f3kdb Documentation

Release 2.0

SAPikachu

November 27, 2015

Contents

f3kdb (a.k.a. flash3kyuu_deband) is a deband library and Avisynth / Vapoursynth filter, ported from an aviutl plugin.

It works by replacing banded pixels with average value of referenced pixels, and optionally add grain (random dithering) to them.

CHAPTER 1

Contents

1.1 Basic usage

- Avisynth:

```
# Note: Only planar YUV colorspaces are supported

# Load plugins, source video etc...
f3kdb(...)
```

- Vapoursynth:

```
import vapoursynth as vs

core = vs.get_core()

# Load plugins, source video etc...

last = core.f3kdb.Deband(last, ...)
```

1.2 Parameters

range
Banding detection range.

Default: 15

Y, Cb, Cr
Banding detection threshold for respective plane. If difference between current pixel and reference pixel is less than threshold, it will be considered as banded.

Default: 64

grainY, grainC
Specifies amount of grains added in the last debanding stage.

Default: 64

sample_mode
Valid modes are:

- 1: Take 2 pixels as reference pixel. Reference pixels are in the same column of current pixel. Add grain after processing.

- 2: Take 4 pixels as reference pixel. Reference pixels are in the square around current pixel. Add grain after processing.

Default: 2

seed
Seed for random number generation.

blur_first

- true: Current pixel is compared with average value of all pixels.
- false: Current pixel is compared with all pixels. The pixel is considered as banded pixel only if all differences are less than threshold.

Default: true

dynamic_grain
Use different grain pattern for each frame.

Default: false

opt
Specifies optimization mode.

- -1: Use highest optimization mode that is supported by host CPU
- 0: No optimization (Intended for testing only)
- 1: SSE2 (Pentium 4, AMD K8)
- 2: SSSE3 (Core 2)
- 3: SSE4.1 (Core 2 45nm)

Default: -1

mt
Only available in Avisynth

Multi-threaded processing. If set to true, U and V plane will be proccessed in parallel with Y plane to speed up processing.

Default: true if host has more than 1 CPU/cores, false otherwise.

dither_algo

- 1: No dithering, LSB is truncated
- 2: Ordered dithering
- 3: Floyd-Steinberg dithering

Notes:

1. Visual quality of mode 3 is the best, but the debanded pixels may easily be destroyed by x264, you need to carefully tweak the settings to get better result.
2. Mode 1 and mode 2 don't look the best, but if you are encoding at low bitrate, they may be better choice since the debanded pixels is easier to survive encoding, mode 3 may look worse than 1/2 after encoding in this situation.

   (Thanks sneaker_ger @ doom9 for pointing this out!)
3. This parameter is ignored if *output_depth* = 16.
4. 10bit x264 command-line example:

```
avs2yuv -raw "script.avs" -o - | x264-10bit --demuxer raw --input-depth 16 --input-res 1280x
```

Or compile x264 with the patch on https://gist.github.com/1117711, and specify the script directly:

```
x264-10bit --input-depth 16 --output "out.mp4" script.avs
```

Default: 3

keep_tv_range
If set to true, all processed pixels will be clamped to TV range (luma: 16 ~ 235, chroma: 16 ~ 240).

- It is recommended to set this to true for TV-range videos, since pixel values may overflow/underflow after dithering.
- DON'T set this to true for full-range videos, as all out-of-range pixels will be clamped to TV range.

Default: false

input_mode
Not available in Vapoursynth, since this can be inferred from clip properties

Specify source video type.

- 0: Regular 8 bit video
- 1: 9 ~ 16 bit high bit-depth video, stacked format
- 2: 9 ~ 16 bit high bit-depth video, interleaved format

Default: 0 (*input_depth* = 8 or not specified) / 1 (*input_depth* > 8)

input_depth
Not available in Vapoursynth, since this can be inferred from clip properties

Specify bit-depth of source video.

Range: 8 ~ 16

Default: 8 (*input_mode* = 0 or not specified) / 16 (*input_mode* = 1 or 2)

output_mode
Not available in Vapoursynth, f3kdb will only output high bitdepth clip in native format

Specify output video type. Meaning of values are the same as *input_mode*.

When *output_mode* = 2, frames will be 2x wider and look garbled on preview, it will return to normal after correctly encoded by high bit-depth x264)

Default: 0 (*output_depth* = 8 or not specified) / 1 (*output_depth* > 8)

output_depth
Specify output bit-depth.

If *output_depth* = 16, dither algorithm specified by *dither_algo* won't be applied.

