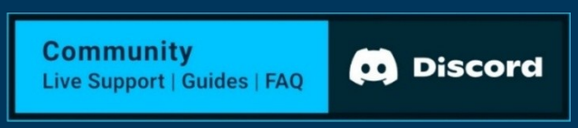
MotionCam PRO
RAW VIDEO CAPTURE

FULL USER GUIDE

4.0.5-Pro Release

Manual Version: 1.0.0

https://discord.gg/Vy4gQNEdNS



https://www.motioncamapp.com/

Contact Us: motioncam.app@gmail.com

[0.1] USER INTERFACE BASICS (QUICK START 1/2)

DIRECT PREVIEW MODE: Tap toggle DP mode On/Off. You may also hold Vol + button to trigger

HISTOGRAM QUICK TOGGLE: Tap to toggle Histogram On/Off

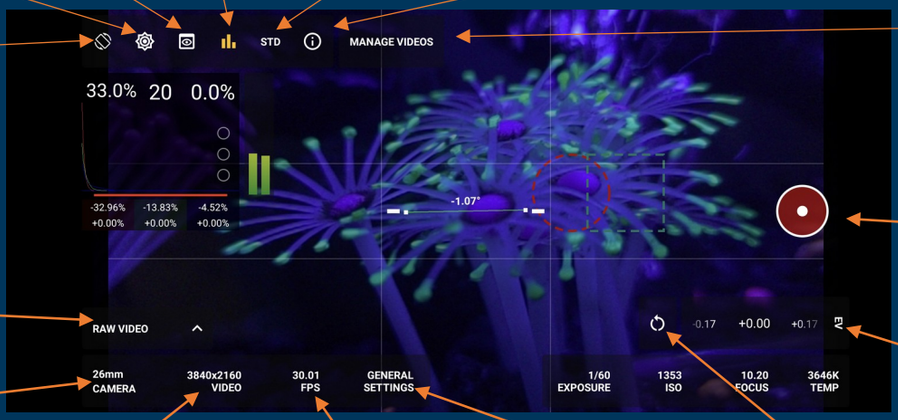
VIEWFINDER RESOLUTION MODE: Press to alternate the preview resolution from STD = Standard (Current selection) to MAX = Maximum, useful for visual granular focusing.

ABOUT: Displays information about MotionCam, including app resources and contact us info

BRIGHTNESS: Tap to toggle (Full vs System setting)

MANAGE VIDEOS: Opens the MCRAW (MotionCam RAWs) file manager. Permits batch handling and provides access to in-app CinemaDNG exporter (Export DNG) as well as single image extractor (Edit Photo) and Video Renderer (Edit Video) modes

ORIENTATION: Tap to rotate, hold to unlock



RECORD BUTTON: Tap to Start/Stop recording. Hold to record with buffer mode

MODE SELECTION: Displays current mode. Tap to open list of modes

EXPOSURE COMPENSATION VALUE: Slide left or right to decrease or increase the exposure compensation amount.

CAMERA & VIEWFINDER SETTINGS: Opens list of device lens IDs and allows for viewfinder adjustments. Swipe left or right to quickly change lenses

CAPTURE SETTINGS: Allows adjustment of capture settings such as resolution crop/binning, RAW stream configuration, storage designation, and more

FRAMERATE: Opens FPS selection (Note: Unsupported values may still work)

GENERAL SETTINGS: Opens General Settings menu options (Histogram, Grid lines, preferences, etc.)

RESET CONTROLS: Press to reset all exposure, metering, focus and white balance settings back to full auto mode

USER INTERFACE BASICS (QUICK START 2/2)

LUMINESCENCE: Displays the overall amount of light hitting the camera sensor as a numerical scale from 0 (pure black) to 1000 (pure white). Great to assess the sensor light intake and scene conditions.

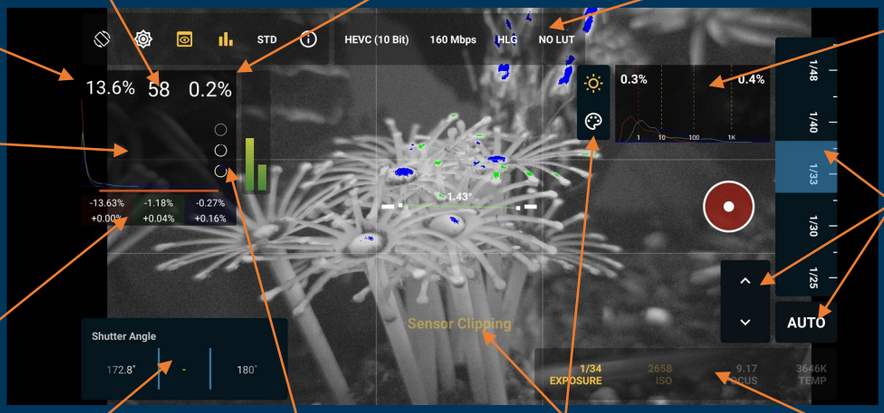
CRUSHING PERCENTAGE: Indicates percentage of total pixels reading fully black (shadow data loss)

CLIPPING PERCENTAGE: Indicates percentage of total pixels reading fully white (Highlight data loss)

DIRECT LOG SETTINGS: Shows current Direct Log mode capture settings. Tap any to open menu or hold LUT to toggle LUT chosen on/off

ENCODER HISTOGRAM: Provides histogram data specific for the selected Direct Log settings as they will appear after encoding. Shows crushing and clipping percentages

RAW HISTOGRAM: Tap to expand. Shows sensor RAW data histogram (Not JPEG). Provides precise measurements of light captured. Left = Crushed Shadows. Right = Clipped Light. Vertical bars indicate audio input (Higher bar indicates louder).



CONTROL SLIDER AND TOGGLES: Provides slider for granular controls and arrows for jumping between settings. Press AUTO to undo manual setting. Adapts based on Exposure, ISO, Focus or White Balance/Temperature

INDIVIDUAL CHANNEL CLPPING/CRUSHING: Indicates individual color channels (Red, Green and Blue) and their percentages clipped (positive) and crushed (negative). Ideal for highlight recovery gauging

SPECIAL CONTROL MENU: Provides options for Shutter angles, ISO quick presets and focus racking. Changes based on active selection (Exposure, ISO, focus)

CLIPPING TRAFFIC LIGHTS: Colored indicators for individual channels clipping (Red, Green and Blue). Off indicates no clipping, circle filling up indicates +0.01% of channel clipped and fully circled and lit up indicates +1.00% or more clipped

DIRECT PREVIEW OVERLAYS: States the current DP mode engaged; sRGB/Rec.2020, False Color (Paint palette) or Sensor Clipping (Sun)

SHOOTING PARAMETERS: Shows the Exposure/Shutter Speed, ISP, Focus and White Balance in use. White = Auto and Yellow = Manual. Long press category to lock

[0.2] Introduction

Welcome to the world of MotionCam Pro! You've now taken the red pill of mobile cameras and are about to embark on a journey that will challenge everything you thought you knew about filming on a phone.

This guide will cover every aspect of the app, from basic operation, user interface, getting started, how to handle outputs, to the technical details behind its capabilities. We'll strive to fully answer most common questions, address all potential issues you may encounter, and explain exactly how things work. Our goal is to give you comprehensive understanding of the app, all available in a one-stop location so you can confidently dive into the world of RAW and advanced mobile videography!

While we will cover most of the essentials, this guide won't delve into complex topics like color grading or ‘best settings’ in DaVinci Resolve, nor the specifics of third-party mods and enhancements relying on root access (which are often popular with the app). These topics are highly complex and depend on individual workflows, preferences and even device models. However, MotionCam Pro stands on its own and is designed to harness the full potential of your device's camera, with or without these external enhancements or programs.

Anyhow, you’ve arrived at the Mt. Everest of Android camera apps! I’m Armando, better known as Ragusaucy, your MotionCam Sherpa! We’ll start by covering what the app is, what makes it special, then cover important basics and technicals. We’ve got a crash course for newbies too if you are new to this! Afterwards, we'll hit the ground running with usage and UI! Let’s get started then!

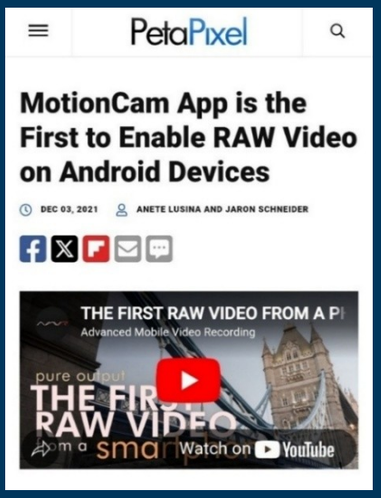
Our Origins

A highly skilled developer, Mirsad, was inspired by Google Pixel phones using multiple frames to create HDR photos. He created a camera app that used a similar technique of harnessing a stream of RAW images, and aimed to improve upon it by avoiding shutter lag and blur during movement. The app continuously captured a burst of RAW images from the same stream used by Google's HDR pipeline and leveraged a motion aware stacking algorithm, resulting in photos with no shutter lag that are resilient to movement, hence it was appropriately called MotionCam.

A visionary power user, Sebastian, having sought for the best possible smartphone camera quality for some time prior, saw the app's potential to capture a burst of RAW images and asked Mirsad to extend the burst, explaining the idea of an app capturing RAW video on a mobile device - an impossibility at the time

As a result of this collaboration, the app achieved a way to capture this burst of RAW images for longer and longer, culminating with the creation of a proprietary, purpose-built format called MCRAW! The app's ability to capture true RAW video was born, dawning a new chapter in Android video! Rest is history!

As the app matured, further functionality was added such as processing MCRAW captures in-app into delivery ready or intermediate log video for non-RAW workflows. The next milestone was reached with the "Direct log" mode which does not record MCRAW, but instead performs RAW image-to-video conversion on-the-fly during capture and encodes directly into options like HEVC or ProRes (and log too) without the overhead of RAW video storage. This is an exceptional ability for an android camera app, since it performs the RAW image processing entirely on its own without relying on the android camera stack or the phone's integrated ISP.



The development of MotionCam Pro is also a story of community collaboration. Development over the app has been very active on platforms like Discord and Telegram, engaging with users to get feedback, troubleshoot issues unique to the android landscape of device variety, and understand the needs of mobile filmmakers, all whom have contributed to the powerhouse that the app has morphed into in the current day. This close relationship with the user base has been crucial to the app's continuous development and its ability to stay at the forefront of mobile video technology.

