

Film Grain Engine

A ComfyUI build guide. Real film grain for video, with the controls a colorist wants.

Workflow case study CS2 · build it yourself · v3 · 2026-06-06

This guide gives you two ways to the same result. **Track 1** assembles the engine from stock nodes, no code. **Track 2** collapses it into one custom node you build with Claude Code, which is the version that doubles as a hire signal. Build Track 1 first so you have a working result and you understand the moving parts, then graduate to Track 2 for the clean artifact.

The point of this workflow

Anyone can drop a grain filter on a clip. Almost nobody grains the way 35mm actually behaves: grain in all three emulsion layers with the blue layer grainiest, crisp fine clumps rather than soft blur, heaviest in the midtones but never gone from the shadows or highlights, added to the image rather than overlaid, and alive frame to frame without boiling. That physics is the whole case study. The result has to read like film, not like noise.

What you need: ComfyUI installed and updated, ComfyUI Manager, and a short test clip (a few seconds of flat, ungraded footage). Time: about an hour for Track 1.

Step 0 · The nodes you need

Track 2 needs almost nothing. Track 1 needs two small packs plus nodes that already ship with ComfyUI. Install through ComfyUI Manager, then restart.

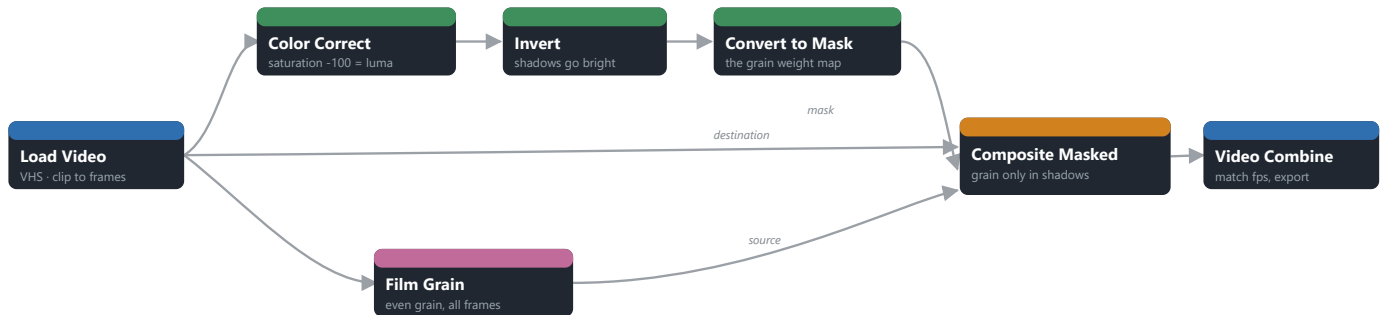
- **ComfyUI-VideoHelperSuite (VHS)** — load a clip as frames and save frames back to video. Both tracks use it. Nodes: `Load Video`, `Video Combine`.
- **ComfyUI-post-processing-nodes** — the grain source and clean image math. Nodes: `Film Grain`, `Color Correct`, `Blend`. Track 1 only.
- **Core ComfyUI** — nothing to install. `Invert Image Colors`, `Convert Image to Mask`, and `Image Composite Masked` are built in. Track 1 only.
- **The Colorist Film Grain node** — you build this in Track 2. It is torch only, so it loads with a restart and needs no other pack.

One thing that will save you an hour

WAS Node Suite has grain, levels, and blend nodes too, and they look like the obvious choice. Skip them for video. Most WAS image nodes process a single frame and throw an error the moment you hand them a clip, because they convert each tensor to a PIL image and a video batch is not a single image. Everything below stays on VideoHelperSuite, post-processing, and core nodes, which all run across the whole batch.

Track 1 · Build it from stock nodes (no code)

Build left to right. The graph has two branches off the same frames: one builds a weight map that says where grain belongs, the other makes the grain, and a masked composite marries them. Test on a single short clip before you commit.



The full Track 1 graph. Blue is video in and out, green is the weight branch, pink is the raw grain, amber is the masked composite that puts grain where it belongs.

