HAVC_cmnet2dit — CMNET2 Colorization with DiT Reference Frames

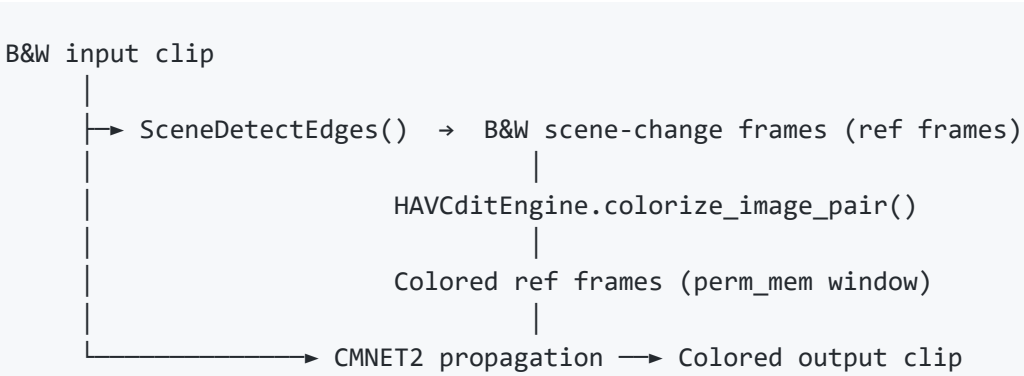
Overview

`HAVC_cmnet2dit()` is a self-contained colorization filter that combines two complementary models:

- **HAVCditEngine** — a DiT-based model (Nunchaku/FLUX quantized) used to colorize **B&W scene-change frames** that will act as reference images.
- **CMNET2** — a propagation model that uses those colorized reference frames as anchors and propagates color across the full clip via a sliding permanent-memory window.

The key difference from `HAVC_cmnet2()` is that **no pre-colored reference clip is needed**. The function detects scene changes in the B&W input clip itself, colorizes those frames with the DiT engine, and feeds them into CMNET2 as reference anchors. Everything is handled internally.

Architecture



Reference frames are colorized **in pairs** (`colorize_image_pair()`) before being loaded into the CMNET2 permanent-memory window. This halves the number of DiT forward passes compared to colorizing one frame at a time. When only one B&W reference frame remains at the end of the clip, `colorize_image()` is used as a fallback. The permanent-memory window size is always even, which is a requirement of the pair-wise colorization scheme.

Requirements

DiTServerRPC (mandatory prerequisite)

`HAVC_cmnet2dit()` requires an external RPC server to be **installed and running** before the VapourSynth script is executed. The server loads the DiT model into GPU memory and exposes the colorization API consumed by `HAVCditEngine`.

The server is maintained as a separate project:

> **https://github.com/dan64/DiTServerRPC**

Refer to that repository for installation instructions, model download procedure, and server startup options. Once the server is running and listening on the configured `host` / `port` (default: `127.0.0.1:8765`), `HAVC_cmnet2dit()` will connect to it automatically on the first colorization call. If the model is already loaded on the server (e.g. from a previous session or from the HAVC GUI), the pipeline load step is skipped and the first call returns immediately.

> **Note:** The VapourSynth process and the DiTServerRPC server can run on the same machine (recommended, using shared memory for zero-copy image transfer) or on separate machines connected over a local network (using the `host` and `port` keys in `dit_engine_params`).

Other requirements

- CUDA GPU with sufficient VRAM (tested on RTX 50-Series with fp4 precision, and on RTX 30/40-Series with int4 precision).
- CMNET2 model weights accessible at the HAVC package model directory (or at the path specified by `torch_dir`).

Basic Usage

```
import vsdeoldify as havc

clip_bw = havc.HAVC_read_video(source="video_bw.mp4")

clip = havc.HAVC_cmnet2dit(clip_bw)
# retry_model=1 (default) — DiT fp4 used for retry if needed
```

The defaults are safe for most clips. No reference clip, no external frame directory, no manual configuration required.