In summary, MotionCam Pro was not born out of a corporate lab but from a vision of two developers and their perseverance. It's an origin story rooted in a technical challenge (unlocking the RAW power of Android's cameras) and driven by a desire to provide professional-grade tools to a community of creators who were previously underserved, and more importantly about the ability to dream!

[0.3] About MotionCam Pro...

As you are here, you are already half way to achieving total control of your device's cameras. You've probably heard all the rave about iPhones and their ProRes Log recording by now, often incorrectly referred to as 'RAW' video. Well, we will show you how to push your Android device to achieve capabilities beyond that! MotionCam empowers users to unlock their device's full camera potential, regardless of their videography experience. Our core mission is to give you absolute control over your phone's imaging quality as well as providing you the tools to achieve it.

Whether you want to capture true RAW video, ProRes Log, everyday video or simply use the app for photography, MotionCam provides comprehensive controls and quality beyond imagination. We believe you'll quickly discover the benefits of this.

What truly sets MotionCam apart is its unique utilization of the Android RAW stream. Typically, Android devices offer developers firmware tools to access camera functions through various image "streams." For example, the photo stream captures JPEGs, and the video stream (also called YUV stream) records compressed video in codecs like HEVC/H.265. You can also capture single RAW images (DNGs) from the RAW stream, which provides the complete and pure sensor data before any processing or compression even takes place; ideal for RAW photography!

MotionCam's innovation lies in its ability to keep that RAW stream open, allowing it to harness and save every bit of uncompressed data generated by the sensor, rather than just a single RAW photo. This is where the magic happens, giving you unprecedented control and visual fidelity over video and photo captures. Forced sharpening? Bad denoising? Dynamic local tonemapping or over saturation? All a thing of the past!

By capturing only from the RAW data source unlike other options on Android or iPhone, our app is capable of fully bypassing your phone's badly tuned and often undesired image processing that has become associated with phone video capture.

[0.4] What content to expect ahead...

Ready to begin your journey? We've got a lot to cover, from the basics and fundamentals (if you are new to this, don’t worry, we’ve got you covered!), to the nitty-gritty details, all designed to help you ascend into a true MotionCam maniac like us!

Think of this guide less like a boring washing machine manual and more like a fun "how-to" book for your app experience. Our goal is to walk you through not just the app's many features, but to turn you into a filmmaking force to be reckoned with, all while navigating and learning the intricacies of the Android world. Perhaps by the end of this you’ll even curse towards your phone manufacturer a little bit!

Even if you’re already a pro with a DSLR or Mirrorless camera, stick around! The Android world and smaller sensor capturing have both their own quirks and tricks, and we'll cover all of them (or at least we’ll try our absolute best!) But hey, if you're already familiar with the basics, feel free to skip to the good stuff like the UI, App functions, and the FAQ or Workflows sections (as well as the technical chatter)! Otherwise, we’ll start with basics and essentials!

And remember, even pros needs a little help sometimes! Don’t be a stranger. The MotionCam Pro community is full of amazing people who are ready to lend a hand, some of which I’m happy to call friends, mentors and fellow maniacs! You can find us primarily on Discord, as well as Reddit and even on some discreet Telegram groups. Our community is a huge part of what makes this app great, and many of our users are happy to share their knowledge and expertise. The wealth of expertise, knowledge and skill sets present in our ranks is unparalleled when it comes to any Android Camera app, or many apps altogether. Embrace our RAW brotherhood (or sisterhood, you’re welcome too!)

With that said, let us begin your learning journey!

Table of contents

4.0 USER INTERFACE & FUNCTIONS —— p.39

[1] Essentials Crash Course/The Basics

[1.1] What is RAW video/photography?

If you read about all those aspects and remain undeterred, then you have my congratulations! You are part of the 1% in the Android user base. Still feeling confused? Well, let's try to speed run this!

So what's RAW video really? If you are already familiar with all of this, feel free to skip ahead as this course section is for greenhorns, but otherwise, let's demystify this whole "RAW" thing.

Quick heads-up, I'm going to assume you have only a basic understanding of photography at first because if you're not fully familiar with what a RAW image is, you might fall behind quickly. It's totally okay, though! A surprising number of people find their way to MotionCam while searching for the ultimate quality and control, therefore I'm going to do my absolute best to help everyone follow along every step of the way and ensure nobody gets left behind! Let's talk about why we're here: quality.

You've probably heard the term RAW or even DNG (Digital Negative – a RAW file format) thrown around by photographers and videographers (as well as ourselves earlier) who sound like they're speaking a different language. Don't worry, it's not as complicated as it sounds! Think of it this way; a cooked meal vs a bag of ingredients

RAW is quite a fitting name surprisingly. Imagine you're a chef. You've got all these fresh ingredients: a nice tomato, some crisp basil, and garlic. You could just grab a jar of Ragu (haha, see what I did there?) and call it a day. That jar of Ragu is basically a JPEG file. The camera has already chopped, cooked, and bottled everything for you. It's ready to go, but you can't really change the recipe. It's a done deal. It's like saying you want to take out the onions in that jar of Ragu… Good luck! In the same spirit, try changing the white balance of a JPEG, you'll quickly realize it's a bad idea

A RAW file, on the other hand, Is that shopping bag full of untouched ingredients. It's all the original data from the camera's sensor, but it's not a finished image you can share. You have to "cook" it yourself in an editor program like Lightroom for example. You get to be the chef, deciding how much brightness, contrast, and color to add, to name a few of the limitless possibilities. Remember that white balance I mentioned as an example? You can change it after capturing with zero penalties! This gives you ultimate creative control.

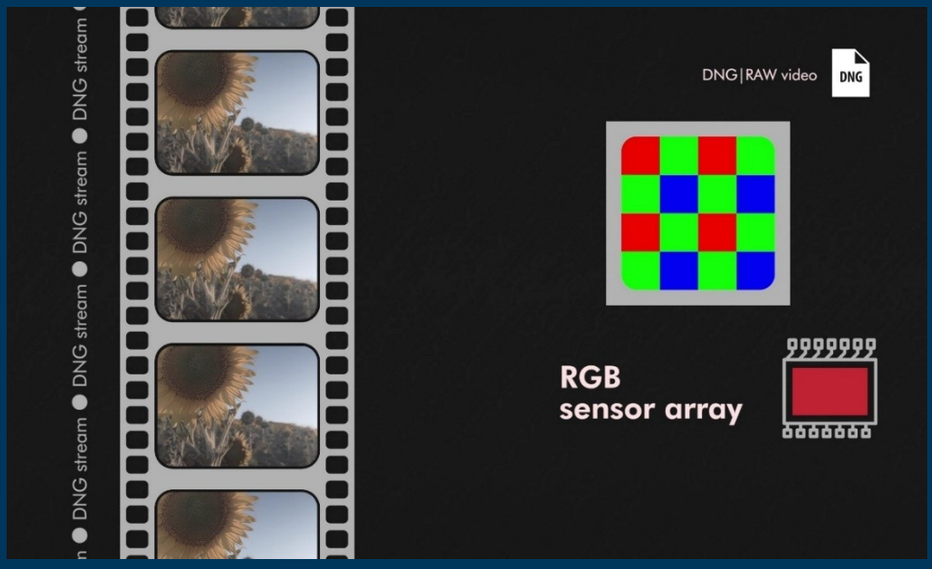
Now of course, if you cook everything fresh, you'll need a whole pantry and refrigerator to store all the uncooked ingredients which take far more space when not ready. In the same spirit, RAW files are much larger than a JPEG.

You also don't open up your fridge to impress your partner with the future lasagna you are gonna cook them, nor do you serve them a frozen meat slab and say look how high quality the wagyu is (if you do, you'll be single soon!)

Essentially, be aware these are not files you can simply open and expect them to look great at first. You will often need special programs to actually see these files too, MotionCam RAW outputs are no different.

It's completely normal for the image to be noisy, soft, flat looking/colorless, or downright underwhelming at first (not always). This will quickly change once you get to cooking however (denoise, grade, stabilize, etc.), and a great chef can absolutely make a steak come alive! The same goes for RAW imagery!

Now, what is RAW video? It's quite literally the same! Except instead of a single image, you are now shooting at 60 RAW frames every second with audio for example. All the same benefits of RAW photography apply!



Here's where things diverge from the photo comparison however. With photos, as it's a single file, you can afford to have a decent file size at the end, so you 'compress' a processed RAW image with all but the essential details and make it a JPEG, but with video this is still too much information, so you gotta make it even smaller! Enter the solution…

[1.2] Encoding/Decoding

Video encoding is the process of compressing a RAW video file into a smaller, more manageable file size, beyond what even JPEG accomplishes for photos. It's essential because RAW video, which is the uncompressed footage straight from a camera, is enormous. Think of it like taking a giant, bag of tomatoes and fresh herbs, then making it into a nice bottle of ketchup. The goal is to reduce the file size without a noticeable loss in quality. It's like saying lol, you encoded 'laugh out loud' into 'lol', and the reader decodes it back into laugh out loud again once they read it. Simple enough, right?

Imagine you have a long sentence you want to tell a friend. You could write it down in three different ways:

No Compression (The RAW, Full, Unedited Sentence)
"I will be there in approximately ten minutes to pick you up so that we can go to the concert together." This is the original message, with every word spelled out. It's the most accurate representation of what you want to say. In video terms, this is like RAW video —huge in size, but with every single pixel of information preserved perfectly.

Lossless Compression (Using Abbreviations)
"I will b there in ~10 mins to pick u up so we go 2 the concert together." Here you simply used common abbreviations. You've made the sentence shorter, but your friend can still perfectly reconstruct the original meaning. This is like lossless video compression, file size is smaller, but all the original data can be recovered perfectly without any loss of quality.

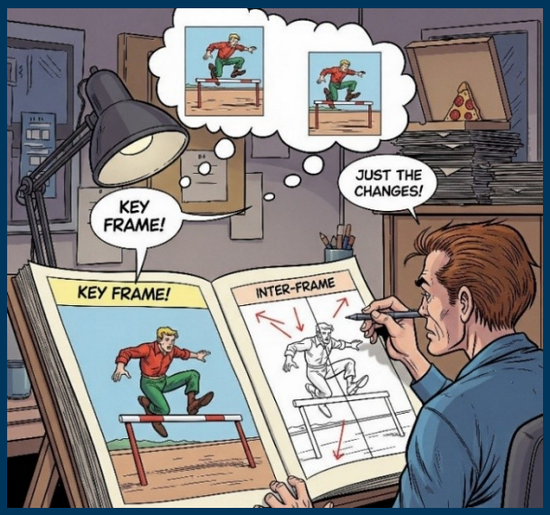
Lossy Compression (Using Slang)
"brb, omw to pick u up for the concert. ttyl." Now you've drastically shortened the message, but you've also lost some of the original information. "I will be there in approximately ten minutes" is compressed into "omw" (on my way). The "approximately ten minutes" part is lost entirely. "so that we can go to the concert together" is also heavily compressed. You have to infer the full meaning, and some details are gone forever.