Range: 8 ~ 16

Default: 8 (*output_mode* = 0 or not specified) / 16 (*output_mode* = 1 or 2)

random_algo_ref, random_algo_grain
Choose random number algorithm for reference positions / grains.

- 0: Algorithm in old versions
- 1: Uniform distribution

- 2: Gaussian distribution

  (StdDev (sigma) is settable through *random_param_ref* / *random_param_grain*, Only values in [-1.0, 1.0] is used for multiplication, numbers outside this range are simply ignored)

Default: 1 / 1

random_param_ref, random_param_grain
Parameter for the respective random number generator. Currently only have effect for the Gaussian generator.

Default: 1.0

preset
Use preset parameters. Preset will be applied before other parameters so that you can easily override individual parameter.

See available presets

1.3 Available presets

Name	Parameters
depth	y=0/cb=0/cr=0/grainy=0/grainc=0
low	y=32/cb=32/cr=32/grainy=32/grainc=32
medium	y=48/cb=48/cr=48/grainy=48/grainc=48
high	y=64/cb=64/cr=64/grainy=64/grainc=64
veryhigh	y=80/cb=80/cr=80/grainy=80/grainc=80
nograin	grainy=0/grainc=0
luma	cb=0/cr=0/grainc=0
chroma	y=0/grainy=0

Presets can also be combined together, for example "medium/nograin" is the same as y=48/cb=48/cr=48/grainy=0/grainc=0 .

1.4 C++ API

Note: Due to the use of some C++ feature in header files, they can't be compiled under pure C without hacking. This will be change in future.

Please check the following example to see how to use f3kdb in your program:

```
// Compile with: g++ -std=c++11 -Wall -Wextra example.cpp -lf3kdb

#include <stdio.h>
#include <stdint.h>
#include <f3kdb/f3kdb.h>

int main()
{
    static const int FRAME_WIDTH = 640;
    static const int FRAME_HEIGHT = 480;

    f3kdb_video_info_t vi;
    vi.width = FRAME_WIDTH;
    vi.height = FRAME_HEIGHT;

    // YUV 4:2:0
```

```
    vi.chroma_width_subsampling = 1;
    vi.chroma_height_subsampling = 1;

    // 8-bit video
    vi.pixel_mode = LOW_BIT_DEPTH;
    vi.depth = 8;

    // Set this to estimated value if frame count is unknown
    vi.num_frames = 42;

    f3kdb_params_t params;
    int result;
    result = f3kdb_params_init_defaults(&params); // Always call this to initialize parameters to co
    if (result != F3KDB_SUCCESS) {
        printf("f3kdb_params_init_defaults code = %d\n", result);
        return result;
    }

    // Fill parameters. You can also directly modify params
    result = f3kdb_params_fill_preset(&params, "medium/nograin");
    if (result != F3KDB_SUCCESS) {
        printf("f3kdb_params_fill_preset code = %d\n", result);
        return result;
    }
    result = f3kdb_params_fill_by_string(&params, "seed=42/dither_algo=2");
    if (result != F3KDB_SUCCESS) {
        printf("f3kdb_params_fill_by_string code = %d\n", result);
        return result;
    }

    // No need to call f3kdb_params_sanitize or f3kdb_video_info_sanitize,
    // unless you want to inspect how the parameter object look like after
    // replacing default values with inferred values

    f3kdb_core_t* core;
    char extra_error[1024];
    result = f3kdb_create(&vi, &params, &core, extra_error, sizeof(extra_error));
    if (result != F3KDB_SUCCESS) {
        printf("f3kdb_create code = %d, msg=%s\n", result, extra_error);
        return result;
    }

    // For demonstration purpose, we just try to deband some garbage data
    static const int buffer_size = FRAME_WIDTH * 480 + 15; // Keep room for 16-byte alignment
    unsigned char src_buffer[buffer_size];
    unsigned char dst_buffer[buffer_size];

    // Align the buffers
    // Note: Source buffer can be unaligned, but performance may be degraded for unaligned buffer
    // Destination buffer MUST be aligned, otherwise bad things may happen
    // Use posix_memalign / _aligned_malloc to allocate aligned buffer in real world
    unsigned char* src_buffer_ptr = (unsigned char*)(((uintptr_t)src_buffer + 15) & ~15);
    unsigned char* dst_buffer_ptr = (unsigned char*)(((uintptr_t)dst_buffer + 15) & ~15);

    // Y plane
    result = f3kdb_process_plane(core, 0, PLANE_Y, dst_buffer_ptr, FRAME_WIDTH, src_buffer_ptr, FRAME
    if (result != F3KDB_SUCCESS) {
        printf("f3kdb_process_plane code = %d\n", result);
```

```
        return result;
    }
    // Cb plane
    result = f3kdb_process_plane(core, 0, PLANE_CB, dst_buffer_ptr, FRAME_WIDTH, src_buffer_ptr, FRAM
    if (result != F3KDB_SUCCESS) {
        printf("f3kdb_process_plane code = %d\n", result);
        return result;
    }

    // Clean up
    f3kdb_destroy(core);
    return 0;
}
```

