A Guide to x264, Detail Loss & Desperation (WORK IN PROGRESS)

Mosc

February 2, 2016

x264 has a ton of configurable settings, which can be quite intimidating if you're just starting out. Luckily, you won't need to change the vast majority of them. Nonetheless, the settings that do matter usually don't have sufficient official documentation. Many guides have tried to demystify the settings, but often end up being outdated, wrong, and/or overly complex. Useful information is scattered over a large number of websites. Here's my (futile) attempt fix this situation, although there's no guarantee it won't suffer from one or more of the same fates.

Contents

Encoding speed settings

Settings in this section are pretty much only dependent on how long you're willing to wait for your encode to finish. Using heavier settings can increase the quality or lower the encode's size at the same quality, but may raise the amount of time the encode takes to an unreasonable amount. Some settings may influence decoding speed as well, but not significantly so.

--preset

preset is a highly under-appreciated setting that helps you reduce the length of your command significantly by setting many speed vs. quality settings at once. *b-adapt*, *direct*, *me*, *subme*, *trellis*? You don't really need to know about any of these (as well as others) because they are handled by *preset*s. The `veryslow` *preset* is a solid base preset to use for those who care about quality. People with insane machines can go for the `placebo` *preset* for that under one percent extra improvement[1]. The `slower` *preset* is pretty decent if `veryslow` is not fast enough, and will still pass most trackers' 'gold' requirements (but be sure to check first).

If you've read other x264 guides before, you've likely been taught to use a formula to calculate an optimal *ref* value (related to something called the DPB capacity[2]). That's not needed when using *preset*, because it will automatically determine the correct number of reference frames if the *level* setting is also specified. Much easier.

Default	`medium`
Start with	`veryslow`
Consider	`slower`, `veryslow`, `placebo`

--merange

merange controls the search distance in pixels to find motion vectors, starting at a predicted motion vector. Slower *preset*s will raise the default value of `16` to `24`, but many feel that this is not high enough. The reason for this is that for high video resolutions, objects will typically move more pixels from frame to frame when compared to low resolutions. Using the same logic, we can derive that high-motion movies will benefit more from higher values.

This larger search range does come at a significant cost in encoding speed, which is why you don't want to bump this up to something crazy like `1024`

[1] `http://forum.doom9.org/showpost.php?p=1611911`

[2] `https://en.wikipedia.org/wiki/H.264/MPEG-4_AVC#Decoded_picture_buffering`

(which happens to be the limit enforced by x264). An x264 author explains why high values may even hurt quality slightly in some cases[3].

> Extremely high *merange* (e.g. >64) is unlikely to find any new motion vectors that are useful, so it may very slightly decrease compression in some cases by picking motion vector deltas so large that they worsen prediction of future motion vectors in the rare cases they're locally useful, making them worse than useless.
>
> The effect is so small as to be near-negligible, though, and you shouldn't be using such insane settings.
>
> — *Dark Shikari, 2011-12-11 (Doom9)*

In general, higher values do result in better quality, but with significant diminishing returns. Although it's possible to tweak this value to your video, you'll likely just end up setting this value based on how much extra time you want to allow x264 to spend on the encode. Which is fine.

Default 16 (24 for *preset*s veryslow and placebo)
Start with ~32
Consider 24, 32, 48, or 64 if you are insane

Compatibility settings

The compatibility settings in this section largely decide how compatible your encode will be with decoding software and hardware. Usage of more decoding resources tends to allow for more efficient encoding, and that's a trade-off that is worth thinking about.

--level

level refers to the H.264 level specification. Higher *level*s allow for higher bitrates and more reference frames, but their hardware support is generally not as good. A *level* of 4.1 is pretty much perfect for high definition encodes: it's high enough to not significantly impact quality at the resolutions we use, while having very extensive hardware support. It's the highest level supported for Blu-ray discs as well. A *level* of 4.0 is fine for standard definition video. It's generally the lowest *level* that still allows you to still use all the x264 features you might want. If hardware support is not something you're worried about, not explicitly setting *level* has the potential for a small quality boost, as x264 will automatically select an appropriate *level* based

[3] http://forum.doom9.org/showpost.php?p=1544428

on the resolution and configured reference frames. Be warned that many trackers will not accept encodes with levels past `4.1`, which could happen if you don't limit it yourself.

