I've had no business relationships with TMPG Inc.—perhaps because they're based in Japan? I like their tool, though.

Workstation Tools

Workstation tools are your classic commercial-focused tools, the compression equivalent to Photoshop, Final Cut Pro, and Excel. These generally allow for high-touch encoding, with a variety of knobs and dials to tune in high quality settings.

ProCoder/Carbon Coder (Windows)

Grass Valley ProCoder and Rhozet Carbon Coder are marketed separately today, but they're fundamentally the same product underneath, with Carbon the "pro" ProCoder with more frequent updates and much deeper support for professional media format input and output. ProCoder is fine for DVD authoring and web formats, but Carbon is required (and shines) at CableLabs, ATSC, Blu-ray, and other advanced format support.

Input format support

ProCoder supports QuickTime and DirectShow APIs, and has good built-in MPEG-2 decoders, including VOB. Carbon extends this to a panoply of video server formats like SeaChange and Grass Valley. Carbon can open pretty much anything if the right codec is installed.

Preprocessing

ProCoder introduced implicit preprocessing, a key innovation of the last decade. Thus, you can mix a bunch of source files of different aspect ratios and frame rates, as well as progressive/interlaced, and encode to a bunch of different targets varying in the same ways, and the right thing "just happens" with files getting matted, deinterlaced, or not, and so on, as appropriate. The only glaring limitation is that output aspect ratio and frame rate can't be matched to the source in GUI settings. However, this is scriptable using Carbon Server.

There's also a nice two-pane preprocessing preview that allows variable zoom (Figure 5); it's handy to make sure cropping is just right.

Output format support

The Canopus MPEG-2 codec has been a standout for years, and is particularly good at softening rather than getting blocky at low bitrates. It also has a unique quality VBR-plus-buffer mode that combines fixed quality with enforced buffer constraints. Both have nice

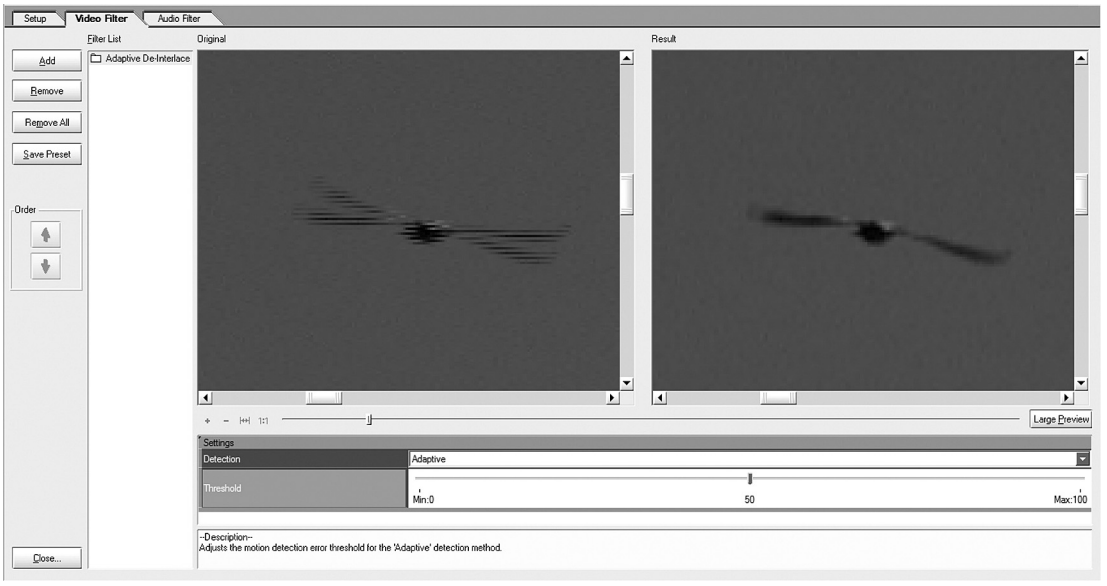


Figure 5 Carbon's preprocessing setting, which enables A/B compare while zoomed.

features like .m2ts and .vob generation, and Carbon has great support for MXF, ATSC, and other delivery formats with precise specs (Figure 6).

Speed

ProCoder was a revelation when it was launched, combining good fundamental optimization with a multithreaded pipeline. It was the first tool that would simultaneously encode multiple output files from a single input file, taking advantage of extra cores and eliminating the wasted work of multiple source decode and preprocessing passes of the same source for different outputs. The queue manager makes it easy to manage complex jobs; keeping all 16 cores of a big workstation running at once is very doable.

Automation

ProCoder and Carbon support watch folders and droplets, plus have a very powerful queue manager interface for background rendering. Carbon can also work as a client of the very rich Carbon Server API, and can be automated directly.

The queue manager can be configured with the number of simultaneous jobs.

Live capture support

Carbon supports DirectShow capture cards, including many Osprey models, for live capture of source video. Neither has any support for live streaming.