1.5 Change log

1.5.1 2.0.0

- Support Vapoursynth natively
- New parameter: preset
- C++ API support
- Dropped YUY2 support, please process in YV16 instead
- Dropped f3kdb_dither, please use `f3kdb(preset="depth", ...)` for bitdepth conversion
- Dropped several deprecated parameter values

1.5.2 1.5.1 (2012-04-07)

- Supports setting StdDev (sigma) for the Gaussian random number generator

1.5.3 1.5.0 (2012-03-12)

- (There isn't any new feature in this version, only some parameters are modified to reduce user confusion)
- ditherY/ditherC are renamed to grainY/grainC
- dynamic_dither_noise is renamed to dynamic_grain
- precision_mode is renamed to dither_algo, mode 4 and 5 are removed
- random_algo_dither is renamed to random_algo_grain
- enable_fast_skip_plane is removed, this optimization will be enabled implicitly whenever possible (Filter result won't be changed by this optimization)

1.5.4 1.4.2 (2011-11-10)

- Fixed crash on some non-mod16 videos

1.5.5 1.4.1 (2011-11-05)

- Fixed broken YUY2 support (still slow)
- Improved default value handling of bitdepth-related parameters
- precision_mode 4 / 5 are now deprecated and may be removed in future versions, you can use output_mode 1 / 2 to achieve the same result

1.5.6 1.4.0 (2011-10-30)

- 9 ~ 16 bit-depth input/output
  - Related parameters: input_mode/input_depth/output_mode/output_depth
- New random number generator, reference position and dither noise can be generated in uniform or gaussian distribution
  - Related parameters: random_algo_ref / random_algo_dither
- diff_seed is replaced with dynamic_dither_noise, when enabled, noise pattern will be different for each frame
- Another new parameter: enable_fast_skip_plane
- Short filter alias: f3kdb
- Now the ICC-compiled DLL should be runnable on pre-SSE2 systems (untested)
- Several bug fixes

1.5.7 1.3.0 (2011-09-07)

- Added x64 version
- Added a downsample filter: f3kdb_dither
- Internal precision is increased to 16bit
- New parameter: keep_tv_range, please see readme.txt for details
- Default sample_mode is changed to 2 as it is better in most cases
- Fixed: Floyd-Steinberg dithering may produce incorrect result for full-range videos
- Fixed: Broken YUY2 debanding
- Minor optimizations

1.5.8 1.2.0 (2011-08-01)

- Added support for YUY2 (not optimized yet)
- Added support for all planar YUV format in AviSynth 2.6
- The filter is now compatible with both AviSynth 2.5 and 2.6
- 16bit output (precision_mode = 4/5)
  - Note: The internal processing precision is still 14bit, this will be improved in future versions

1.5.9 1.1.0 (2011-06-18)

- Fixed a bug that high threshold values would produce incorrect result in high precision modes.
- Threshold values was scaled based on other parameter in previous versions, it is unscaled now to increase flexibility. Using same value has weaker effect in new version. Effect of default parameter set is also changed.
- SSE optimization for high precision mode.
- Rejects some invalid parameter combination instead of silently skips them

1.5.10 1.0.2 (2011-06-06)

- High precision mode
  - (currently non-optimized, SSE routine will be added later)
- Frame edges are properly processed now
- Fix crash in some cases (unaligned frames are handled correctly)
- Other bug fixes

1.5.11 1.0.1 (2011-05-27)

- Multi-threaded processing
- Skip planes which threshold is 0

1.5.12 1.0.0 (2011-05-21)

- Lots of bug fix
- SSE optimization, massive speed up

1.5.13 0.9.1 (2011-05-05)

- Fix: Incorrect results when blur_first=true

1.5.14 0.9 (2011-05-04)

- Initial release

CHAPTER 2

Web

http://forum.doom9.org/showthread.php?t=161411

http://www.nmm-hd.org/newbbs/viewtopic.php?f=7&t=239

CHAPTER 3

Acknowledgments

flash3kyuu

06_taro