And, as mentioned for the *preset* setting, if you combine the *preset* and *level* settings, the amount of reference frames will be automatically limited to the maximum allowed value. How convenient!

Default	*not set*
Start with	`4.1` for high definition, `4.0` for standard definition
Consider	`4.1`, `4.0`, *not set*

--vbv-maxrate / --vbv-bufsize

VBV stands for video buffer verifier. <Insert short explanation here, emphasizing that *vbv-maxrate* is not the maximum bitrate of the video.>

The actual workings of VBV may not be completely intuitive, but the key point to remember is that there are no optimal values. It completely depends on the specifications of the playback hardware you want your video to be compatible with. An obvious candidate is `62500 / 78125`. It's the limit defined by *level* `4.1`. Value pairs for other *level*s can be derived as well[4]. So why doesn't the *level* setting in x264 apply this limit automatically, like it does for reference frames? Only because some people didn't want their bitrates to be limited to these values that are unlikely to be the exact limitations of playback devices[5]. It's a bit of a silly argument, but that's how it is.

Another possible combination of values is `40000 / 30000`. These values are based on the official Blu-ray specification[6]. The result of this is that compatibility with Blu-ray players should be pretty much guaranteed. `38000 / 30000` is a frequently used alternative, but as far as I can tell, there is no good reason to use that specific combination.

Default	*not set*
Start with	`62500 / 78125` or whatever the limits of your *level* are
Consider	`62500 / 78125` or whatever the limits of your *level* are, `40000 / 30000`, `38000 / 30000` if you want to look cool on Awesome-HD, *not set*

[4] Consult Table A-1 in `https://www.itu.int/rec/T-REC-H.264-201402-I`. MaxBR corresponds to *vbv-maxrate*, and MaxCPB to *vbv-bufsize*. These values then need to be increased by 25% because we are using the `high` *profile*, as per cpbBrVclFactor in Table A-2. Yes, it's quite strange that *level* limits depend on the *profile*, but whatever.

[5] `http://forum.doom9.org/showpost.php?p=1586636`

[6] `http://forum.doom9.org/showpost.php?p=1399419`

Source-dependent settings

This section certainly contains the most difficult settings to configure. As the section name indicates, the optimal choices for these settings depend on the characteristics of the video you're trying to encode. Sometimes you can use these characteristics to guesstimate values that would work well (and you'll get better at this with experience), other times you'll just have to test dozens of combinations of settings and stare at screenshot comparisons for hours on end. That's the life of an encoder.

--deblock

deblock controls the thresholds that are used for determining whether the H.264 deblocking filter should be applied to sharp edges. Sharp edges can be a sign of encoding artifacts, so x264 will smooth them if it thinks these are caused by a lack of bitrate. Lower *deblock* values will bias x264 more towards thinking edges are real details in the video, and should be preserved, while higher values will do the inverse. The value you set actually consists of an alpha and a beta part. The alpha value influences artifact search based on edge sharpness at the border of a block, while beta does the same based on complexity (noise, detail) within a block — higher complexity is associated with a higher chance of an edge being real[7].

> Internally, picking deblocking thresholds is playing a game of "hmmmm how much do you reckon this is a real edge and how much do you reckon this is blocking caused by low quality encode".
>
> — *F J Walter, 2014-09-04 (Doom9)*

The deblocking filter is helpful for low-bitrate video, but often hurts more than it helps in encodes that aim for transparency. `-3:-3` is therefore the de facto, safe choice for most encoders. Feel free to experiment with slightly bumped values, especially for animation, but not too high. Going over `0:0` is almost certainly an incorrect choice for our purposes.