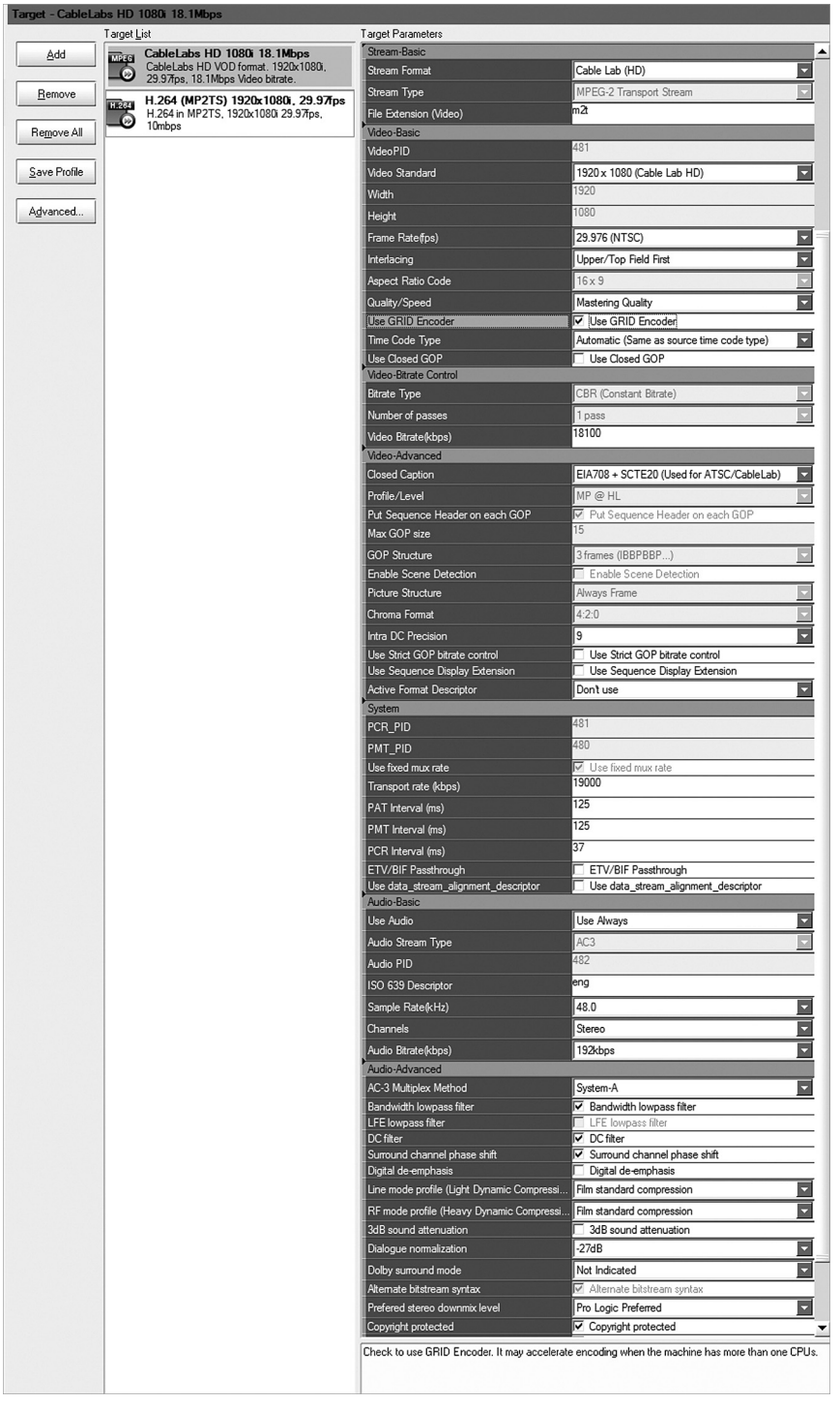


Figure 6 Carbon's CableLabs 1080i settings, following the spec to every hard-to-find detail.

Metadata

Carbon supports extraction and insertion of 608 and Line 21 captions, and muxing of SCC caption files into DVD and M2TS. Actual manipulation of the files requires another tool, but still, this is great stuff when you need it.

Otherwise Carbon and ProCoder both support the basic header metadata options.

Felicities

- Implicit preprocessing—for the win!
- Audio preview for audio filters
- Can chain multiple sources into single output file
- Select audio file as replacement audio track
- Audio time resampling for 24p < >25p and 29.97/30–style conversions
- Can select letterboxing or cropping for aspect ratio conversions
- Zoom into the preview window to easily verify crop boundaries are correct

Disclosure

I consulted with Canopus (later purchased by Grass Valley) and then Rhozet on the design and features of ProCoder and Carbon. I also worked with several of their major accounts in developing settings and workflows, including at Microsoft (it's used by MSN Video). A version of the first edition of this book was bundled with ProCoder 1.0.

Compressor (Mac)

Compressor is Apple's workstation compression tool, bundled with Final Cut Studio. Although it has great preprocessing, it's limited by the consumer-grade QuickTime encoders for popular codecs. It is used mainly for DVD compression and mezzanine files.

Input format support

Compressor can encode from anything QuickTime supports, and as it is installed with Final Cut Studio, will have access to Apple's MPEG-2 decoder and other professional formats bundled with FCS.

Compressor can also be exported to straight from the Final Cut timeline, which can save disc space by not requiring that an intermediate file be rendered in advance.

Compressor has a very nice visual importer for multichannel audio that makes it easy to take mono source files into a 5.1 group (Figure 7), and a image sequence importer handling

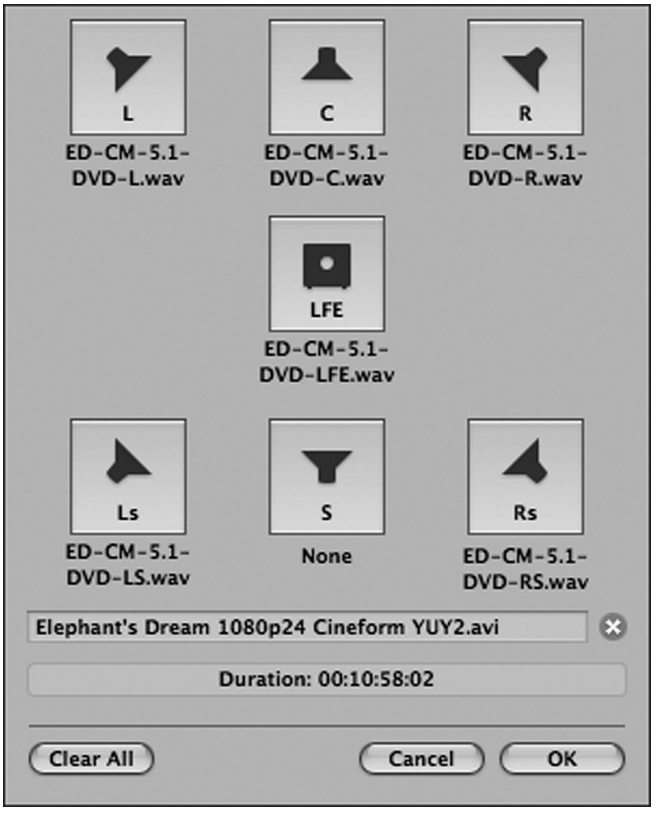


Figure 7 Compressor's multichannel audio GUI.

high-end formats like DPX. This makes Compressor very handy for working with digital intermediates, and transcoding them into more easily edited and processed formats.

Preprocessing

Preprocessing is Compressor's standout feature, derived from optical flow technology from Apple's Shake (may it rest in peace). By default it uses fast, decent algorithms; the high-end stuff is exposed via the Frame Controls pane (Figure 8). This includes great scaling and good deinterlacing. Compressor is uniquely good at frame rate resampling; for example, it can convert between 50 and 60 fps using motion estimation to synthesize new frames without introducing judder. This complex processing can be extremely slow, of course; be judicious on what your project really needs. The most expensive features are the Motion Compensated deinterlacing and rate conversion modes. But when you need them, they can do quite well (although deinterlacing still isn't as good as with the best AVISynth filters).

Beyond the Frame Controls, Compressor has some good Final Cut Pro image processing filters, including three-point color correction (albeit hobbled by a slider-based interface).

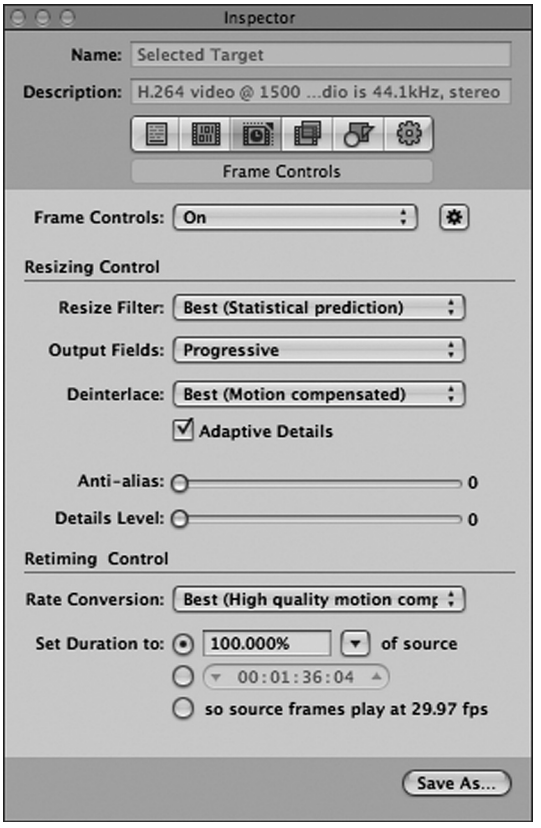


Figure 8 Compressor's Frame Controls. These settings can look pretty great, but would take a very long time to render.