Will your friend notice enough to care? Probably not. This is like lossy video compression. You get a much smaller file size which is what you may be used to, which is great for streaming or storage, but you permanently lose some of the original data. You can't get back the original quality once it's been compressed too much. With videos or photos, the human eye might not necessarily notice the missing details, but they are gone forever.

Encoders tend to operate under different rules that they follow for compressing, depending on which use case they are intended for, as well as how advanced or new they are (like generational slangs!)

It's kind of like asking 'w.t.f.' to a polite person, they may think Why The Face? You need to have a standard 'set of rules' to encode (shrink) and decode (unshrink) in a way that can be followed along. These 'rules' or compression languages are called codecs (COmpressor/DECcompressor)

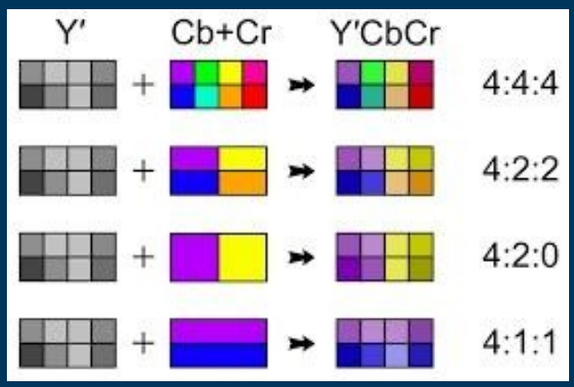
Some codecs are better for editing (Like ProRes) which are not as tightly packed and compressed but give a good enough size reduction without compromising quality. Some are better for final delivery or storage, like H.264/H.265 or AV1, but may be harder to edit because of how compressed they are, as well as they may have compressed away too many important details sometimes.



Video codecs work by getting rid of information our eyes won't miss. For example, in a video of someone talking, the background probably doesn't change much. The codec notices this and saves that background information just once, instead of thousands of times for every single frame. This clever trick makes the file size much smaller. Because a lossy codec throws away some data, it's a trade-off: you get a small, efficient file in exchange for a tiny, often unnoticeable, drop in quality.

Now, if you survived all of this, I'm sorry for the torture, but there's also another aspect, haha! Chroma subsampling… I was unsure if I should have covered this one but the gist of it is a video "shortcut" that reduces the amount of data needed to store or stream a video by sharing color information between nearby pixels to avoid saving it for all of them. It works because our eyes are much more sensitive to changes in brightness than color, so we don't notice the missing color detail.

I won't go beyond here but you'll see some codecs can do 4:2:0, 4:2:2 and 4:4:4, essentially its how densely you compress the colors. If you see someone flex they can record in 4:2:2 and you can only do 4:2:0, joke's on them— RAW video is even better than 4:4:4! It's a detail that's a magnet for forum arguments and that makes the inner spec junkie happy. On a serious note however, using better chroma subsampling can help you with stuff like green screen editing and whatnot but don't stress too much about it if you didn't know what it meant already as that means there's a good chance you didn't need it (unless you are a pro, then you probably know when it's needed)



As far as decoding goes, it's simply about uncompressing all of the above via the same set of rules or language used. Both encoding and decoding may be done via hardware acceleration (an integrated device chip that specializes in encoding/decoding) allowing for efficient recording and playback as well as software acceleration (Completed via brute forcing by CPU) which is much harder but allows for recording or playback even if you don't have a special chip – however it's substantially more demanding and will impact performance more.

[1.3] Bitrate

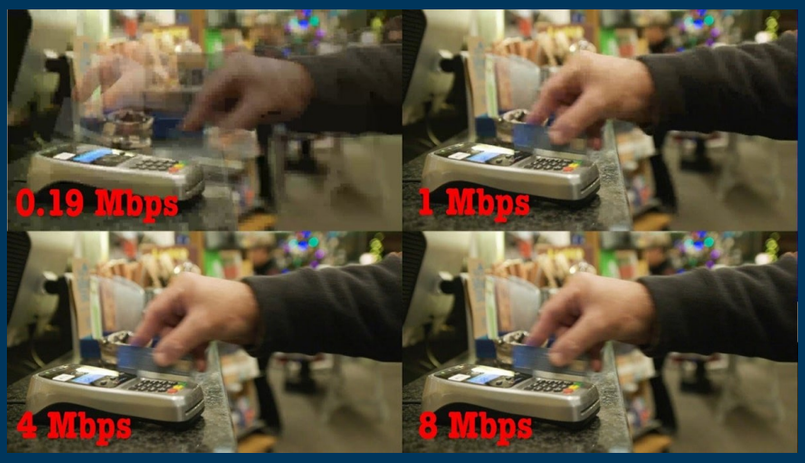
This one's a breeze, I promise! Bitrate is simply the data density of your video—how much information is packed into each second. It’s measured in Mbps (Megabits per second, not Megabytes—a very important distinction!). Think of it as the maximum weight allowance for your digital luggage.

Have you ever flown on an airplane? Let’s say the airline gives you a codec, your luggage (the set of rules for compressing your video). The luggage’s job is to efficiently carry all your "essentials" as a standardized container. The bitrate is your luggage's weight limit. If you have a 30kg limit, you have to stick to it or risk paying a penalty.

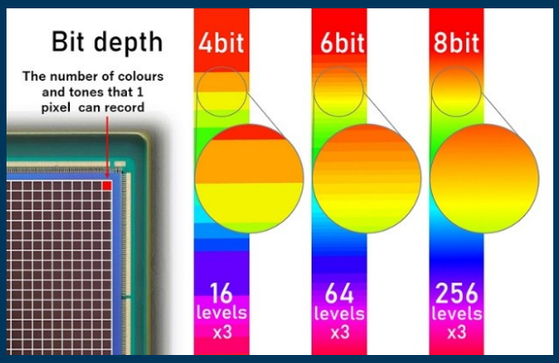
The same goes for your video! Your codec will try its best to maximize data efficiency while staying within the bitrate limit you've set.

Too much bitrate? You get to pack extra things you might have left behind otherwise, like that nice bottle of fancy liquor. The video will look pristine, and while at some point you might hit diminishing returns (you can only pack so much detail!), it's generally a good thing.

Too little bitrate? This is where you have to make tough choices. You might be forced to decide if you can even bring an extra pair of underwear. The quality suffers, and you’ll start seeing nasty compression artifacts like blurry smudges, "ghosting" during movement, or blocky blobs in dark areas.
It's all about finding that sweet spot. It doesn’t matter if you have higher resolution (a bigger luggage), your weight limit will ultimately restrict you too. It’s why some older YouTube videos in 720p looked better than today's 1080p content—they were simply given a higher bitrate allowance, meaning more "data density" and less artifacting. So, choose your bitrate wisely, and avoid a digital baggage claim disaster (Yes, I am a Dad... How did you know??)

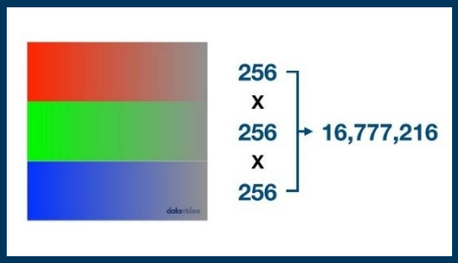


[1.4] Color depth/Bit depth/White level



8-bit, 10-bit, HDR, SDR, Log… What does it all mean? Well, an important factor about recording is the bit depth of your content or that the sensor output at. Also referred to as color depth, bit depth is a fancy way of saying how many colors your sensor can capture and reproduce.

Think of your camera sensor as a little artist, and its job is to paint a picture of the world it sees. The tools this artist has at its disposal are what we're talking about here.
Bit depth/color depth, is a super-fancy way of saying "how many colors can my camera use?" Imagine you're drawing a beautiful sunset. If you only have a small box of crayons, say 8-bit, your options are limited. The sky might look a little blocky, with a few chunky transitions from orange to red. You've got 256 shades of each primary color (red, green, and blue - so 256 x 256 x 256), which gives you over 16 million possible colors in total. That sounds like a ton, right? It is! But believe it or not, our eyes are so good that they can still see those little "steps" in the color gradient in particular if you push the edit too hard or compress too much, especially in smooth skies or subtle shadows. This is called color banding.

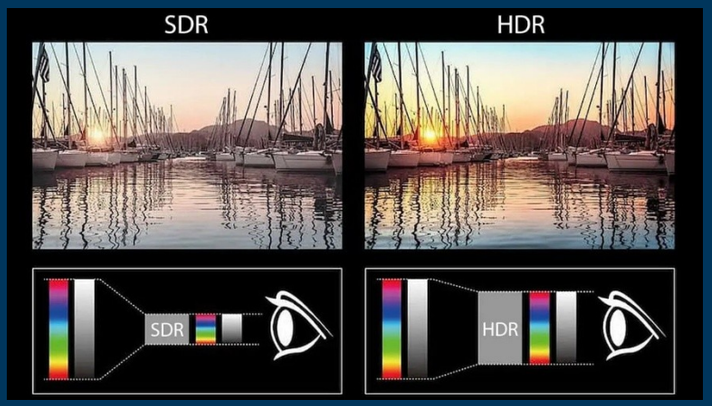


Now, if you upgrade to a bigger crayon box (a 10-bit sensor) you get a whopping 1,024 shades for each color. That's over a billion possible colors! With all those extra crayons, your sunset gradients are going to be buttery smooth. The extra colors fill in those gaps, giving you a much more lifelike and detailed picture. This is the superpower of higher bit depth: it captures and reproduces richer colors and smoother transitions.

To summarize, 8-bit is 256 shades per channel, 10-bit is 1,024 shades, 12-bit is 4,096 then 14-bit is 16,384 and so on.

Essentially this allows you to capture more data and reproduce it as well. As far as capturing goes, more never hurts, but of course more data means more storage is required!