Full Script Example

```
import vapoursynth as vs
import sys, os

core = vs.core
core.max_cache_size = 16384

import vsdeoldify as havc

clip_bw = havc.HAVC_read_video(source="video_bw.mp4")

clip = havc.HAVC_cmnet2dit(
    clip_bw,
    encode_mode        = 0,          # remote CMNET2 backend (recommended)
    render_speed       = "auto",     # automatic resolution scaling
    max_memory_frames  = 20,         # sliding window: 20 ref frames (10 pairs)
    sc_min_freq           = 25,         # force a ref frame every 25 frames
    render_vivid       = False,      # no saturation boost
    retry_model        = 1,           # DiT fp4 (default) for retry
    retry_threshold    = 0.25,        # trigger retry when coverage < 25%
)

# Convert to YUV420P10 for x265 encoding
clip = core.resize.Bicubic(
    clip        = clip,
    format      = vs.YUV420P10,
    matrix_s    = "709",
    range_in_s  = "full",
    range_s     = "limited",
    dither_type = "error_diffusion",
)

clip.set_output()
```

Parameters

`clip` — `vs.VideoNode`

B&W source clip. Any VapourSynth format is accepted; it is converted internally to RGB24 before processing and restored to the original format on output.

`render_speed` — `str`, default `"auto"`

Controls the resolution at which CMNET2 runs inference. Larger resolution means more accurate colors but slower processing.

Value	Max long side	Notes
`"auto"`	adaptive	`"fast"` for portrait (ratio < 1.6), `"medium"` for landscape
`"fast"`	384 px	Fastest; colors may be washed out
`"medium"`	512 px	Good balance (default for most landscape clips)
`"slow"`	640 px	More vivid; noticeably slower
`"slower"`	original	Full resolution; use only for short clips

The clip is scaled down to the inference resolution, colorized, then scaled back to the original resolution. The original luma is grafted back after scaling to preserve sharpness.

`render_vivid` — `bool`, default `False`

When `True`, applies a gentle hue (+3°) and saturation (+15%) boost to the colorized output. Useful when DiT colors appear slightly muted. Implemented via `vs_tweak()`.

`sc_thresh` — `float`, default `0.035`

Scene edges-detection threshold in the range [0.01, 0.15]. Lower values detect more scene changes (more reference frames, more DiT calls at preload time); higher values detect fewer.

Guidance:

- `0.02–0.05` — standard films with clear cuts.
- `0.05–0.10` — clips with dissolves or moderate changes.
- If too few reference frames are generated, `HAVC_cmnet2dit` will raise an exception at startup. Lower `sc_thresh` or increase `sc_min_freq` to fix this.

`sc_tht_ssim` — `float`, default `0.80`

Threshold used by the SSIM (Structural Similarity Index Metric) selection filter. If > 0, activates a post-scene-detection filter that discards scene-change frames that are visually similar to already-selected reference frames. This improves the quality of the reference set by avoiding redundant frames.

Suggested values: 0.35–0.85. Default `0.80`. Set to `0.0` to disable.

`sc_min_int` — `int`, default `25`

Minimum number of frames that must separate two consecutive scene-change detections. Range [1, 25].

Prevents the scene detector from generating reference frames too close together, which would waste DiT inference time on near-duplicate frames.

`sc_tht_offset` — `int`, default `2`

Frame offset used for scene-change comparison. The detector compares `frame[n]` against `frame[n - offset]`. An offset > 1 helps detect blended scene transitions (dissolves, cross-fades) that a single-frame difference would miss. Range [1, 25].

`sc_min_freq` — `int`, default `0`

If greater than zero, guarantees at least one reference frame every `sc_min_freq` video frames, regardless of scene changes. Range [0, 1500].

Useful for long scenes without cuts where CMNET2 would otherwise propagate color from a single reference for many seconds, potentially drifting.

Example: `sc_min_freq=25` forces a new reference frame every 25 frames (~1 second at 25 fps), producing richer temporal anchoring without relying solely on scene-change detection.

`encode_mode` — `int`, default `0`

Selects the CMNET2 processing backend.

Value	Backend	Description
`0`	Remote (XML-RPC subprocess)	Recommended. CMNET2 runs in a child process; no GPU memory contention with the DiT engine. Fully exploits the sliding permanent-memory window.
`1`	Local (in-process)	CMNET2 runs in the same process. Useful when subprocess spawning is not available. May be slower due to GPU memory sharing with the DiT engine.