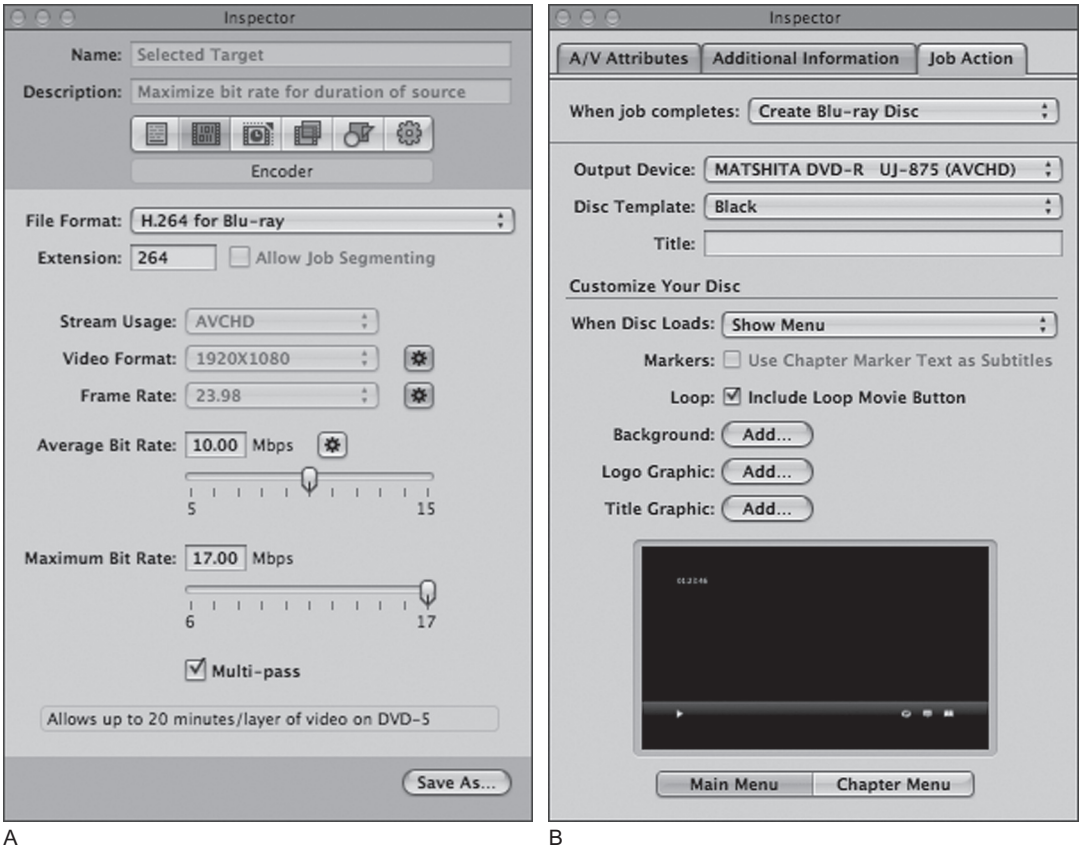
It also has decent auto-crop of letterboxing, a feature I've wanted in compression tools for ages. Overall, Compressor does a decent job of implicit preprocessing, defaulting outputs to the source aspect ratio, frame rate, and fields.

Compression supports 10-bit luma processing when working with 10-bit sources with Frame Controls.

Output format support

Compressor supports all of the QuickTime and Final Cut Pro output formats, including MPEG-2, Dolby Digital, and H.264. Unfortunately, these are just Apple's relatively limited implementations. They also suffer from QuickTime's occasional luma range mismatches, which H.264 seems particularly prone to.

Compressor was the first tool to include encoding for HD DVD, and may well be the last. Compressor 3.5 added a bunch of new export wizard-like modes. The most notable is basic Blu-ray encoding and burning in 3.5, adapting the existing codecs to Blu-ray constraints.



A B

Figures 9 (a) and (b) Compressor's Blu-ray H.264 encoding (22.9a) and mastering (22.9b) settings.

This includes H.264 progressive (in AVCHD and full Blu-ray modes; see Figure 9) and MPEG-2 progressive and interlaced. It can burn the files it encodes to a very basic disc, suitable for dailies and demos, but nothing close to full authoring, à la DVD Studio Pro's DVD and HD DVD support.

Beyond that, it can also do a similar simple DVD, and access the YouTube, MobileMe, and other QTX Player presets.

Episode can function as a Compressor plug-in, bringing support for more and deeper codec implementations.

Speed

Speed is the downside to Compressor's great preprocessing technology; complex HD preprocessing can be many times slower than the codecs themselves. In the highest-quality

mode, I've had a simple SD 480i30 to 480p60 conversion take a couple hours per minute of source on a fast dual-core system.

Compressor supports Apple's QMaster grid rendering technology, enabling encoding jobs to be split among up to 10 other Macs on the network. There is a downside to segmenting with QMaster for 2-pass VBR encoding, though: each machine does an independent 2-pass over a section of the video. So a 120-minute movie encoded over 10 machines with QMaster is treated as ten 12-minute encodes, without the ability to redistribute bits between sections. Segmenting can be turned off and QMaster can still distribute jobs between encoding nodes, but each file will be entirely encoded on a single box.

Compressor doesn't parallelize decode or preprocessing when doing multiple outputs from the same source.

Compressor Tip: Preprocess to an Intermediate

Preprocessing can take many times longer than encoding with the Frame modes. If you're doing a complex preprocessing job and are likely to do more than one version, you can save a bunch of time by doing the preprocessing to an intermediate file and then using that for further encoding. This can be faster when doing multipass encoding as well, or working with complex to decode sources like greater-than-8-bit 2K DPX image sequences.

ProRes HQ (for visually lossless Y′CbCr, including 10-bit), PNG (for lossless 8-bit RGB), Cineform (if available), or uncompressed (for 10-bit and lossless) are good choices for intermediate codec.

Automation

Compressor supports AppleScript and command-line control for automation. It also can create Droplets.

Compressor has a batch mode, but it can be clunky. All encoding is done in the separate Batch Monitor app, which makes it easy to manage big QMaster systems. However, it provides no way to take an encoded file and the edit its settings, a core feature of most compression tools. Unless you saved a setting in Compressor, you need to remake your settings from scratch. A nice touch is that Compressor can email you with job status info.

Live capture support

None, although Final Cut Pro makes a fine capture engine.

Metadata

Compressor can set file header metadata, and some time-based metadata. Beyond the basic chapter and manual I-frame (called "Compression Marker"), it also supports the Podcast Marker, which can contain a URL to an image to be played at that point in an audio podcast.

Disclosure

Apple tried to recruit me to be the Compressor product manager on a couple of occasions; that position wasn't ever filled, as they decided to bundle Compressor with Final Cut rather than make it a standalone product.

Expression Encoder (Windows)

Expression Encoder is part of the Expression Studio suite of design tools targeting Silverlight and .NET development. To that end, it's highly focused on providing deep support for Silverlight-compatible formats, and being highly usable by designers who aren't video technologists. But there are advanced encoding features a few clicks away for the more advanced user.