Stage 1 · Load the clip as frames

- Add **Load Video** (`VHS_LoadVideo`) and point it at your clip. It outputs an IMAGE batch (your frames) and the frame rate.
- Note the frame rate. You will set the same number on save. Matching fps is not optional.

Stage 2 · Make the luminance map

- Add **Color Correct** (post-processing) and feed it the frames. Set `saturation` to **-100**. That gives you a clean grayscale, which is your luminance: bright pixels are highlights, dark pixels are shadows.

Stage 3 · Flip it so shadows are bright

- Add **Invert Image Colors** (core). Now the shadows are white and the highlights are black. White is where you want grain.
- Optional balance control: drop another **Color Correct** after the invert and raise `contrast` or lower `gamma` to make the falloff steeper. That is your shadow and highlight grain balance, the dial no filter gives you.

Stage 4 · Turn the map into a mask

- Add **Convert Image to Mask** (core), channel `red`. This is your grain weight map: strong in the shadows and mids, fading to nothing in the highlights.

Stage 5 · Make the grain

- Add **Film Grain** (post-processing) and feed it the original frames. Start around `intensity 0.5`, `scale 8`. This grains every frame evenly, which is exactly what you do not want on its own. Stage 6 fixes that.

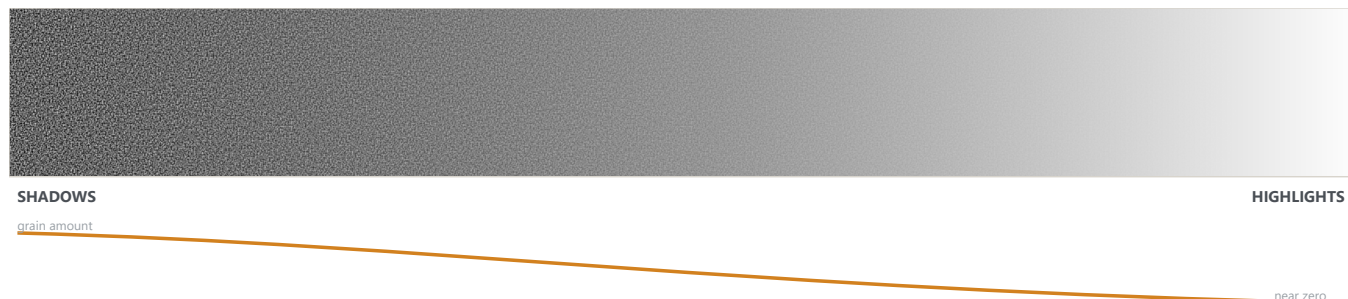
Stage 6 · Put the grain only where it belongs

- Add **Image Composite Masked** (core). Set `destination` to the original frames, `source` to the grained frames, and `mask` to the weight map from Stage 4.

- Now the grained version shows through in the shadows and mids and the clean original shows through in the highlights. That is the whole craft move in one node.

Stage 7 · Grain goes last, then save

- If you grade inside this graph, grade first and grain after, never the reverse, or the grade crushes and amplifies the grain wrong.
- Add **Video Combine** (`VHS_VideoCombine`), feed the composited frames, set fps to the source fps from Stage 1, and export a high quality codec (ProRes, or high bitrate h264 for a quick demo).



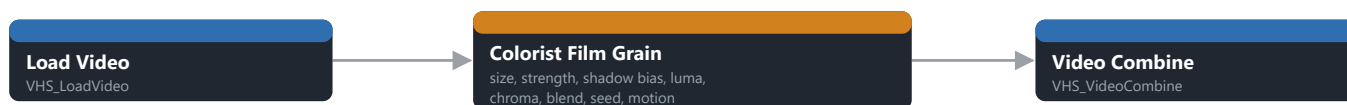
What the weight map does. Grain is dense in the shadows and mids and thins to almost nothing in the highlights, the way real stock behaves. Match that and the eye believes it.

Track 2 · One custom node, built with Claude Code

Track 1 works, but it is a sprawl of seven nodes. The stronger artifact is a single **Colorist Film Grain** node with every control on one panel. You do not hand write it. You build it with Claude Code, which is the AI native working style the bridge seat tech roles now assume. Building it this way is a credential, not a shortcut.