`max_memory_frames` — `int`, default `0` (→ 20)

Size of the CMNET2 sliding permanent-memory window, i.e. how many colorized reference frames are kept in CMNET2 long-term memory at any given time. `0` defaults to `DEF_XRF_WINDOW_SIZE = 20`.

This value is **always rounded down to the nearest even number** internally, because reference frames are colorized in pairs by `colorize_image_pair()`.

Effect on quality and speed:

- Larger window → CMNET2 has more color context → better temporal consistency, especially across scene changes.
- Larger window → more DiT calls at startup (all pairs must be colorized before the first output frame) → longer initial delay.
- The window slides by 2 frames at a time (one DiT pair call per slide). The slide is triggered roughly at the mid-point of the current window.

Suggested values:

Value	Use case
10	Fast preview; minimal startup delay
20	Default; good balance (10 DiT pair calls at startup)
40–60	Long complex films; more stable color propagation
100+	Very long single-scene sequences; significant startup delay

---

`dit_engine_params` — `dict`, default `None`

Optional dictionary of keyword arguments forwarded to `HAVCditEngine`. Only the keys you specify override the defaults; unspecified keys retain their default values.

Default values:

Key	Default	Description
`host`	`"127.0.0.1"`	RPC server IP address
`port`	`8765`	RPC server TCP port
`model_name`	`"nunchaku-qwen"`	Nunchaku model family
`model_precision`	`"fp4"`	Quantization precision (see below)
`model_rank`	`"32"`	SVD rank: `"32"` (faster) or `"128"` (more detail)
`model_inference_steps`	`"4"`	Steps used to identify the model file to download
`cache_dir`	`""`	HuggingFace cache dir (empty = `~/.cache/huggingface`)
`full_model_path`	`""`	Absolute path to a local `.safetensors` file
`prompt`	`"Colorize this image, natural colors..."`	Text prompt guiding the DiT colorization
`steps`	`2`	Inference steps per image at runtime
`img_size`	`0`	Max long side before inference (0 = no resize)

`model_precision`

This is the most common parameter to override. The correct value depends on your GPU generation:

Value	Required GPU	Notes
`"fp4"`	NVIDIA RTX 50-Series (Blackwell)	Default. Fastest on supported hardware.
`"int4"`	NVIDIA RTX 30-Series or 40-Series	Required for pre-Blackwell GPUs.

Example — switching to int4 for an RTX 3090 or 4090:

```
clip = havc.HAVC_cmnet2dit(
    clip_bw,
    dit_engine_params = {"model_precision": "int4"},
)
```

`model_rank`

Controls the SVD rank of the Nunchaku model. Higher rank captures more detail at the cost of increased VRAM and compute.

```
# Higher detail, slower colorization
clip = havc.HAVC_cmnet2dit(
    clip_bw,
    dit_engine_params = {"model_rank": "128"},
)
```

`full_model_path`

When set, bypasses model_name / model_precision / model_rank / model_inference_steps and loads the specified `.safetensors` file directly. Useful for local model files or custom fine-tunes.

```
clip = havc.HAVC_cmnet2dit(
    clip_bw,
    dit_engine_params = {
        "full_model_path": "D:/models/svdq-fp4_r32-qwen-image-edit-2509-lightning-
4steps-251115.safetensors",
    },
)
```

`prompt`

The text prompt is forwarded to the DiT model to guide colorization style. The default prompt emphasizes natural colors and structure preservation. Modify it to steer the style for specific types of footage.

```
# Warmer, vintage look for old newsreel footage
clip = havc.HAVC_cmnet2dit(
    clip_bw,
    dit_engine_params = {
        "prompt": (
            "Colorize this historical photograph with warm, vintage tones. "
            "Strictly preserve all shapes, edges and background details."
        ),
    },
)
```

`steps`

Number of DiT denoising steps at inference time. The minimum effective value for Nunchaku flow-matching models is 2 (the default); setting `steps=1` does not reduce inference time on this model family due to fixed CUDA kernel overhead.

`retry_threshold` — `float`, default `0.0`

Threshold used to identify frames that may benefit from an additional reference frame during colorization. When a frame's permanent-memory coverage is below this threshold, CMNET2 triggers a retry: the frame is re-colorized using the engine selected by `retry_model` and injected as a new reference.