Earlier versions were $199 outside of the Studio bundle, but Expression Encoder 3 has been announced as the official replacement for Windows Media Encoder (which isn't officially supported on Windows 7). Thus a relatively full version of Expression Encoder is free, containing all the WMV features. The extensively named "Expression Encoder with IIS Smooth Streaming" is bundled with Expression Studio, and is available standalone for $49. It includes:

- MPEG-2, H.264, and AC-3 decoders
- H.264 and AAC encoding
- Smooth Streaming encoding (VC-1 and H.264)

Input format support

Expression Encoder supports DirectShow and QuickTime (if installed separately) APIs, and the paid version includes its own MPEG-2, AC-3, and H.264 decoders. The MPEG-2 and H.264 input is quite good, including interlaced, 4:2:2 MPEG-2, and a variety of wrapper formats, including transport stream and MXF.

Preprocessing

EEv3 has relatively simple preprocessing, but with high-quality algorithms. It uses the great SuperScaler scaling mode which works well for big resizing ratios, and a good motion-adaptive deinterlacer. It doesn't have any image processing or noise reduction, so analog sources likely will need preprocessing before import.

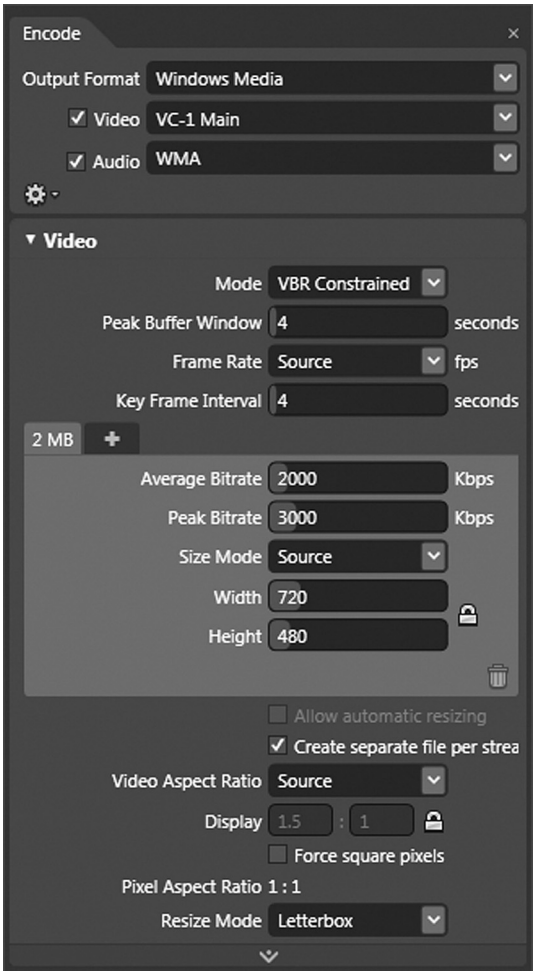


Figure 10 Expression Encoder enables a lot of "Same as Source" settings, a big timesaver when working with a variety of sources.

EE has also done the deepest implicit preprocessing of any product so far, including adjusting output aspect ratio and frame rate based on the input. The same preset can be applied to 1080i30 square-pixel, 576p50 16:9, and 480p24 4:3 files, and the right thing just happens if the source file has proper image metadata.

The only settings that may need to be tweaked per source are cropping any letterboxing and turning on inverse telecine if needed. See Figure 10.

Output format support

Expression Encoder started out as a WMV-only tool, but has expanded to new formats as Silverlight has gained support for them. Its WMV support is top-notch, launching a new

version of the VC-1 Encoder SDK (although without interlaced VC-1). V3 added 5.1 and 7.1 WMA Pro encoding.

EE's H.264 is Baseline and Main only, but with a decent implementation of both. It can do better quality than QuickTime, although it's not up to the full quality or performance of a well-tuned Main Concept or x264.

Lastly, Expression Encoder can create Smooth Streaming (.ismv) files with both VC-1/WMA and H.264/AAC codecs. The VC-1 Smooth Streaming includes both the original 1-pass CBR mode, and the new Smooth Streaming Encoder SDK implementation with 2-pass VBR and dynamic resolution switching. The H.264 Smooth in v3 is 1-pass CBR only.

Speed

EEv3 is quite fast, with multithreaded preprocessing filters and 8-way threaded versions of VC-1 and H.264. It can encode the same source to multiple data rates in parallel for both WMV (as both separate files or an Intelligent Streaming bundle) or Smooth Streaming.

EEv3 launched with an updated VC-1 Encoder SDK incorporating the Win 7 improvements of 8-way threading, and about 40 percent faster performance.

EEv3 adds the option of DXVA hardware accelerated source decode, helpful with interframe encoded sources. It can be turned off if a particular machine's decoder doesn't handle every format.

Automation

EEv3 exposes a .NET object model for extensibility and scripting for both transcoding and live streaming. This requires a certain level of programming skill, but allows for very rich programmatic control. A command-line interface using the SDK is available for traditional batch encoding.

EE can include a Silverlight media player along with the encoded file from included or user-generated templates (editable in Expression Blend), and has WebDAV publishing and publishing plug-in API for easy upload to media servers.

Live capture support

Expression Encoder 3 can capture live to file and do live broadcasting to WMV (but not Smooth Streaming). For live inputs, EEv3 supports DirectShow compatible devices, and also can do an Aero Glass–compatible screen capture. It can even do synchronized simultaneous capture or broadcast of video and screen. This is very handy for recording a presenter and presentation at the same time.

Metadata

EE can extract captions from ASF files, and import SAMI, DXFP, and XML caption formats. These can be muxed into both WMV and Smooth Streaming files, and even re-exported to XML.

Figure 11 Expression Encoder can add, import, and export time-based metadata, for both chapter marks and captions. The locations of each are shown on the timeline, triangles representing chapters and circles captions.

EE also supports chapter and keyframe markers. See Figure 11.

Disclosure

As Expression Encoder is Microsoft's primary compression product, I've advised that team on features and usability since the project was conceived, used it extensively, and have made many, many feature requests. It incorporates more of my personal "wish list" features than any other product on the market, although there are plenty more I'm already lobbying to get into v4 and beyond.

Tip: A/B Compare in Expression Encoder

One of EE's unique features is its A/B compare mode (Figure 12). Beyond the simple single-frame before/after preprocessing of past tools, it can render a user-defined section of the video, and then compare that to other compression settings. This in A/B comparison, side-by-side, or a unique Bands mode can make it very quick to dial in optimized settings.

Note that VBR encoding analyzes only the current region, so previewing on a very complex section of video with VBR may underestimate how many bits and hence what quality that section could get.

Figure 12 Expression Encoder's A-B compare. If you type in a bitrate of 100 instead of 1000 by accident, the difference is obvious even in a black-and-white print in a book.

Adobe Media Encoder (Mac and Windows)

Prior to CS4, Adobe Media Encoder (AME) was only an export mode of Premiere and After Effects. AME CS4 is now a separate app, although it can still be exported to directly from Premiere and After Effects.

Input format support

AME supports QuickTime on Mac and Windows, and DirectShow on Windows. It's built in and uses Main Concept's decoder libraries for MPEG-2 and H.264, supporting program, transport, and elementary streams.

AME can also import straight from .aep and .pproj files, very handy if you're using After Effects or Premiere Pro. Items can be added straight from either to the AME queue as well.

