VMAF integration into FFMPEG framework

There is no pre-built version of FFMPEG for Windows with libvmaf available online. Compiling ffmpeg under Windows is a little more complicated than Mac OS and Linux. Here's a link to a 32-bit Windows version of FFMPEG built with VMAF: https://drive.google.com/open?id=1HmQNAZhiXgwbZrpDzzdUXjNJMY9QeAN5

You may be able to use https://github.com/jb-alvarado/media-autobuild_suite to enable libvmaf in FFMPEG yourself. Instructions to build using the suite are:

Step 1: Download the media auto build suite from above link as a zip file and unzip folder. THis folder will be called the root folder in rest of this article.

Step 2: The root folder has a batch file called media-autobuild_suite.bat . Run it in command line as administrator and follow instructions to choose libraries you want enabled. Options chosen for the attached build are:

Feature option	Selection
Select the build target system: 1 = both [32 bit and 64 bit] 2 = 32 bit build system 3 = 64 bit build system	2
Build FFmpeg with which license? 1 = Non-free [unredistributable, but can include anything] 2 = GPLv3 [disables OpenSSL and FDK-AAC] 3 = GPLv2.1 [Same disables as GPLv3 with addition of gmp, opencore codecs] 4 = LGPLv3 [Disables x264, x265, XviD, GPL filters, etc. but reenables OpenSSL/FDK-AAC] 5 = LGPLv2.1 [same disables as LGPLv3 + GPLv2.1] If building for yourself, it's OK to choose non-free. If building to redistribute online, choose GPL or LGPL. If building to include in a GPLv2.1 binary, choose LGPLv2.1 or GPLv2.1. If you want to use FFmpeg together with closed source software, choose LGPL and follow instructions in https://www.ffmpeg.org/legal.html OpenSSL and FDK-AAC have licenses incompatible with GPL but compatible with LGPL, so they won't be disabled automatically if you choose LGPL.	2
Build standalone binaries for libraries included in FFmpeg? eg. Compile opusenc.exe if --enable-libopus 1 = Yes 2 = No	1
Build vpx [VP8/VP9 encoder]? 1 = Yes 2 = No Binaries being built depends on "standalone=y"	1
Build aom [Alliance for Open Media codec]? 1 = Yes 2 = No Binaries being built depends on "standalone=y"	1
Build rav1e [Alternative, faster AV1 standalone encoder]? 1 = Yes 2 = No	2
Build dav1d [Alternative, faster AV1 decoder]? 1 = Yes 2 = No	2

Build x264 [H.264 encoder]? 1 = Lib/binary with 8 and 10-bit 2 = No 3 = Lib/binary with only 10-bit 4 = Lib/binary with 8 and 10-bit, and libavformat and ffms2 5 = Shared lib/binary with 8 and 10-bit 6 = Same as 4 with video codecs only (can reduce size by ~3MB) 7 = Lib/binary with only 8-bit Binaries being built depends on "standalone=y" and are always static.	1
Build x265 [H.265 encoder]? 1 = Lib/binary with Main, Main10 and Main12 2 = No 3 = Lib/binary with Main10 only 4 = Lib/binary with Main only 5 = Lib/binary with Main, shared libs with Main10 and Main12 6 = Same as 1 with XP support and non-XP compatible x265-numa.exe 7 = Lib/binary with Main12 only Binaries being built depends on "standalone=y"	1
Build Kvazaar? [H.265 encoder] 1 = Yes 2 = No	2
Build SVT-HEVC? [H.265 encoder] 1 = Yes 2 = No	2
Build xvc? [HEVC and AV1 competitor] 1 = Yes 2 = No	2
Build Fraunhofer VVC? [H.265 successor enc/decoder] 1 = Yes 2 = No	2
Build SVT-AV1? [AV1 encoder] 1 = Yes 2 = No	2
Build SVT-VP9? [VP9 encoder] 1 = Yes 2 = No	2
Build FLAC? [Free Lossless Audio Codec] 1 = Yes 2 = No	1
Build FDK-AAC library and binary? [AAC-LC/HE/HEv2 codec] 1 = Yes 2 = No	1
Build FAAC library and binary? [old, low-quality and nonfree AAC-LC codec] 1 = Yes 2 = No	2
Build mediainfo binaries [Multimedia file information tool]? 1 = Yes 2 = No	2
Build sox binaries [Sound processing tool]? 1 = Yes 2 = No	2
Build FFmpeg binaries and libraries: 1 = Yes [static] [recommended] 2 = No 3 = Shared 4 = Both static and shared [shared goes to an isolated directory] 5 = Shared-only with some shared libs (libass, freetype and fribidi) 6 = Same as 4, but static compilation ignores shared dependencies	1