When it comes to the final product, your video will be in one of two flavors: SDR or HDR. SDR, or Standard Dynamic Range, is the classic, everyday video format we've been using forever, which is usually 8-bit! Think of it as your regular TV show. It's designed to look good on almost any screen and is easy to deliver over the internet. To give you an idea, most of the videos you see on YouTube are in SDR because it requires less data to stream, and everyone can watch it without any issues. It's a reliable workhorse. If you don't believe it, search any video and open the resolution settings on YouTube and search for the 'HDR' tag, you'll quickly realize how uncommon it remains.



High Dynamic Range (HDR) should be understood as more than just an "upgraded" version of SDR; it gives drastically upgraded representations for light and color. It primarily leverages higher bit depths, like the 10-bit recording you use, to capture and display a vastly wider range of brightness and colors. This means you can see much more subtle detail in the super-bright highlights without clipping them, which is the core goal of professional capture. Essentially, 10-bit allows your screen to have over a billion more potential "tones" or brightness levels to display colors smoothly. When you try to compress a high-contrast scene into an older 8-bit space, you are forced to create the same sense of brightness with dramatically less steps (or fewer "crayons") to work with, which easily leads to visual banding artifacts when you grade. This light compression is known as tone mapping.

This concept is easiest to grasp by comparing it to audio. Have you ever seen a movie where the dialogue is barely audible, and then an explosion happens that is so loud it violates your ears? This is audio dynamic range at play; the dramatic difference in loudness is meant to retain the realism of the event. The reason this effect occurs however is that the audio designers have to account for subpar or limited audio systems that can't provide with ultra high fidelity audio separation not unlike your video needing to account for an 8-bit/SDR display with limited tones. So they 'compress' the noise, like we would do via tone mapping, by making the speech dimmer (like we would make the shadows dimmer to accentuate the highlights, leaving us extra 'crayons' to better represent the intensity of the bright areas), but this creates sometimes obnoxiously loud sound effects versus speech as a drawback. In our case, this makes it that you must often sacrifice deep shadows not unlike movies sacrifice the loudness of speech.

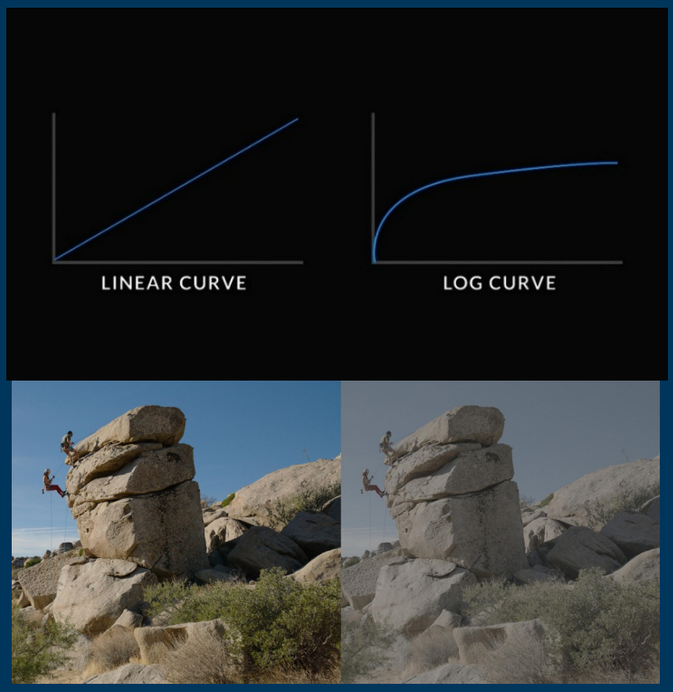
If the explosion were barely louder than the speech, it wouldn't be perceived as a loud event. With your eyes, it's the same thing. If the sun in your footage is just another flat, bright orange patch compared to a blistering, brilliant white, is its intensity even perceptible? That lack of perceptual difference is the weakness of SDR. HDR eliminates this by giving you the ability to capture and display those blistering whites and deep blacks by separating them with additional tones that you can further push to accentuate this drastic difference in brightness. This, ensures the visual impact of brightness is retained from capture to delivery.

However, delivering HDR content is tricky. Not only are the file sizes larger (remember the luggage example?), but the screen you're watching it on has to be able to display it properly by also getting way brighter to accentuate these extra tones, else you're not able to make them look any different for the viewer. If a display isn't HDR-capable, your content will appear lacking in contrast and saturation, which is why you don't see HDR everywhere yet—it's still a bit of a fancy, high-maintenance friend best reserved for professional projects and those with the right gear. Don't be allergic to 8-bit/SDR exporting! You'd be shocked at how many people feel using it is a bad idea.

Now I hear you thinking, why shoot in 10-bit if you'll export in 8-bit? Well, it's similar to shooting in 4K to export in FHD 1080p, in this example you could afford more cropping or reframing while maintaining more details on your final edit that would have been lost had your initial capture been in 1080p. Editing latitude!

You also get to control the tone mapping process better. As mentioned prior, Tone mapping is the crucial process that distributes the wide range of brightness from a high dynamic range camera capture onto a standard display, or otherwise compresses a wide array of brightness in a scene into a narrower range. This ensures that you can see detail in both the brightest highlights and the deepest shadows, preventing clipped whites or crushed blacks that exceed your display's limits (exceeding its palette reach). Going too hard on it however can sacrifice the pronunciation or intensity of rich highlights and deep shadows, making them feel duller, which is why preserving control over how much is applied works better; and hence the allure of HDR – you can preserve this sense of brightness far better without compressing it or downplaying it!

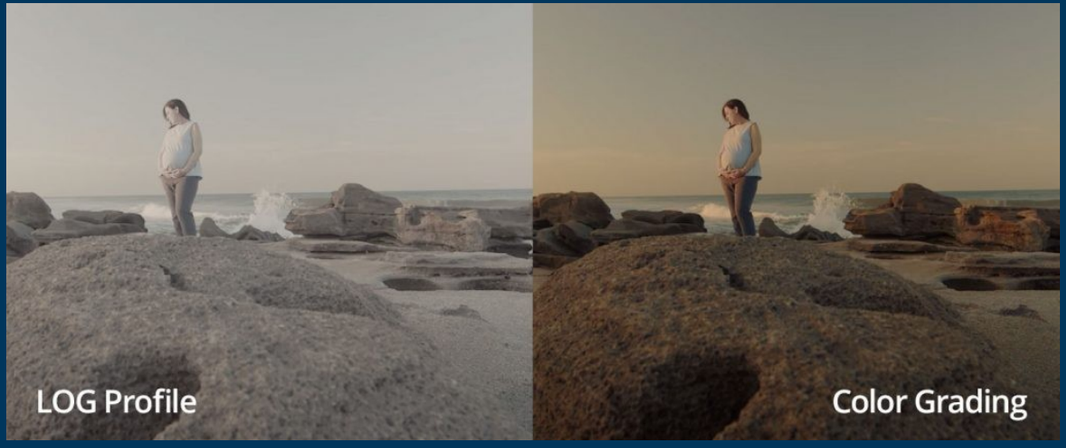
So, you've decided to shoot in 10-bit to get all those lovely colors. But what about Log? Think of Log as a special way of packing your video data to protect it. When you record normal encoded video, the image colors are generally already in a way to that may look good right out of the camera. But this can sometimes impact the deep colors out the very bright or very dark parts of the image during encoding, losing some of that precious detail in those areas.



Log, on the other hand, is a recording mode that flattens out the colors and contrast on purpose. The resulting footage looks a little gray and washed out, which seems strange, right? But it's actually a brilliant trick! It's like putting a "fragile" sticker on your luggage, but instead of the airline being careful, this sticker tells your editing software, "Hey, this luggage is full of precious data, so don't compress it too hard!" This flat look gives you a ton of flexibility later on.

When you get the footage into your editing software, you can "de-log" it using a special filter called a LUT (Look-Up Table) or by simply telling the software what camera setting you used. This brings the colors and contrast back to life, but you now have way more room to play with the image; adjusting brightness, colors, and shadows without the picture falling apart.

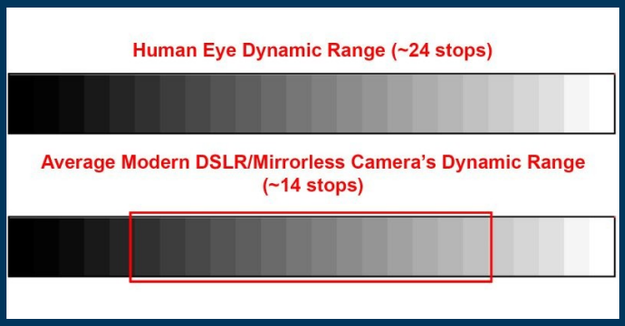
While it's no RAW video by any means, Log does preserve more data than standard delivery methods, while keeping a notable size advantage. While it's possible to shoot Log in 8-bit, it's generally not recommended because you'll lose a lot of that rich data you were trying to protect in the first place! The magic of Log really shines when it's paired with a 10-bit (or above) sensor. And don't worry! Most phone cameras nowadays almost are guaranteed to have a 10-bit sensor! More on that later.



[1.5] Dynamic Range

You've reached the end of the crash course! Before we jump into the app, let's cover one of the most critical aspects of cameras that you'll have to deal with every time you shoot. FYI, we'll be talking a lot about stops (1 stop = doubling or halving of light intake, whether through ISO, exposure or aperture – eg. Going from ISO 500 to 1000 = +1 stop)

Dynamic range is a camera's ability to capture detail in both the brightest and darkest parts of a photo at the same time. Let's think of a scene's range of light, from the deepest shadows to the brightest highlights, as a piano. A scene with a very wide dynamic range is like a huge, expansive piano with many, many keys, representing a vast range of tones. A scene with a narrower dynamic range is like a smaller piano with fewer keys.



Now, your camera's dynamic range is like the reach of the pianist's arms as they sit at the piano. A camera with a limited dynamic range has a pianist with shorter arms, only able to comfortably reach a certain section of the keys. A camera with a wider dynamic range has a pianist with longer arms, able to play across a broader section.

The human eye is like a pianist with incredibly long arms, able to perceive and distinguish almost all the keys on even the biggest piano. However, a typical modern camera's "arm reach" (dynamic range) is more limited, usually capturing around 12-14 stops of light. This means that when faced with a "very big piano" scene (high dynamic range), the camera can only capture the details within the reach of its "arms."

When the "piano" (the scene's dynamic range) is wider than the "pianist's reach" (the camera's dynamic range), you face a choice. This is like the pianist's thoughts, deciding how to prioritize their limited reach. If you want to make sure the dark areas have detail, you'll need to expose for the shadows.