Preprocessing

AME's built in preprocessing is startlingly limited: just Gaussian blur, cropping, and resizing. It doesn't even have explicit deinterlacing; it will apply it dynamically if the source is flagged correctly, but that feature isn't reliable, failing with files as common as DV in AVI. Of course, Premiere and After Effects can do any kind of image processing imaginable. The blur does include a vertical-only mode, which could be used to perform a blend deinterlace, but it can get somehow very soft and blocky at the same time.

AME has an optional thumbnail preview of frames as they're encoding, but it's too small to verify quality.

AME is able to render at greater-than-8-bit to output formats that support it if Maximum Bit Depth is checked.

Output format support

Adobe gets its MPEG-2 and H.264 support from Main Concept. Beyond that, it uses the standard DirectShow, and Windows Media, and QuickTime SDKs, although the latter is limited to .mov only, without support for QuickTime export components. The Mac version thus lacks Windows Media support, and Flip4Mac isn't available, as it's an export component.

The codecs are tuned more for simplicity than flexibility. MPEG-2 in particular exposes a tiny fraction of the overwhelming set of Main Concept parameters exposed in early versions of AME. H.264 is particularly barren, with no codec level parameters beyond *profile@level*.

FLV support is provided via the non-Pro VP6 implementation from On2.

AME has one unique video format: PDF. With the Clip Mail feature, a WMV or QuickTime file can be embedded in a PDF and then have real-time comments added as an Acrobat form. It's an intriguing idea, but suffers from the integration pain of embedding the WMP and QuickTime players inside Acrobat. Note that media playback has been supported in PDF for well over a decade; it just hasn't ever been very interesting without better decoder integration.

AME supports the following delivery formats:

- AAC (LC and HE v1 and v2; up to 5.1).
- AVI (DirectShow and DV)
- Animated GIF (does anyone still use this?)
- MP3
- QuickTime (stock)
- Windows Media (Format SDK)

- FLV (Using the non-Pro version of VP6)
- F4V (Main Concept, with AAC-LC, and HE v1 and v2)
- MP4 (same as the above, plus part 2)
- Blu-ray (MPEG-2 and H.264)
- MPEG-1 (including Video CD)
- MPEG-2 (including DVD and transport stream)
- 3GP (same as previous—doesn't constrain to 3GPP-compatible settings)

AME also supports transcoding to a number of production formats like AIFF and P2.

AME suffers from a big pet peeve of mine: inconsistent units. When using WMV and some formats, bitrate is measured in Kbps, while with F4V and others, it's in Mbps (Figure 13). The use of Mbps for F4V is particularly odd, since most of the time decimal bitrates will need to used, which is clunky.

Speed

AME is decent but not spectacular. It can take advantage of multiple cores as much as the input and output codecs will allow.

One timesaver is the ability to reuse cached preview renders when working with the other CS4 video products.

Automation

AME has very basic workflow batch interface; it can't even Copy/Duplicate an existing item. It does support watch folders.

Live capture support

None; however, Premiere is a fine live capture product.

Metadata

Adobe ramped up its metadata support in CS4 with XMP, a broad effort to offer standardized metadata workflows. I applaud this effort.

AME can author and edit file-level XMP metadata, and better yet can pass metadata on from the source file or timeline. Surprisingly, it doesn't have direct support for captioning, although Premiere is quite capable here.

Disclosure

I consulted for Adobe on the design and features of the first three versions of Adobe Media Encoder, before it was a standalone application. I also consulted with Adobe on HD authoring

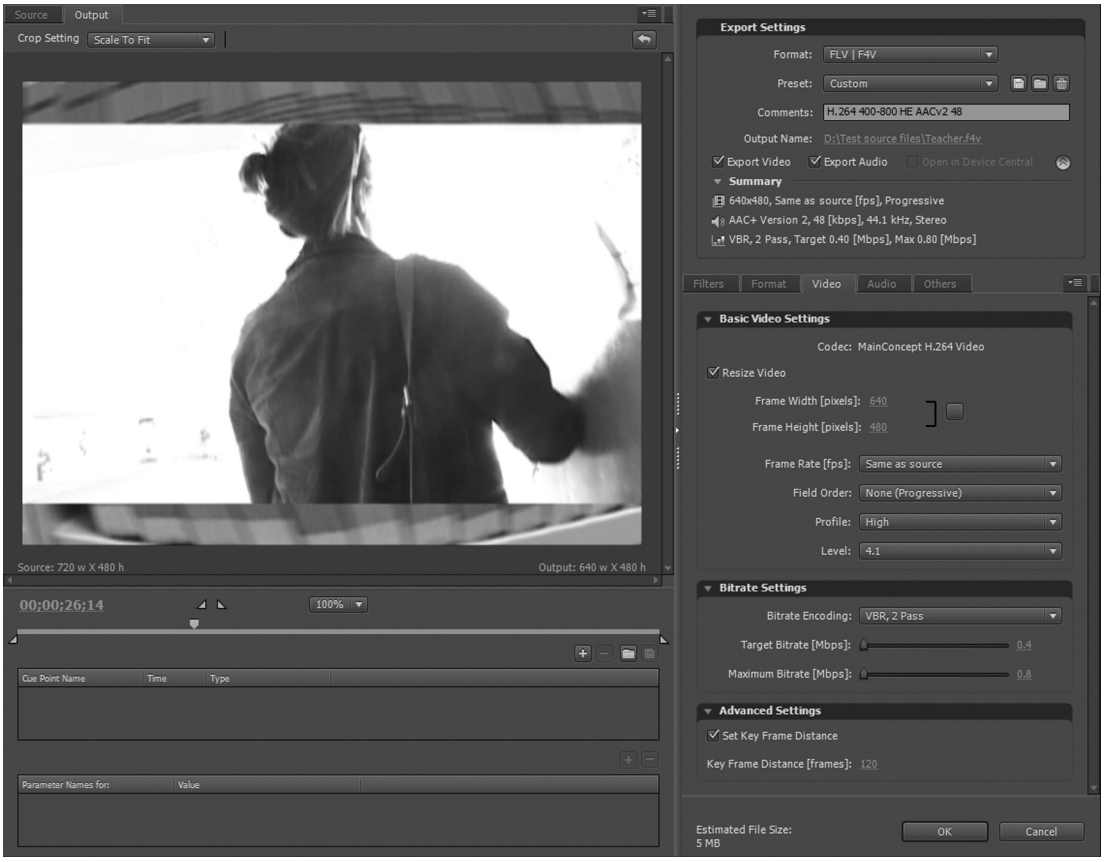


Figure 13 Adobe Media Encoder's F4V setting. Yes, that's all the video controls. Note the units are Mbps; 0.4 is really 400 Kbps.

strategy and messaging, trained Adobe's video products team in HD technology, and was lead presenter on the 2004 Microsoft/Adobe HD Roadshow.

Sorenson Squeeze (Mac and Windows)

In the late 1990s, Sorenson Media had a great symbiotic relationship with Terran Interactivity; Sorenson made the best QuickTime codec, and Terran made the best encoder for QuickTime in Media Cleaner Pro.

But as Sorenson was preparing to launch Sorenson Video 3, Cleaner was reeling from many reorganizations and loss of most of the core team. The new management wouldn't commit to support SV3 on any particular schedule.