What makes it read as 35mm, not as noise

All of it is physics. Grain in three independent channels, with the blue layer grainiest and its clumps larger, so it carries the faint color shimmer of real stock. Crisp band-pass clumps: full resolution noise shaped with a difference of Gaussians so the large-scale blotch is gone and only the fine even sparkle remains. Amplitude that peaks in the midtones and never fully clears the shadows or highlights, the curve real film follows. And grain added to the image, not laid over it. Miss any one and the eye calls it digital.



Track 2: the same result in three nodes. This clean graph is the screenshot that sells the case study.

The controls, on one panel

Control	Range	What it does
grain_size	0.5 to 4.0	Clump size. Smaller is fine 35mm, bigger is coarser, older stock.
strength	0 to 1	Overall grain amount.
shadow_bias	0 to 1	Tonal placement. 0 is the physical midtone peak, higher pushes grain into the shadows.
luma_amt	0 to 1	The achromatic, shared part of the grain.
chroma_amt	0 to 1	The chromatic, per channel part. Color film has this; set 0 for B&W stock.
blend_mode	add, overlay, soft_light	add is film true. The others are stylistic.
seed	integer	Repeatable grain. Same seed, same field.
grain_motion	0 to 1	Temporal life. 0 is frozen, 1 is fresh every frame, the middle is alive but calm.

Making the grain alive (and where the code earns it)

Grain has to move, and how it moves is the difference between film and a mistake. Freeze one plate across the clip and it reads as a dead texture stuck to the lens. Real 35mm is the opposite: every frame carries its own fresh, independent layer of grain. Because the grain here is fine, fresh every frame is the authentic look and it does not boil. Boil comes from grain that is too coarse, not from grain that is fresh.

grain_motion is that control. 0 is a frozen plate, 1 is fresh every frame, which is how film behaves and the right default. The band in between is for one job: if you run coarse grain that starts to crawl, slowing it adds coherence so it drifts instead of boiling. Under the hood it crossfades random keyframes with a constant power blend, so even the slowed setting never pulses.

Here is where the code actually earns its place, and it is not the motion. Both stock Film Grain and this node can give you fresh grain every frame. What stock nodes cannot assemble on a video batch is the texture that makes it read as 35mm: grain in three independent channels with the blue layer grainiest, crisp clumps instead of blur, amplitude that peaks in the midtones and never clears the frame, all added rather than overlaid. That is a real piece of engineering by hand, and it became one node built with Claude Code. The tool did not do the work. It cleared a wall you would otherwise have walked away from.



grain_motion. Frozen reads as a dead plate; fresh every frame is how 35mm behaves and the default here. The band between adds coherence for coarse grain that would otherwise crawl, with a constant power crossfade so it never pulses.

The Claude Code kickoff prompt

Paste this into Claude Code in a fresh folder. It builds the whole 35mm grain model as a single node.

Build a ComfyUI custom node called "Colorist Film Grain".

Structure: a standard custom_nodes package. `__init__.py` re-exports `NODE_CLASS_MAPPINGS` and `NODE_DISPLAY_NAME_MAPPINGS` from the node module.

The node class needs:

```
CATEGORY = "image/postprocessing"
FUNCTION = "apply_grain"
RETURN_TYPES = ("IMAGE",)          # trailing comma matters
RETURN_NAMES = ("image",)
INPUT_TYPES as a @classmethod returning {"required": {...}} with a
default/min/max/step on every FLOAT and INT widget:
    image (IMAGE), grain_size (FLOAT 0.5-4.0), strength (FLOAT 0-1),
    shadow_bias (FLOAT 0-1), luma_amt (FLOAT 0-1), chroma_amt (FLOAT 0-1),
    blend_mode (["add", "overlay", "soft_light"]), seed (INT),
    grain_motion (FLOAT 0-1)
```

Behavior, torch only, on a batch of frames [B,H,W,C] float 0-1.