Default `0:0`
Start with `-3:-3`
Test `-3:-3` for anything, `-3:-2` or `-2:-2` for animation

[7] `http://forum.doom9.org/showpost.php?p=1692393`

--no-mbtree

<Something about MBtree usually being bad.>

Default *not set*
Start with *set*
Test *not set*, *set*

--qcomp

Assuming *no-mbtree* used, *qcomp* controls the balance between having a constant bitrate (*qcomp* of `0`), and scenes getting whatever bitrate x264 wants them to have to achieve what the encoder considers constant quality (constant quantizers per frame type) (*qcomp* of `1`). Constant quality sounds pretty good, right? Well, not entirely. Humans are much better at noticing detail in static scenes as opposed to high-motion scenes. That means it might not be desirable to let a couple of action scenes claim a large portion of the bits you'll end up allocating to your video.

The default value is decent. If you notice that fast scenes just aren't looking as good as the slower scenes, raising the value is reasonable. Many people tend to raise the value a little bit from the default for this reason. However, lowering *qcomp* might help to improve the quality in dark scenes, an area x264 traditionally doesn't perform well in. Dark scenes typically don't contain a lot of motion, so lowering the bitrate variance can provide a welcome quality boost. It can be argued that this is a problem that should be remedied with AQ, though.

In case of MBtree being enabled, *qcomp* simply controls the strength of its effect, where values closer to `1` make it weaker. Although I personally don't have enough experience with MBtree encoding to give a solid recommendation for *qcomp* with it enabled, the consensus appears to be that its value needs to be raised from the default for optimal quality.

Default `0.6`
Start with ~`0.65` with *no-mbtree*, ~`0.75` without *no-mbtree*
Test `0.5`–`0.8` with *no-mbtree*, `0.7`–`0.8` without *no-mbtree*

--ipratio / --pbratio

The H.264 standard defines three base frame types: I, P and B. The I stands for Intra, and a frame of this type is very similar to what you would get if you would simply store a JPEG image. P-frames are predictive frames. They only contain information of parts of the frame that are different from the

previous frame. B-frames go one step further by also allowing bidirectional prediction to store differences to both previous and following frames.

P-frames calculate differences from the I-frame or another P-frame, and B-frames can depend on any other frame type. This creates a clear hierarchy: I-frames are the most important, because all other frame types eventually depend on them. B-frames are the least important, because only B-frames can depend on them. This makes it beneficial to allocate a relatively larger amount of bits to the more important frames. *ipratio* and *pbratio* control this. *ipratio* sets how much larger I-frames will be compared to P-frames, and *pbratio* sets how much smaller B-frames will be compared to P-frames.

The default values are not bad, but often not optimal. Video with lots of grain, because of its random nature, will make the difference calculation less useful, so making one frame type much bigger than another will result in a disproportionate difference in quality. It is therefore a good idea to choose values closer to `1` the more grainy the video is. `1.1` / `1.1` is a frequently chosen combination for very grainy video, while `1.3` / `1.2` is reasonable for normal film.

Of the two, *pbratio* is much more influential because a typical video will have a lot more B-frames than I-frames. It's still a good rule-of-thumb to keep the *ipratio* equal to or slightly higher than the *pbratio*. *pbratio* has no effect if MBtree is enabled, as the MBtree handles this automatically.

Default `1.4` / `1.3`
Start with ~`1.3` / ~`1.2`
Test `1.1`–`1.4` / `1.1`–`1.3`

--aq-mode / --aq-strength

<Big explanation here.>

One encoder makes a compelling argument for keeping the *aq-stength* low on animated sources.