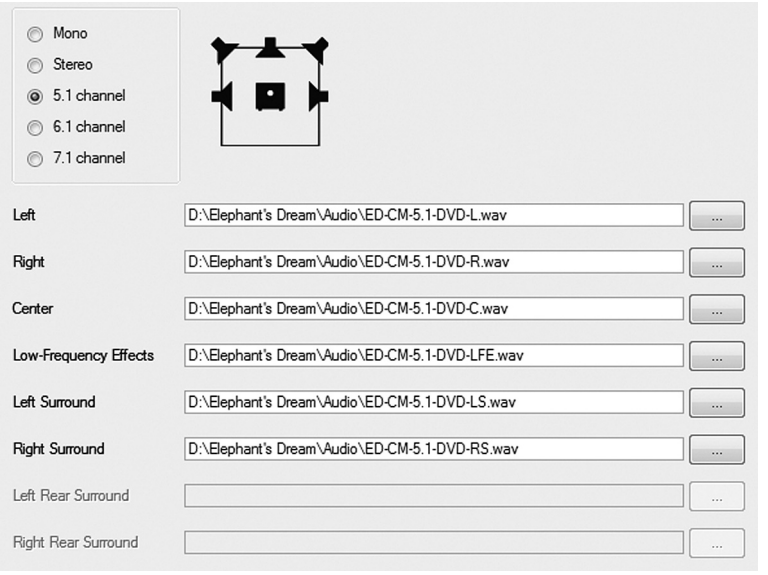


Figure 14 Squeeze's multichannel input control.

So Sorenson decided they needed their own tool, stat! and hired Cleaner's former lead developer to make Squeeze. The first version was basic, but Squeeze has evolved over the years into a credible workstation encoder. Squeeze got a big boost when Sorenson Spark was selected for FLV, becoming the leading high-end Flash compression tool.

Input format support

Squeeze mainly uses DirectShow on Windows and QuickTime cross-platform to read files. The Windows version installs ffdshow by default, which adds quite a lot of format support. Note that this is just the normal ffdshow that any app can use; if you're reinstalling Squeeze, make sure that you're not overwriting a more recent version of ffdshow that you might have installed.

Squeeze can assign a replacement audio track to a video file. It has a very handy multichannel mixer (Figure 14) that can assign mono audio channels to the appropriate channels.

Preprocessing

Squeeze (Figure 15) has long had a broad set of preprocessing features, but they weren't of great quality. Big improvements in scaling and deinterlacing finally made Squeeze 5.1 quite competitive.

Squeeze also supports the usual luma level controls, including black-and-white restore to remap values at the extremes of the range. It has a good inverse telecine as well.

Figure 15 Squeeze can get some very complex batch windows, combining multiple jobs, each with multiple sources, and with preprocessing and compression settings applied at either the job or source level.

Squeeze likes to automatically set the Auto Deinterlace and Crop filter on every file, even if your output is progressive. The default preprocessing mode can be changed or turned off in Preferences, which you should do if the default doesn't match your typical sources; they're really tuned for analog SD captures.

Output format support

For the most part, Squeeze is focused on delivery formats, particularly web formats. An optional upgrade adds the full Professional version of VP6, making it an excellent Flash encoder. Although Sorenson Media started as a codec company, except for MPEG-4 part 2 the useful implementations in Squeeze are the standard APIs from Microsoft, Apple, RealNetworks, and Main Concept.

Squeeze offers the choice of Sorenson, Apple, and Main Concept for H.264. However, that's pretty much in order of quality; always use Main Concept. Supported delivery formats include the following:

- FLV (Spark and VP6)
- AC-3
- MP3 (Fraunhofer, CBR only)
- MPEG-1 and 2 (Main Concept, supporting VCD, SVCD, DVD, HDV, ATSC, Blu-ray, and IMX)
- MPEG-4 part 2 (via the quite nice Sorenson Pro, for MP4, MOV, PSP, and 3 GPP)
- H.264 (generic MP4, Flash, PSP, and Blu-ray)
- QuickTime (any installed codecs, including Sorenson Video 3 Pro)
- RealVideo (Windows only; this an older SDK without RealAudio 10, but with RealVideo 10)
- WMV (Format SDK, including multiple bitrates for Intelligent Streaming)
- VC-1 for Blu-ray (Main Concept)

As of Squeeze 5.2, neither the VC-1 or H.264 "Blu-ray" presets are actually Blu-ray-compliant; that's due to be fixed in the next major version.

Speed

Squeeze 5 added simultaneous rendering for a big performance boost on multicore machines. However, this requires manually setting the number of simultaneous encodes, which is hard to tune if your workflow mixes codecs with different degrees of multithreading. Setting it to half your number of cores is a good default. These are parallel individual jobs, so it doesn't take advantage of simultaneous decode or preprocessing for multiple outputs from a single source.

Automation

Squeeze has a rather complex nesting structure for batch management, so a setting can be applied to all the files in a job, or just to a particular file.

Squeeze can make droplets and watch folders. It also supports command-line automation.

Live capture support

Squeeze has live capture of DV via FireWire.

Metadata

Squeeze supports basic file header metadata and chapter tracks.

Disclosure

I consulted with Sorenson Media periodically on feature planning for Squeeze 1–5. I also did a Squeeze tutorial DVD.

Fathom (Windows)

Inlet's first product was Fathom, which started life as a high-end, hardware-accelerated VC-1 encoder targeting the WMVHD, Blu-ray, and HD DVD market. That market developed more slowly than anticipated, and Inlet diversified into other markets, particularly its heralded Spinnaker live encoder series.

Inlet has continued to develop Fathom, now decoupled from the hardware and grown into a deep, industrial-grade encoder. Its UI is somewhat industrial as well, but it gets the job done.

Fathom stands out for its fine integration of transcoding and live compression (including broadcast), and exposing lots of controls for its various formats.

Input format support

Fathom will use DirectShow and QuickTime to read files, and has its own MPEG-2 decoding, including MXF and GFX. It also has support for the very high-end .yuv format used in HD authoring and codec testing.

Audio can be from the file or outboard, and Fathom can build 5.1 or 7.1 out of mono files.

While AVISynth should "just work" with most DirectShow apps, it is explicitly supported by Fathom.

Preprocessing

Fathom is designed for professional content, and so its preprocessing is more focused on transcoding clean sources than high-touch tweaking of dirty ones. It offers scaling, deinterlacing, inverse telecine, and cropping, of course. Beyond that, it only has a good but inflexible pre-filtering denoiser with just Off/Normal/Smooth settings.

Fathom supports dithering 10-bit to 8-bit with a few different modes—a great feature for high-end content, and very useful to reduce banding.

Output format support

Fathom has uncommonly deep support for its target formats, beyond that of most tools. Fathom's MPEG-2 and MPEG-4 codecs are from Main Concept and VC-1 from Microsoft, but Inlet does very down-to-the-metal integration of codecs, and a lot of their own muxer work. It supports:

- VC-1 (via Encoder SDK, for WMV and Blu-ray, with Smooth Streaming coming)
- MPEG-4 part 2 (including ISMA, PSP, and iPod)
- H.264 (MP4, transport, and elementary streams, Smooth Streaming coming)
- FLV (VP6 Pro)
- MPEG-2 (program, transport, and elementary streams)
- AVI (just Cineform, Huffyuv, DV, and 4:2:0 and 4:2:2 uncompressed)

Beyond the formats, Fathom also includes presets for a very broad array of scenarios, including web delivery, Blu-ray, ATSC, CableLabs, IPTV, Mezzanine, Digital Intermediate, and devices. These can drill down to vendor-specific settings, like for AT&T IPTV and EchoStar satellite delivery.