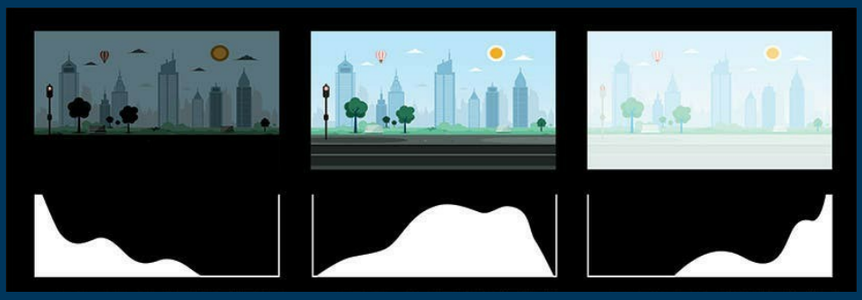
This is like the pianist focusing their reach on the lower keys to capture the deep bass notes. However, this risks "clipping" the highlights, where the brightest parts of the scene turn into pure white with no detail.

Alternatively, you can expose for the highlights to preserve detail in the bright areas. This is like the pianist shifting their reach to the higher keys to capture the bright treble notes. But this risks "crushing" the shadows, making the darkest parts of the scene become pure black with no detail.

Understanding that your camera has a limited "arm reach" compared to the vast "piano" of the real world, and learning how to choose which part of the "keyboard" to prioritize through exposure, is crucial for capturing the image you intend.

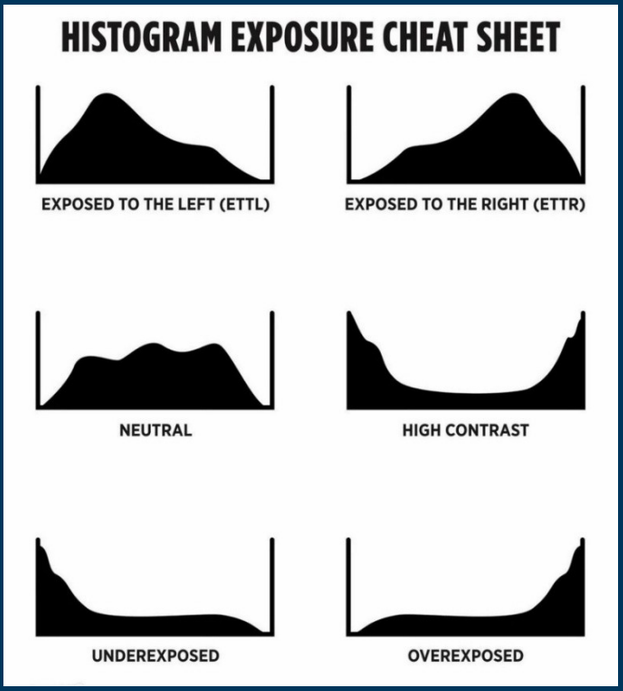
Now you are probably wondering, can't I increase my reach somehow? The answer is – it depends. Bigger sensors generally have more dynamic range, with phones being around 11 stops give or take and a Full Frame about 14-16 stops. Phones can apply HDR merging in which they can sometimes, via hardware methods, capture multiple exposures simultaneously during video, and blend them. We won't touch on this yet (*Cough* DCG… *Cough*) but it is indeed possible to extract such power with MotionCam. Some phones do this, like iPhones, but since they often use the different (inferior) methods, it can introduce visual artifacts like ghosting and whatnot. Nevertheless, it is indeed possible to go head to head with a much bigger sensor.

Back on track, how do you know if your "pianist's arms" are long enough for the "piano" you're trying to play? This is where the histogram comes in. A histogram is a graph that visually represents the tonal values in your photo. The left side of the graph represents the shadows (the low notes), the right side represents the highlights (the high notes), and the middle represents the mid-tones. The height of the graph at any given point indicates the number of pixels at that specific brightness level.



Think of a histogram like a soundboard or a visualizer for the piano example. If the graph is all clustered in the center, it means your scene has a very narrow dynamic range, like a small piano with only a few keys. If the graph is spread out across the entire horizontal axis, it means your scene has a wide dynamic range, like a big piano.

When a histogram touches the left edge, it indicates that the shadows are "crushed"—there is no detail in the darkest areas. The pianist's arm couldn't reach the lowest notes, so all those details are lost to pure black. When a histogram touches the right edge, it indicates that the highlights are "clipped"—there is no detail in the brightest areas. The pianist's arm couldn't reach the highest notes, and all that information is lost to pure white.

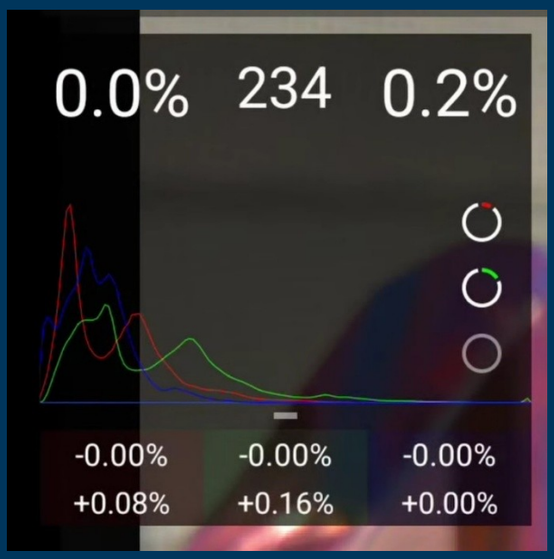


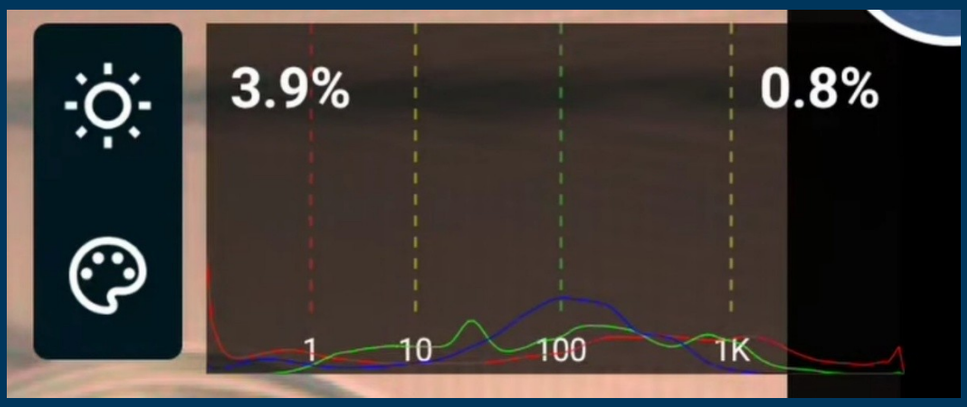
By learning to read the histogram, you can quickly assess the dynamic range of a scene and decide how to best expose your image to capture the most important details, avoiding either crushed shadows or clipped highlights.

We'll touch more on the histogram during the UI section, but rest assured, with MotionCam we've got one of the best ones available in modern cameras! A RAW + Video exposure histogram each, since as stated prior different strategies apply! More on that later, but for now, enough of the theoretical – you've now officially graduated the Ragusaucy Foundational Crash Course!

I do want to give a disclaimer, some aspects have been heavily oversimplified to ensure basic initial grasp of these concepts. As we ramp up, we'll provide the correct and in-depth technical explanation on these topics for you to achieve true mastery of your outputs, but for now, we'll start with this! If you fancy more hard-core content, we've got your back as well! Consider this your departure from Everest Base camp!

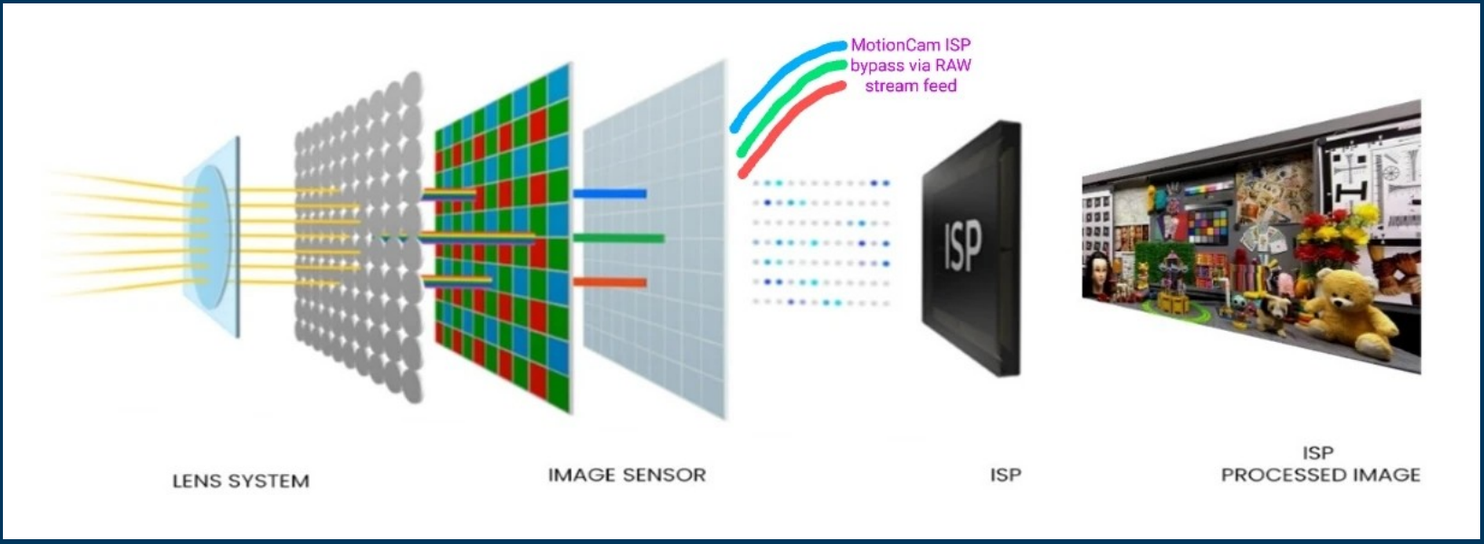
Ready to get into the app at last? I can see you may be still confused, but don't worry! No time to think, we'll make it all make sense! Let's begin!!





[2] Why use the RAW stream (Simplified)

You may be wondering why use the RAW stream for video capture when a dedicated video stream exists. To understand its advantages, consider that all phones capture photos and videos with distinct "personalities." This isn't a flaw; it's a feature. Manufacturers tune their camera processing algorithms (e.g., saturation, noise reduction, sharpening, tone mapping) to produce results at their discretion.