> A quick note about *aq-strength*: It works by taking bits from high contrast areas and putting them in 'flat' areas like backgrounds. You normally want to keep that around `.6` for animation because high contrast areas translates into lines. Lines are what you notice most in animation. If you are bit-starving your lines by having a high *aq-strength* you will notice the artifacts very easily.
>
> — *Krispy, 2012-10-01 (SDBits)*

<Reference `https://hdbits.org/forums/viewtopic?topicid=58000`?>

Default `1` / `1.0`
Start with `3` / ~`0.7`
Test `1`, `2`, `3` / `0.5`–`1.0`

--psy-rd

psy-rd aims to use magic to make frames more pleasing to human eyes, even if it comes at the cost of theoretical accuracy. In practice, it means that x264 will favor encoding distorted blocks with similar complexity ('visual energy') over encoding blurry blocks, as it tends to do without this setting. The value you set consists of two parts — the first is the 'regular' psycho-visual rate distortion optimization (RDO), while the second part configures a trellis-based variant, usually referred to as psy trellis.

The RDO part is very beneficial to the quality of encodes. The default value is also actually pretty good for most sources. Higher values risk the introduction of artifacts, and will increase the bitrate required to encode the motion accurately. Lower values tend to blur more. Feel free to tweak this value up or down a bit, although it's probably not required for a good encode. Animated sources are an exception to this, as they should generally be encoded with values significantly lower than the default.

Based on the following explanation by the author, we can conclude that psy trellis attempts to do roughly the same thing.

> Psy trellis biases towards higher DCT coefficients in the reconstructed frame when quantizing residuals. This results in a sharper image and also sharper grain, but runs the risk of both artifacting and **loss** of detail due to the fact that it raises quantizers at the same bitrate.
>
> — *DarkShikari, 2008-09-21 (SDBits)*

Empirical tests have shown that psy trellis is often not that amazing compared to the non-trellis variant, as it tends to introduce ringing artifacts — especially on animated sources. Only values that initially seem very low, e.g. `0.05`, are considered to be viable if one wants to use it at all. Again, feel free to do some testing!

Default `1.0:0.0`
Start with ~`1.0:0.0` for film, ~`0.7:0.0` for animation
Test [`0.8`–`1.2`]:[`0.00`–`0.10`] for film, [`0.6`–`0.8`]:`0` for animation

--no-dct-decimate

H.264 encoding applications split up frames in blocks. If such a block has a very low amount of changes compared to the block in the previous frame, x264 will opt to not write the changes to the encoded file at all. This process, known as DCT decimation, speeds up the encode because no rate distortion optimization has to be performed. This is usually a good thing. The reason this can be turned off is that there is a risk of that low amount of changes containing something noticeable. However, even grain preservation might not be a good enough reason to disable it[8].

> [...] grain is usually complex enough that if your settings are high enough to retain the grain, DCT decimation won't affect it [...]
>
> — *Dark Shikari, 2008-01-11 (Doom9)*

On the other hand, dithering might be[9].

> Dither is extremely difficult to retain—the best method is a lot of AQ, lots of psy trellis, *no-dct-decimate*, and low *qpmin*.
>
> — *Dark Shikari, 2010-01-11 (Doom9)*

What this setting definitely will do is increase the bitrate and encoding time. The big question is whether this is better than simply increasing the bitrate in other ways, such as lowering the CRF. It is sometimes suggested that *no-dctdecimate* is particularly helpful for clean sources, such as computer-generated movies, or for very dark movies with grain that is difficult to detect. Other than that, there's no clear consensus.

Default *not set*
Start with —
Test *not set*, *set*

Advanced source-dependent settings

Thought you were done? Well, that's up to you. For the truly dedicated/insane, the fun doesn't stop here. Additional settings can be experimented with for even more potential optimization. However, note that these tend to be pretty experimental. Not much documentation is available,

[8] `http://forum.doom9.org/showpost.php?p=1085524`

[9] `http://forum.doom9.org/showpost.php?p=1376172`. Keep in mind that the complete post represents an unpopular opinion in the encoding community. We generally do want to preserve dithering in dark areas.

so you're mostly on your own. Finding optimal values *will* take much of your time. Please don't do that to yourself on your first encode!

--aq2-mode / --aq2-strength

<Insert stuff.>