Always build FFmpeg when libraries have been updated? 1 = Yes 2 = No 3 = Only build FFmpeg/mpv and missing dependencies	1
Choose ffmpeg and mpv optional libraries? 1 = Yes 2 = No (Light build) 3 = No (Mimic Zeranoe) 4 = No (All available external libs)	Selected: 4, otherwise libvmaf will not be included
Build static mp4box [mp4 muxer/toolbox] binary? 1 = Yes 2 = No	2
Build static rtmpdump binaries [rtmp tools]? 1 = Yes 2 = No	2
Build static mplayer/mencoder binary? 1 = Yes 2 = No	2
Build mpv? 1 = Yes 2 = No	2
Build static bmx tools? 1 = Yes 2 = No	2
Build static curl? 1 = Yes (same backend as FFmpeg's) 2 = No 3 = SChannel backend 4 = GnuTLS backend 5 = OpenSSL backend 6 = LibreSSL backend 7 = mbedTLS backend	1
Build FFMedia Broadcast binary? 1 = Yes 2 = No	2
Build cyanrip (CLI CD ripper)? 1 = Yes 2 = No	2
Build redshift (f.lux FOSS clone)? 1 = Yes 2 = No	2
Build ripgrep (faster grep in Rust)? 1 = Yes 2 = No	2
Build jq (CLI JSON processor)? 1 = Yes 2 = No	2
Build jo (CLI JSON from shell)? 1 = Yes 2 = No	2
Build dssim (multiscale SSIM in Rust)? 1 = Yes 2 = No	2
Build avs2 (Audio Video Coding Standard Gen2 encoder/decoder)? 1 = Yes 2 = No	2

Number of CPU Cores/Threads for compiling: [it is non-recommended to use all cores/threads!] Recommended: 4	2
Delete versioned source folders after compile is done? 1 = Yes [recommended] 2 = No	1
Strip compiled files binaries? 1 = Yes [recommended] 2 = No	1
Pack compiled files? 1 = Yes 2 = No [recommended]	2
Write logs of compilation commands? 1 = Yes [recommended] 2 = No	1
Create script to update suite files automatically? 1 = Yes 2 = No	1
Show timestamps of commands during compilation? 1 = Yes 2 = No	1
Use ccache when compiling? Experimental. Speeds up rebuilds and recompilations, but requires the files to be compiled at least once before any effect is seen 1 = Yes 2 = No	2
Are you running this script through ssh or similar? (Can't open another window outside of this terminal) 1 = Yes 2 = No	2

Step: 3 If you encounter error messages due to Powershell related restrictions, you can enable all scripting by opening the Powershell command line and temporarily removing restrictions on running scripts.

Get-ExecutionPolicy;
Set-ExecutionPolicy unrestricted;

Ref: https://superuser.com/questions/106360/how-to-enable-execution-of-powershell-scripts

Close Powershell and re-run media-autobuild_suite.bat batch file in the root folder. If this runs successfully, there is a folder called "local32" in the root folder which has all the binary files.

ffmpeg.exe should have --enable-libvmaf when you run it in the command line. Usage is:

ffmpeg.exe -i distorted -i source -lavfi libvmaf="model_path=vmaf_v0.6.1.pkl:log_fmt=xml:psnr=1:ssim=1:log_path=metrics.txt" -f null -

Step: 4 If (like for me) libvmaf doesn't build but everything else has built, find Msys2.exe which is in the msys64 folder in the root and run it. You have to cd to the build folder where all the git folders are and use the following command lines: .

source /build/media-suite_helper.sh
do_vcs "https://github.com/Netflix/vmaf.git"
make clean
do_make INSTALL_PREFIX="$LOCALDESTDIR"

do_makeinstall INSTALL_PREFIX="$LOCALDESTDIR"

Note: At this stage I got a cflag error which I fixed by adding CFLAGS=-msse4.1 in the makefile of vmaf-git folder. Once this is done, re-run the

command lines from do_make INSTALL_PREFIX="$LOCALDESTDIR".