Range [0.0, 1.0]. Default `0.0` disables the retry mechanism entirely. Suggested values: 0.20–0.35.

`retry_model` — `int`, default `1`

Selects the colorization engine used by the retry path (only active when `retry_threshold > 0`).

Value	Engine	Requires
`0`	HAVC (DeOldify + DDColor)	Nothing extra
`1`	DiT fp4	DiTServerRPC, RTX 50-Series (**default**)
`2`	DiT int4	DiTServerRPC, RTX 30/40-Series

If the selected DiT server is not reachable, the engine automatically falls back to model `0` (HAVC).

> **Conflict with `dit_engine_params`**: if `retry_model` and `dit_engine_params["model_precision"]` are in conflict (e.g. `retry_model=1` but `model_precision="int4"`), the value in `dit_engine_params` takes priority. The DiT server singleton is initialized from the dict first during the reference-frame preload, and `retry_model` cannot override an already-loaded model.

`torch_dir` — `str`, default: HAVC package model directory

Torch hub directory for CMNET2 model weights. Pass `None` to use the default Torch cache (`~/.cache/torch/hub`).

Advanced Usage Examples

RTX 40-Series GPU (int4 precision)

```
clip = havc.HAVC_cmnet2dit(
    clip_bw,
    encode_mode       = 0,
    render_speed      = "medium",
    max_memory_frames = 20,
    sc_min_freq         = 25,
    dit_engine_params = {
        "model_precision": "int4",
    },
)
```

Local model file + custom prompt

```
clip = havc.HAVC_cmnet2dit(
    clip_bw,
    encode_mode       = 0,
    render_speed      = "slow",
    max_memory_frames = 40,
    sc_thresh        = 0.08,
    dit_engine_params = {
        "full_model_path" : "D:/models/svdq-fp4_r32-qwen-image-edit-2509-lightning-
4steps-251115.safetensors",
        "prompt"          : "Colorize with warm, natural tones. Preserve all
details.",
    },
)
```

Remote DiT server on a second machine

```
clip = havc.HAVC_cmnet2dit(
    clip_bw,
    encode_mode       = 0,
    render_speed      = "auto",
    max_memory_frames = 20,
    dit_engine_params = {
        "host": "192.168.1.50",
        "port": 8765,
    },
)
```

High-quality pass for a short clip

```
clip = havc.HAVC_cmnet2dit(
    clip_bw,
    encode_mode       = 1,           # local backend (short clip, less overhead)
    render_speed      = "slower",    # full resolution inference
    max_memory_frames = 60,          # large window for maximum temporal consistency
    sc_min_freq          = 10,          # dense reference frames
    render_vivid      = True,        # slight saturation boost
    dit_engine_params = {
        "model_rank": "128",
    },
)
```

Performance Notes

- Typical throughput is ~4 fps on an RTX 50-Series GPU with `render_speed="auto"`, `max_memory_frames=20`, `encode_mode=0`.
- Setting `steps=1` does not increase speed on Nunchaku flow-matching models (2 is the practical minimum).
- The main cost drivers are: DiT inference during window preload and slide, and CMNET2 propagation frame-by-frame. Increasing `max_memory_frames` reduces the frequency of DiT slide calls at the cost of a longer startup.
- When `host="127.0.0.1"` or `"localhost"`, image data is transferred via shared memory (zero-copy), which is ~23% faster than the base64 RPC path used for remote hosts.
- `encode_mode=0` (remote CMNET2) is recommended over `encode_mode=1` (local) because it avoids GPU memory contention between the DiT engine and CMNET2.

Differences from `HAVC_cmnet2()`

Feature	`HAVC_cmnet2`	`HAVC_cmnet2dit`
Reference frames	Pre-colored (caller provides `clip_ref`)	B&W (derived internally from input clip)
Reference colorization	n/a	`HAVCditEngine.colorize_image_pair()`
`clip_ref` parameter	Required	Not exposed (set to input clip internally)
`sc_framedir`	Supported	Not supported
`colormap` / `dark` / `smooth` filters	Supported	Not supported
`ref_merge`	Supported	Not supported
Window size	Any positive integer	Always rounded to nearest even number
Sliding step	Configurable (percentage-based)	Fixed: 2 frames per slide
`dit_engine_params`	Not present	Required to configure DiT model