Your phone's Image Signal Processor (ISP) efficiently processes camera data, much like a chef prepares a meal from ingredients. Ideally, this should suffice. However, OEMs often mis-tune these settings, leading to poorly processed results. This isn't a hardware issue; rather, these processing settings are often influenced by consumer taste and demand. Imagine if a chef wanted to cook an amazing grilled chicken, but the restaurant owner forced him to only serve fried chicken since he thought people preferred it better! The chef is simply doing as instructed unfortunately, and he’s not taking your (the client) special orders!

The RAW stream however offers pure, unprocessed sensor data, which is crucial for typical RAW imaging that requires preserving the full frame data. By harnessing this aspect, our unique approach allows our Android app to capture RAW video (every frame of the video can be a full and complete RAW) and bypass the Image Signal Processor, giving us complete command of the data and allowing our own pipeline to take over (which you fully command). Consequently, our capture results when compared to stock camera apps and other third-party solutions may not even appear to be shot from the same device, and can achieve video quality previously thought impossible on phones!

[2.1] So what's the catch?

Although the benefits of RAW video and the additional capabilities that come from having complete control are undeniable, here's where we arrive at the complexities of this approach...

[2.2] Camera2API support

Do you recall those developer tools we mentioned previously to enable control of the cameras and such? The Camera2API is a framework on Android that allows third-party apps to access a device's camera. Manufacturers use this to set what camera settings, such as exposure, ISO, frame rates, and resolution, can be controlled and accessed by developers (amongst many other parameters). You can use a Camera2API tester app to find out in better detail about your device's current support levels and functions shown.

However, a problem arises when manufacturers limit or hide capabilities that are present in their own stock camera apps. For example, most notably they may restrict access to higher frame rates or resolutions as well as In-Sensor Zoom crops. This can be an issue for third-party apps like MotionCam, which requires specific functionalities like RAW stream access. If a manufacturer hasn't enabled the ability to shoot RAW images, or in other words, hasn't exposed the RAW stream, the app simply won't be able to run on that device.

While this is less common on flagship phones, it's still an issue for many lower-end devices.

If you encounter these types of limitations, we strongly encourage you to submit feedback to your device manufacturer and let others in your device communities know. Raising awareness can and has sometimes prompted manufacturers to fix these issues in future firmware updates. Strength in numbers!

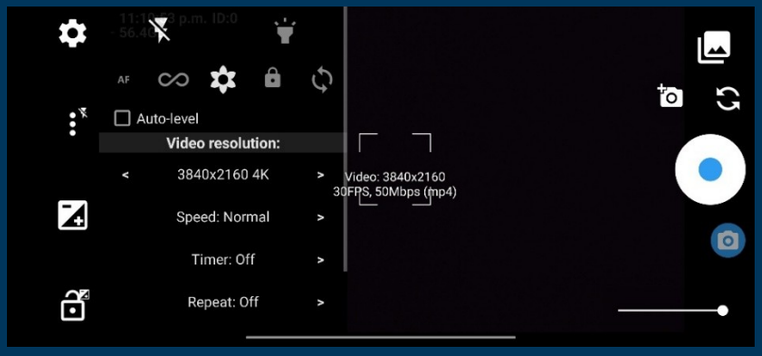
[2.3] RAW stream limitations

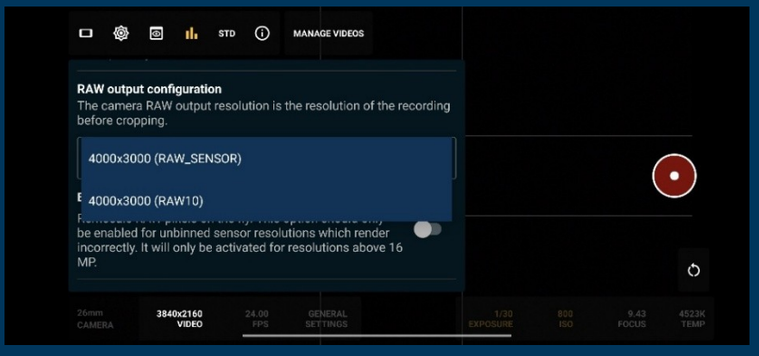
As already stated, the RAW stream provides pure and unprocessed sensor data that the app intercepts before the Image Signal Processor (ISP) touches anything beyond basic sensor operation. The issue however is that the RAW and video stream do not share the same device function access keys.

As an example, if you use a typical video app and request a 4K 60fps SDR video, the device will generally have a capture ‘key’ to use exactly 3840 x 2160 8-bit 60fps HEVC video or something of the like without issues, assuming an acceptable level Camera2API support. In the case of MotionCam however, requesting 60fps means the app is asking your phone for 60 full color depth 12MP images, depending on your sensor, then converting it down to your settings (eg. crop into 4K and reduce down to 8-bit).

What this means is that OEMs may sometimes open up 60fps for video modes, but not for RAW stream capture, leading to a gap in capabilities that shouldn’t exist and is generally a result of poor Camera2API implementation.

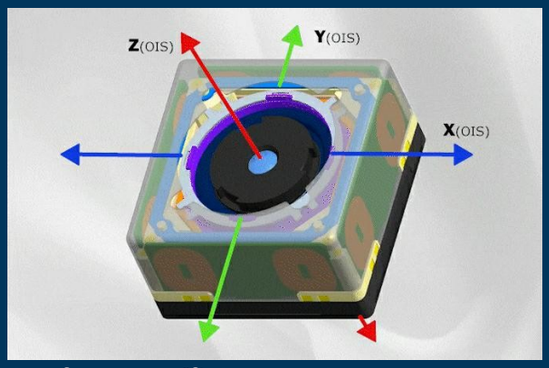
The same goes for resolution settings - where a video stream app may ask for a 7680 x 4320 (8K) processed HEVC video at 30fps, MotionCam would ask for 30 individual, full-color-depth RAW images per second at 50MP.





To be clear, these are strictly firmware limitations at play and will vary per device. Any capabilities available for video mode should be equally available for the RAW stream since all sensor data is initially acquired at the RAW level (even if then processed by the Image Signal Processor afterwards). Google Pixel devices are specially known to show most streams correctly for example, but this is unfortunately the exception rather than the norm.

[2.4] Optical and Electronic Image Stabilization



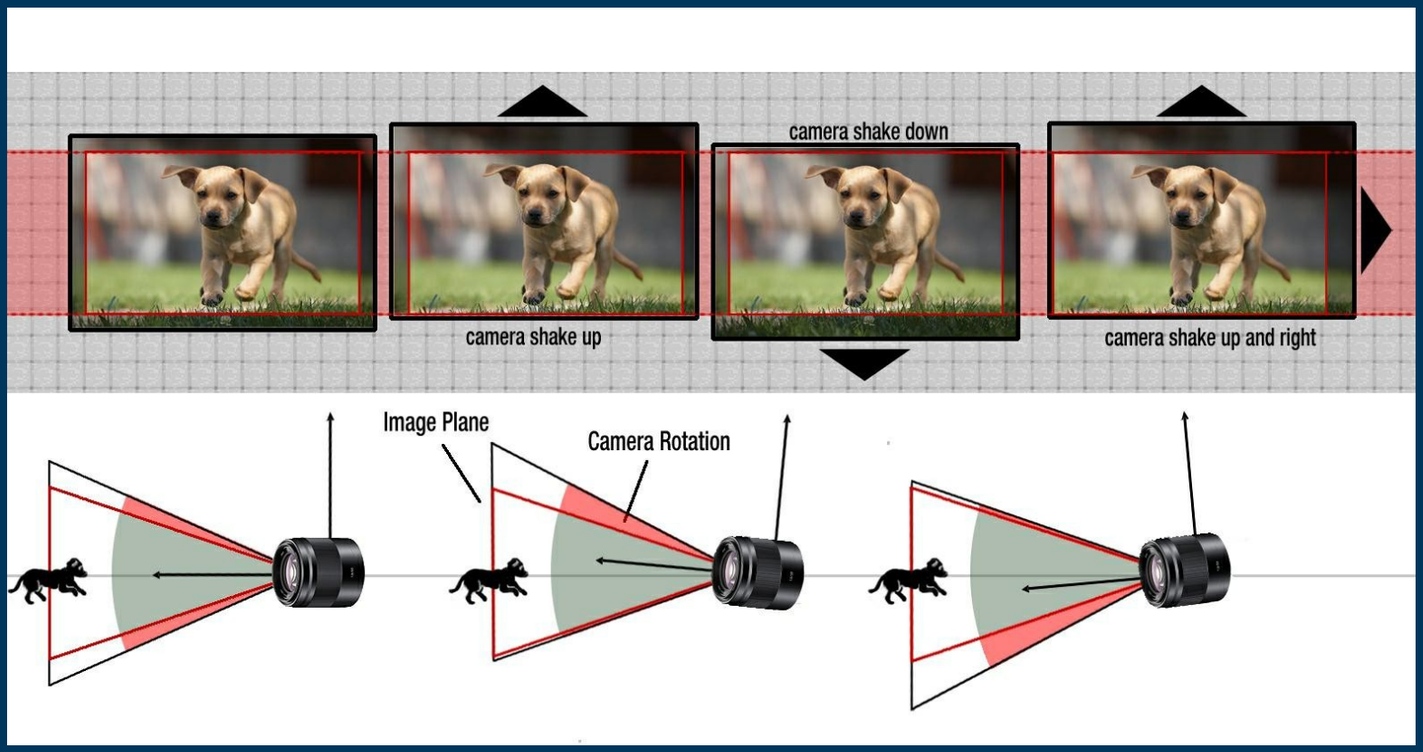
When shooting photos or videos, camera shake is a common issue, especially without a tripod or gimbal. Optical Image Stabilization (OIS), a hardware solution, combats this by physically moving the camera lens to counteract minor shakes and smooth out motion as best as possible, despite lens wiggle room limitations.

Optical Image Stabilization (OIS) is generally the first line of defense against camera shake, especially when using a phone or camera without additional external stabilizers. Although it is an excellent tool, it is often insufficient on its own notably during video, bringing us to the next line of defence.

Electronic Image Stabilization (EIS), also known as Digital Image Stabilization, is often used alongside OIS (Optical Image Stabilization) to smooth out shaky video footage. While users might not always realize it, EIS (Electronic Image Stabilization) is crucial for handling the majority of stabilization during significant movement, especially if shooting handheld. In some cases, EIS may even be the only stabilization available on a device!