Fathom is the only encoder here that supports segment encoding/re-encoding, enabling just a section of a VC-1 video to have its bitrate raised or lowered. Inlet calls this "Seen by Scene." This includes validation that the re-encoded segment will mux back into the file correctly while obeying the buffer constraints.

Speed

Fathom is a very well-optimized product, with 8-way threaded implementations of most of its codecs. While it does only a single transcode at a time, it's generally able to saturate an 8-core machine.

Automation

Fathom supports watch folders with a lot of per-folder configuration possible.

One great workflow feature of Fathom is its post-encode analysis, where it can show bitrate and QP graphs of the encoded output, and flag dropped frames.

The analysis features of Fathom are available in Inlet's Semaphore standalone analysis product, and as part of the Armada enterprise workflow.

Live capture support

Fathom supports both live capture to file and live streaming. It's (once again) a deep implementation with deck control based on ALE/FLEX files (if you need them, you already

know what they are). Most DirectShow cards are supported, up to uncompressed 10-bit HD SDI. Time code is captured along with the source.

Metadata

Fathom has great metadata support, particularly in captioning. It can extract captions on the fly from Line 21 of analog sources or embedded 708/334 in digital sources, and add captions back into the same.

Live captions can be saved as SAMI files along with the capture for future use and editing.

Disclosure

I was a beta tester for Fathom before it launched, and have since worked with Inlet on their WMV, VC-1, and Smooth Streaming implementations. I've also worked on a number of major projects integrating Inlet products, particularly Spinnaker.

I've never made any money off Inlet products directly, but the Inlet crew are time-tested trade-show drinking buddies.

Episode (Mac and Windows)

Episode was originally created by Sweden's PopWire as a Sun Solaris transcoding server for mobile phone formats. When Mac OS X launched, it was enough of a Unix-based OS that porting from Solaris was feasible. And thus was born Compression Master. Telestream acquired PopWire in 2006, renamed Compression Master to Episode, and began merging the best of each company's technologies together.

Episode is now on Windows as well and supports a broad range of output formats. But it stands out with unique depth and features for mobile device encoding.

The server-side ancestor of Episode has evolved along with it and is now Telestream's Episode Engine.

Input format support

Episode can import using QuickTime on Mac and Windows and DirectShow on Windows. It also includes Telestream's own MPEG-2 decoding functionality.

Preprocessing

Episode had an early focus on the mobile market, and so was the first encoder with really good resampling for HD-to-mobile ratios (using a Bicubic with low-pass filtering).

Episode is a sort-of-implicit preprocessor. Preprocessing settings live in the Video tab, but can be tuned with rules that make it somewhat adaptive—e.g., passing through source field mode, aspect ratio, and frame size. Mismatched source and output aspect ratios can be automatically cropped or letterboxed. See Figure 16.

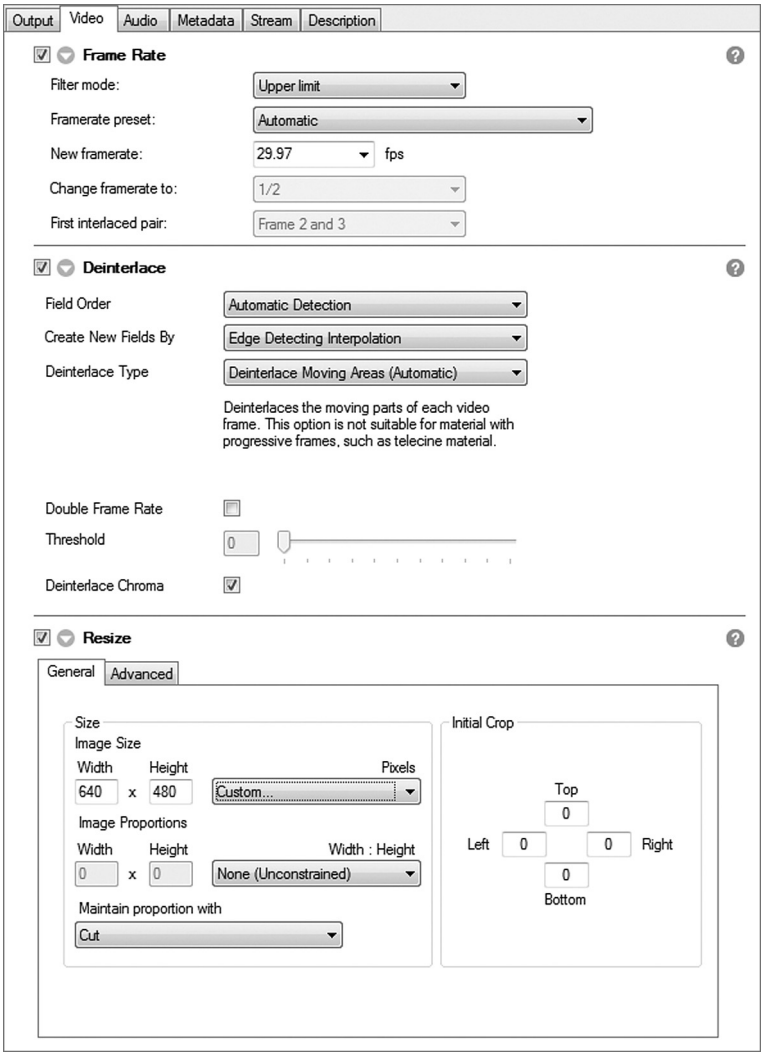


Figure 16 Episode's preprocessing settings allow for a good number of "same as source" options. I particularly like being able to set frame rate as a maximum cap, passing through lower rates.

There are some powerful filters in Episode, including the following:

- Deinterlacing via:
  - Field-to-frame bobbing
  - Fast-and-pretty good edge detection
  - Slowish-and-good motion compensation

- Highly tunable noise reduction, including optional temporal processing
- Frame rate changes with motion compensated interpolation
- LUT-based 10-bit to 8-bit conversion (which just *must* be awesomely useful for something)

Output format support

Episode's codecs are largely in-house creations, either from Telestream or acquired with PopWire. Thus Episode often has very different options than other products.

One unique feature of Episode's codecs is Lookahead-based 2-pass encoding. Instead of the classic 2-pass model where the whole file is analyzed and then the actual encode starts, Episode uses a specified number of frames (200–500) for which the analysis pass runs ahead of the encode pass. So, at 25 fps, a 500 frame Lookahead has analysis 20 seconds ahead of encode. This is faster, since the source decode and preprocessing can be cached for the second pass, and multithreading is easier with analysis and encode running on different cores. However, the limited analysis window means that 2-pass VBR can't be as efficient as a classic 2-pass VBR; Episode can't move bits from the end credits to the opening action sequence. Episode's output formats include the following:

- Windows Media (via Telestream's implementation, although somewhat different than Flip4Mac)
- FLV (H.263 or VP6 Pro, in Episode Pro)
- 3GPP (part 2 and H.264, with AMR and CELP, some modes require Pro)
- RealMedia (not on Intel Macs)
- MPEG-2 (transport and program, with AC-3)
- MP4 (part 2 or H.264, in program and transport, with AC-3 or AAC LC, HE v1 and v2)
- QuickTime (all installed codecs)
- MP3 (via LAME)
- AVI (just DV, Motion JPEG, MPEG-4 part 2, and RGB and 4:2:0 uncompressed)

Episode Pro is chock-full of presets for professional formats like GXF and MXF.