Model it like 35mm color negative, per channel:

1. Make two temporal noise fields at full resolution, one shared (1 channel, achromatic) and one independent (3 channels, one per layer). Each is a few random keyframes crossfaded across the batch with constant-power (cos/sin) weights so grain drifts without its strength pulsing. `grain_motion` sets the keyframe count: 0 frozen, 1 fresh per frame, middle alive. Use a `torch.Generator` seeded by `seed`, never the global `torch.manual_seed`.
2. `grain = luma_amt*shared + chroma_amt*channel_weight*independent`, with channel weights making blue grainiest (about 0.85, 1.0, 1.35 for R, G, B).
3. Shape the clumps with a band-pass (difference of Gaussians): blur at sigma about $0.18 + 0.35 * \text{grain_size}$ (fine, for 35mm; raise it for coarser 16mm), then subtract about 0.35 of a blur 2.5x broader so the large-scale blotch is removed. Less of that subtraction is smoother (35mm), more is crunchier (16mm). Blur the blue channel a little larger and the red a little finer (about 0.85, 1.0, 1.3); keep the shared luminance grain at one uniform size so it stays achromatic. Normalise to unit variance so size does not change the amount.
4. Per-channel signal-dependent amplitude: $a = \text{floor} + (1 - \text{floor}) * 2 * \sqrt{I * (1 - I)}$, floor about 0.2, so grain peaks in the midtones and never clears the shadows or highlights. `shadow_bias` tilts a toward the shadows.
5. add: `out = image + grain * base_sigma * strength * a`, `base_sigma` about 0.12. overlay and `soft_light`: center grain on 0.5 and use the Pegtop formula instead. clamp 0-1.

Then tell me where to drop the folder and how to restart ComfyUI.

- Drop the folder in `ComfyUI/custom_nodes/`, restart, and the node appears under `image/postprocessing`.
- When something breaks, paste the error back into Claude Code. That read and fix loop is the real skill, and it is the honest version of how you work.
- Your whole graph is now three nodes: Load Video, Colorist Film Grain, Video Combine.

Quality check before you call it done

- Zoom to 100 percent. Grain should sit across the whole frame, strongest in the midtones and lighter, but never gone, in the deep shadows and bright highlights. If it only shows in the shadows, the amplitude curve is off.
- Play it. If the grain crawls or boils, lower the strength for the demo and note it (see the limitation below).
- Compare against the flat source side by side. The grained version should feel like it was shot, not processed.

- Skin and clean gradients are the truth test. That is where fake grain shows first.

Deliver it without losing the grain

Fine grain is the hardest thing for a codec to keep, and the wrong export hands you back the soft look you started with. Master to ProRes or a 10-bit high bitrate file. For h264, push the quality right up (low crf, around 12), or the encoder spends its bits smearing the grain. Social platforms re-encode again on upload, so always send the highest quality you can and let them compress it, not you.

Companion: halation for highlight scenes

For shots with practical lights or hot highlights, run a Colorist Halation node before the grain. It blooms the warm red-orange glow real film throws around highlights, done in linear light, localised to the highlights, screened back. The demo's lamplit interior uses it. It is a separate node so the grain stays one clean job, and the order is the film order: halation first (image formation), grain last.

The honest notes (keep them in)

Three layers, three channels

Real color negative has three dye layers whose grain overlaps across the spectrum. This models that as three independent noise channels with the blue layer grainiest. A full model would account for the spectral overlap between layers; three independent channels is the practical version, and it reads right. Worth knowing, not worth hiding.

Living grain costs memory on long clips

A moving grain field means a unique plate per frame, which uses more memory than a single frozen plate. On a long or high resolution timeline, render in batches (VideoHelperSuite has a batch manager) or pull grain_motion toward frozen. Naming the real tradeoff is what makes the whole thing read as a practitioner instead of marketing.

Capture as you build (so the post writes itself)

While you build, drop these in [02_Demo_Assets/](#) : the flat source clip and the grained clip, a split screen with flat on the left and graded and grained on the right (the scroll stopper), two or three clean grabs of the node graph (the three node version is the best one), and three quick notes in your own words: the problem in one studio person sentence, the order you grouped the graph, and every moment you thought this should be easier. That last list becomes the honest limitation beat.

Build sheet and full context: [09_Workflows/02_Film_Grain_Engine/00_Docs/BUILD_SHEET.md](#) . This is a public teaching workflow. Naming ComfyUI and Claude Code here is correct. The film recipes never appear.