Why is this important? Well, EIS is a pretty cool trick, but guess where it is coming from? If you guessed the Image Signal Processor (ISP), congratulations! I owe you a virtual cookie compressed into a .rar file, you'll just have to figure out how to un-zip it.

The ISP does a lot of work behind the scenes, including video stabilization. MotionCam bypasses the ISP entirely as mentioned however. Think of it like this: the ISP is the chef, and you're Gordon Ramsey telling it to get the hell out of your kitchen; all the work now falls on you (yes, that includes EIS…)



In fact, some may even think that stabilization is broken in the app altogether —if you are one of those – aha! We meet at last… Now give me that .rar cookie back and go and sit at the corner for calumniating us on that Google Play review. We know what you did..! What's that? You want to give the app 5 stars instead now? Ok, have it back…

Back on track though, this means when you're shooting with MotionCam, you don't get the ISP-sourced stabilization because a RAW stream is, well, RAW. It's pure, unprocessed data. EIS requires the ISP to perform a few key tasks like…

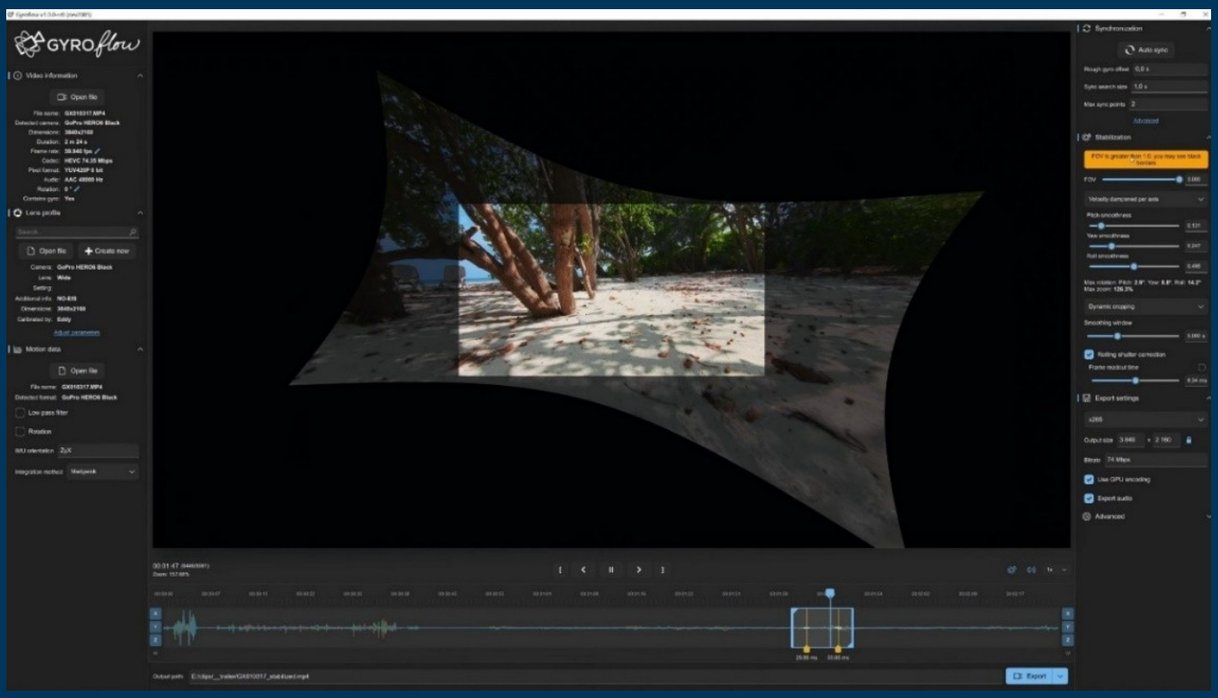
* **Dynamic Cropping**: It crops the video in real-time to smooth out shaky movements.

* ***Rolling Shutter Correction***: It fixes that "jello effect" that can make straight lines look wobbly during movement and panning.

* ***Image De-warping***: It corrects lens distortion to make everything look more proportional (edges in particular).

So, while you might have some shaky footage at first, the trade-off is that you're getting a completely untouched, pure RAW file and the full sensor area to play with. You're in complete control of the final image stabilization, without any of the ISP's automatic (and sometimes heavy-handed) cropping. It's the RAW data versus a "stabilized" convenience, and sometimes, that RAW data is exactly what you want.

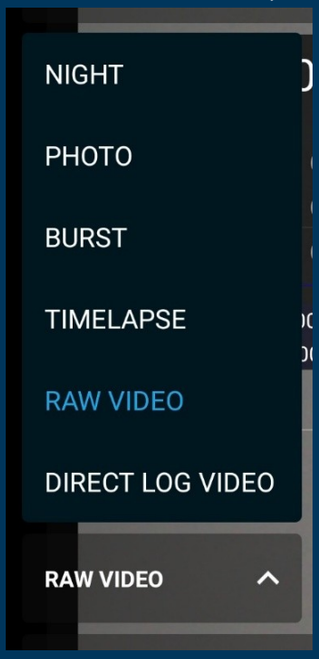
OIS (Optical Image Stabilization) also continues to run smoothly as its lens-based mechanism doesn't conflict with the RAW video stream, nor is it operated by the ISP. For even better stabilization, you have the option to use the EIS/video stabilization tools in any capable editing program like DaVinci Resolve or Adobe Premiere to name a few (as well as Gyroflow, an official app supported option –more on that later), which can provide superior control and results compared to the built-in EIS!



[3] GETTING STARTED

[3.1] App Modes

It's time to put all that prior knowledge into action! I was probably starting to lose you after all that reading, but rest assured it will be worth it! To begin, you will first need to decide what mode you're going to use. As of this current release, here's the options available…



Night: Best for still situations in low light. Overrides exposure set and stacks multiple photos to reduce noise and increase clarity.

Photo: The original MotionCam experience. Captures images continuously in the background to eliminate shutter lag and improve merge quality. Delivers a shutter-lag free, high quality photo.

Burst: Ideal for capturing action. It buffers a specified number of RAW frames (fps selected) from both before and after you press the shutter button, ensuring you can still save that critical moment even if you missed it! Who says you can't go back into the past?

Timelapse: A dedicated mode for shooting RAW timelapses lovers with additional quality of life options vs standard RAW video mode, such as setting interval gaps between frames and total a session duration timer to turn off screen/exit app after completion.

RAW Video: The core and crown jewel of MotionCam. It records RAW video in MCRAW, a highly compressed app proprietary RAW format. It achieves up to 50% data lossless compression over standard uncompressed RAW video. This format saves space and simplifies file management compared to traditional RAW video formats (like CinemaDNG). Heads up! Timelapse mode also captures with MCRAW outputs.

Direct Log Video: Records directly from RAW stream into a compressed video file of your choice instead of MCRAW. Offers a faster, more streamlined workflow by providing a more ready-to-go video file without the RAW size overhead, all while still bypassing the phone's built-in image processor for superior fidelity vs standard video stream options.

This mode is significantly more intensive to perform than RAW video as it captures RAW stream data into a temporary RAM buffer while encoding it on the fly simultaneously (RAW video mode simply buffers and stores without additional steps). Able to use hardware or software acceleration depending on your device's codecs available.

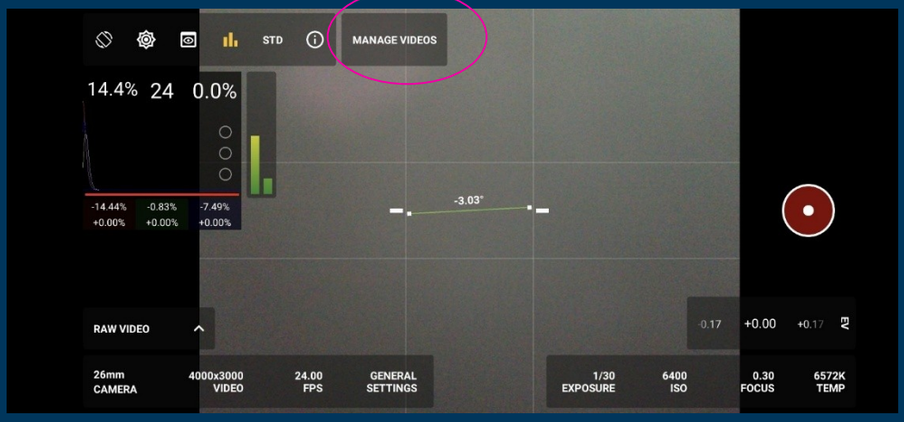
Beware of the performance requirements as this mode will push your CPU and GPU to the maximum!

[3.2] Shooting your first RAW video

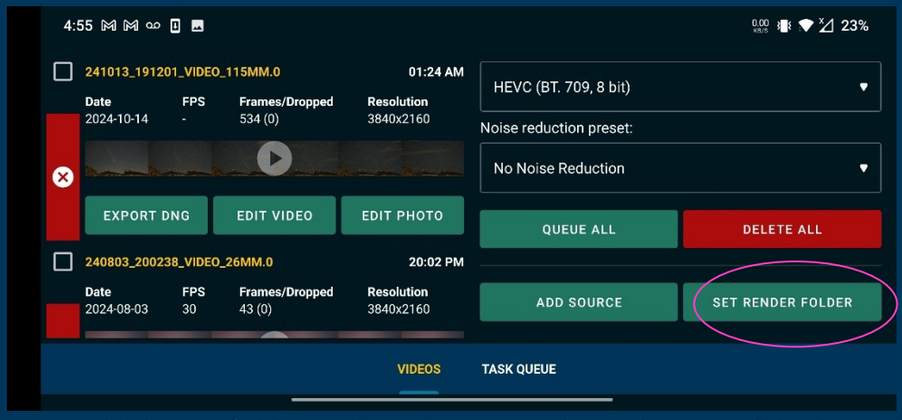
You're probably expecting a long, boring chapter on all the menu buttons. But guess what? We're skipping right to the fun part. The best way to learn is by doing, so let's put that theory into practice. Don't worry about the UI for now; just follow my lead. This is where the magic happens, and it's easier than you think!