Of course, it's the mobile formats that are Episode's strength; it has dozens of well-tested presets for nearly any network and device you can imagine. And it contains mobile audio modes for AMR and CELP not supported in any of our other discussed tools.

Episode Tip

As of Episode 5.2, if you accidentally specify a peak bitrate less than the average bitrate, it'll ignore the peak value without any warning. Since Episode lists peak before average instead of after like all other tools, I'm constantly typing in average for peak and peak for average, and so the 2000 ABR 3000 PBR file I was trying to make comes out as 3000 CBR. So, double-check that you've got the right peak/average values.

Speed

The Lookahead-based two-pass is a good time saver, doubling scalability for CPU cores, and the need for a second decode and preprocess pass.

However, Episode does only a single encode at a time and many of its filters and codecs aren't broadly multithreaded, often just 1–2 threads.

Automation

Episode has a good batch window interface, but not much automation besides that. Deeper workflows are available when used with the server-side Episode Engine.

Live capture support

None.

Metadata

Episode 5.2 added support for extracting and writing VBI (Line 21 and 708) captions.

Disclosure

I consulted for PopWire on the user interface and features of Compression Master for several versions, before it was acquired by Telestream. I advised both Telestream and PopWire on the licensing of the Windows Media Porting kit they both adopted.

My most comfortable sweatshirt is Popwire swag.

Open Source Tools

MeGUI

MeGUI is one of many front-ends to open-source compression tools. Its creator is the eponymous and anonymous Doom9 of Doom9.org and its discussion forums chock-full of High Codec Nerditry.

MeGUI's strengths are reasonable usability (for an open-source tool, at least), lots of depth, and frequent updates few. It can be pretty imposing to the novice, though.

Input format support

AVISynth, and nothing but, for video. Audio can be AVISynth and other standard audio codecs, even AVI. I prefer to use the .avs for both to keep things straight.

It's thus trivial to have audio come from a different video source.

Preprocessing

MeGUI has no preprocessing itself; it exclusively supports AVISynth inputs. However, it includes a nice wizard to generate the AVS files, including cropping, deinterlacing, inverse telecine, noise reduction, and aspect ratio correction.

The resulting AVS can be used with any compatible tool.

Output format support

MeGUI offers three output formats:

- H.264 via x264 to MPEG-4 and MKV
- MPEG-4 part 2 via Xvid to AVI and MKV
- Snow (an open-source wavelet codec unlikely to be of any practical use)

MeGUI includes a well-crafted set of presets, particularly for H.264, tuned for scenarios like iPod, DVXA, PSP, PS3, Zune, and so on. Which is a good thing, because the GUI looks like (and largely is) a bunch of command-line parameters turned into fields.

For audio, LAME for MP3 and FAAC for AAC, and it can also use the free and excellent Nero AAC encoder.

Since MeGUI is a front end to a variety of command-line tools, it's quite easy to drop in updated versions of components before they've been updated in MeGUI's auto-download.

Speed

MeGUI is as fast as AVISynth and its codecs. x264 is a monster of optimization and easily scales to 16 cores (without slices!). You have to go crazy with advanced settings to make it slow on a fast machine. But throw in some complex single-threaded AVISynth filters, and that can easily bottleneck the rest of the process.

Xvid has more limited multithreading, although that's being worked on.

Automation

MeGUI as a simple queue-based batch mode.

In the end, it's really operating as a front end to a bunch of command-line utilities, and it'll show you the command lines it's using. That can make it easy to adapt settings to a pure batch mode for higher-volume encoding.

Live capture support

None.

Metadata

MeGUI doesn't have any built-in metadata support, but it can import a variety of caption formats and mux them into MP4 and MKV (the main strength of the MKV format). These are largely ripping-oriented like SubStation Alpha, Subrip, and Vobsub, plus the Blu-ray .sup format.

Disclosure

I'm a moderator on Doom9, owned by the creator of MeGUI. But I have no business relationship with the site or the tool.

Tip: MeGUI Audio

By default, MeGUI's AVS Script Creator turns audio off. If you want to use the same .avs for video and audio source, you'll want to go into the script tab and change "audio = false" to "audio = true."

Cleaner

The most notable omissions from this chapter are the Cleaner products from Autodesk. Cleaner was long the leading encoding tool, and the foundation of my early compressionist career. Created by Terran Interactive as a tool to encode cut scenes for video games, Movie Cleaner Pro 1.1 was released in 1995 as the first dedicated compression tool for computer playback. Cleaner started as a Mac-only, QuickTime only tool. Support for other formats was added in v3, and support for Windows came in v4. Cleaner's hallmarks were high-quality processing and deep integration with QuickTime, then the richest digital media platform. This extended to a very successful collaboration with Sorenson Media for tight integration of the Sorenson Video codecs, the best available in QuickTime 3-6. Cleaner introduced many core innovations we take for granted today, and some I'm hoping to take for granted again:

- Good-quality scaling (a sine resampler when most products weren't even bicubic)
- Adaptive deinterlacing

- A preview window with A/B compare slider
- Inverse telecine (implemented by request of George Lucas himself)
- Batch encoding and management
- Integrated FTP upload of encoded files
- Automation (via DCOM and command line on Windows and AppleScript on Mac)
- Ability to set the first and/or last frame to maximum quality

Cleaner had five years of glory and Terran five years of growth. In 1999, I agreed to join the company and start their consulting services division. But first I had to get married, and we were honeymooning in Barcelona when we heard that Media 100 was aquiring Terran. This seemed at the time like a great move to help Cleaner grow with the backing of the much bigger Media 100. However, Media 100 had bet big on the dot.com boom, and when that bubble burst in early 2000, things got ugly fast. The Terran founders left later that year, followed by most of the Cleaner team (myself included) in early 2001. Media 100 sold Cleaner along with the rest of its software division to the Discreet division of Autodesk later in 2001.

But with only a few employees left with long-term experience with the product, updates of Cleaner from 5.1 on were buggy and limited. Discreet tried to enter the enterprise encoding market with Cleaner Enterprise but, due to showstopper bugs, I'm not aware of any real-world deployments of it. Discreet also introduced an all-new Windows version called Cleaner XL. It was faster, more stable, and more flexible than the older Windows versions, but had none of the thoughtful usability Cleaner was famous for, and lost most of the advanced QuickTime features.

Things lurched on for a few years, culminating in a Mac OS X–compatible Cleaner 6.5 for Mac and XL 1.5 for Windows. While Autodesk still offers Cleaner for sale, the Mac version is still PowerPC-only and hasn't been updated since 2006. Cleaner XL for Windows got a minor update in 2007.

The long decline and apparent death of Cleaner was a heartbreaker for me at the time, but those ashes proved to be good fertilizer, directly leading to new compression tools being created by Cleaner veterans, or filling critical gaps in the market Cleaner left. These include:

- Sorenson Squeeze, created by Sorenson to make sure its codecs would be well-supported, with a lot of the early engineering done by the former lead developer for Cleaner

- ProCoder and thus Carbon, lead by the former VP of business development for Cleaner
- Adobe Media Encoder, once it was untenable for Adobe to ship a compression tool with Premiere from a Premiere competitor
- Compressor, with Apple having the same reasons

■