Ref: https://github.com/m-ab-s/media-autobuild_suite/blob/ddf92aeb0f71cf3299e803a1ea8fd2716a0b0061/build/media-suite_compile.sh#L1462-L1472

Step: 5: Must be followed if step 4 was used to configure and re-build ffmpeg.

Open MSys2.exe and cd to the ffmpeg-git folder that sits inside the root's build folder. Configure ffmpeg using the following command line.

./configure --prefix=/local32 --bindir=/local32/bin-video --pkg-config-flags=--static --disable-autodetect --enable-amf --enable-bzlib --enable-cuda --enable-cuvid --enable-d3d11va --enable-dxva2 --enable-iconv --enable-lzma --enable-nvenc --enable-zlib --enable-sdl2 --enable-ffnvcodec --enable-nvdec --enable-cuda-llvm --enable-libmp3lame --enable-libopus --enable-libvorbis --enable-libvpx --enable-libx264 --enable-libx265 --enable-libdav1d --disable-debug --enable-fontconfig --enable-libass --enable-libbluray --enable-libfreetype --enable-libmfx --enable-libmysofa --enable-libopencore-amrnb --enable-libopencore-amrwb --enable-libsnappy --enable-libsoxr --enable-libspeex --enable-libtheora --enable-libtwolame --enable-libvidstab --enable-libvo-amrwbenc --enable-libwavpack --enable-libwebp --enable-libxml2 --enable-libzimg --enable-libshine --enable-gpl --enable-avisynth --enable-libxvid --enable-libaom --enable-libopenmpt --enable-version3 --enable-chromaprint --enable-frei0r --enable-libbs2b --enable-libcaca --enable-libcdio --enable-libflite --enable-libfribidi --enable-libgme --enable-libgsm --enable-libilbc --enable-libkvazaar --enable-libmodplug --enable-librtmp --enable-librubberband --enable-libssh --enable-libtesseract --enable-libxavs --enable-libzmq --enable-libzvbi --enable-openal --enable-libvmaf --enable-libcodec2 --enable-libsrt --enable-ladspa --enable-opencl --enable-opengl --enable-mbedtls --extra-cflags=-fopenmp --extra-libs=-lgomp --extra-cflags=-DLIBTWOLAME_STATIC --extra-libs=-lstdc++ --extra-cflags=-DLIBSSH_STATIC --extra-ldflags=-Wl,--allow-multiple-definition --extra-cflags=-DCACA_STATIC --extra-cflags=-DMODPLUG_STATIC --extra-cflags=-DCHROMAPRINT_NODLL --extra-libs=-lstdc++ --extra-cflags=-DZMQ_STATIC --extra-libs=-lpsapi --extra-cflags=-DLIBXML_STATIC --extra-libs=-liconv --disable-w32threads --extra-cflags=-DKVZ_STATIC_LIB --extra-version=gc54268ce02+1

Once configure is complete run *make install*. This will update the ffmpeg.exe file in the root's local32 folder. fmpeg.exe should have --enable-libvmaf when you run it in the command line. Usage is:

ffmpeg.exe -i distorted -i source -lavfi libvmaf="model_path=vmaf_v0.6.1.pkl:log_fmt=xml:psnr=1:ssim=1:log_path=metrics.txt" -f null -

NOTE: VMAF's pkl and pkl.model files are required to run VMAF using FFMPEG. These files can be found in root/build/vmaf-git/model folder. Copy them into the same folder where ffmpeg.exe file is present.