RAW video is notoriously complex to edit and manage, but forget about a horrible workflow. This app gives you everything you need to deal with it, all in one spot! And yes, that includes a built in RAW video editor and exporter! Now, let's get our hands dirty. We will assume you've started in RAW video mode, by the way…

1. Start by going into MANAGE VIDEOS

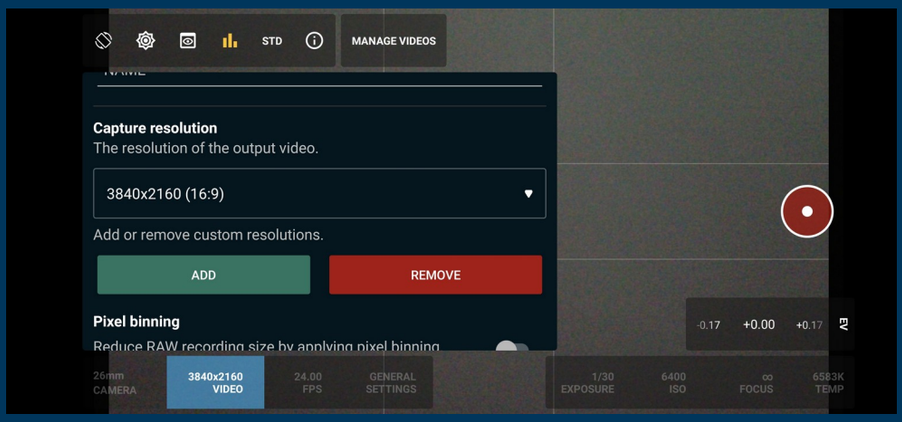


2. Select the SET RENDERER FOLDER button to designate where your RAW video outputs will be exported on completion



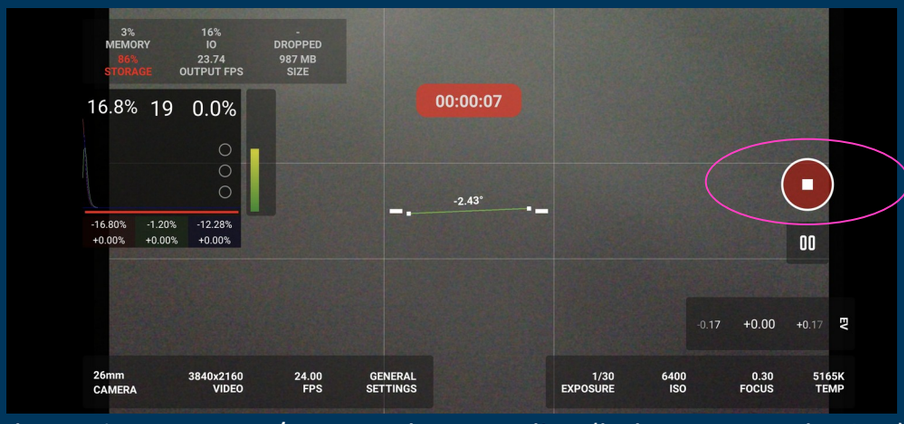
3. After navigating to whichever folder or location you selected, you will be brought back to the MANAGE VIDEOS menu. Simply swipe left gesture to back out of it

4. Once you are brought back, tap on the resolution CAPTURE SETTINGS button, this will open up a sub menu. Simply press on the Capture Resolution box (press ADD to create your own, in case your desired resolution preset is not listed)



5. Once selected, press anywhere outside of the menu on the screen to hide it. Now we are ready to shoot!

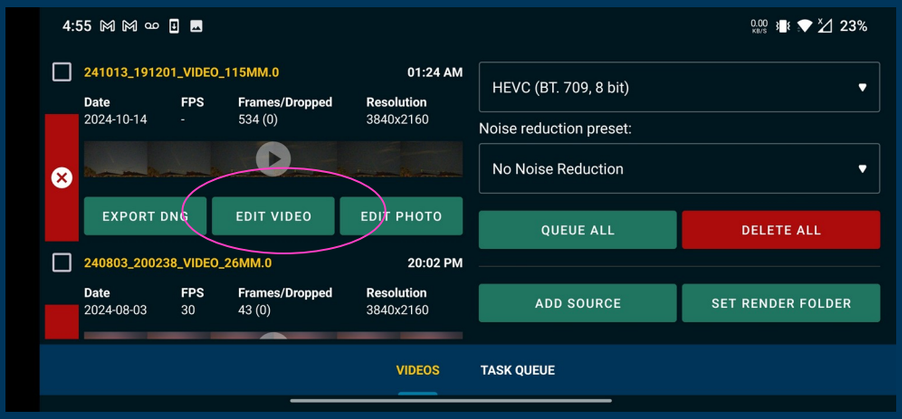
6. Ready? Simply press the red Capture button to begin recording! Once you do, you will notice the User Interface will change.



7. You now have the option to pause/resume the recording (below capture button) as well as the top left will indicate capture session details! Don't worry too much about those, we will explain them later (the most important one being dropped frame, and storage remaining of course)! The capture parameters will remain functional however once you begin recording you can no longer adjust resolution and other such details, beware!

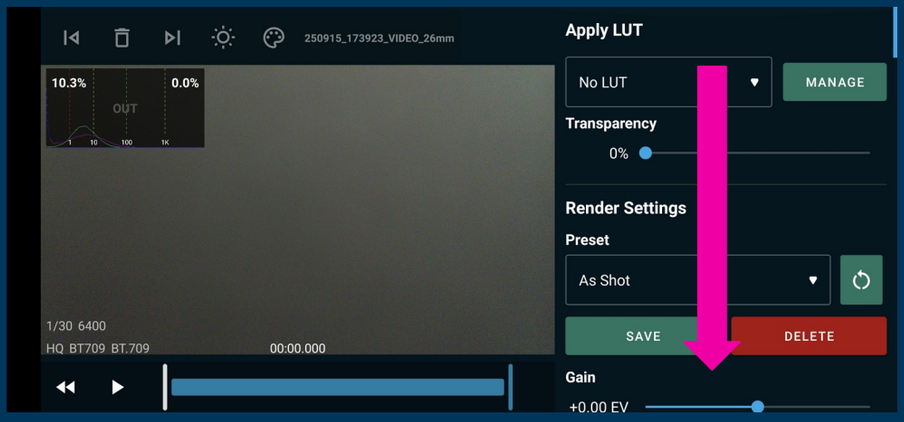
8. Simply press the red Capture button when you are ready. The User Interface will then once again revert back to normal. Press once again on the MANAGE VIDEOS button to re-visit your MCRAW (MotionCam RAW) video captures.

9. You will now see your recent capture (or captures… Did you get carried away by any chance? They pushed crazy numbers on the file size counter, didn't they?? I know, it's beautiful indeed!)

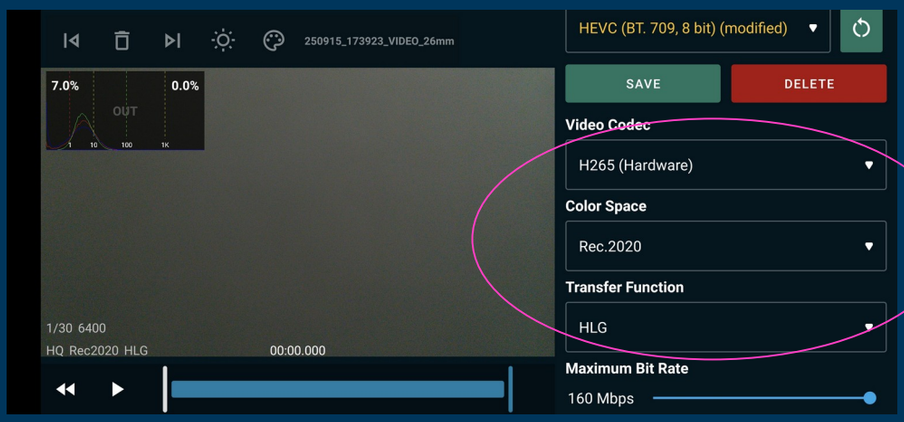


10. On your desired reel, you can observe three options of interest, EXPORT DNG, EDIT VIDEO and EDIT PHOTO. For this demonstration's purposes, select EDIT VIDEO to continue ahead! (Please don't press EDIT PHOTO yet, you're guaranteed to be sucked away and distracted by the cool features there… Yes, this could be reverse psychology…)

11. Assuming you didn't get sidetracked yet, you will now have opened the EDIT VIDEO menu, which allows you to edit, grade, adjust and denoise your MCRAW files! RAW video editing has never been this easy! I would heavily encourage you to play and experiment with the sliders to see how you can warp and modify the source content. Do note the preview is low quality however!

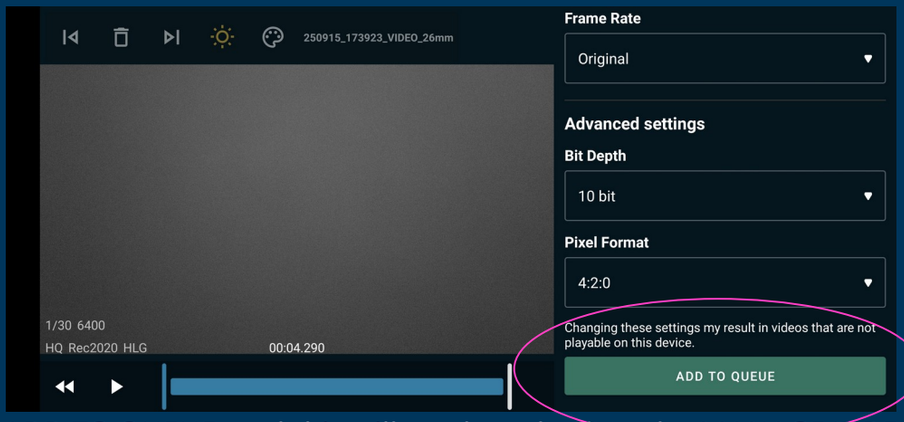


12. After adjusting your video to taste (or not, that's also totally fine!) simply scroll towards the bottom of the settings to prepare your export settings.

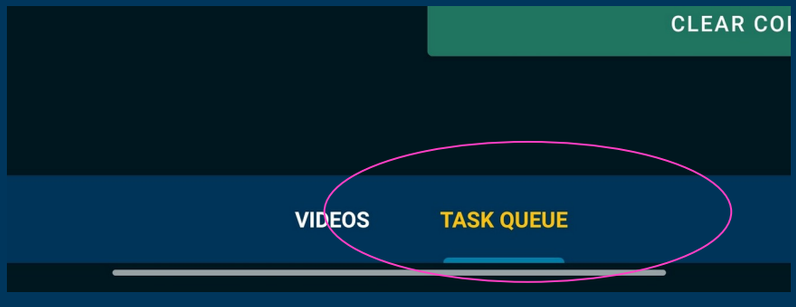


For simplicity purposes, we'll do a classic HEVC file with Rec.2020 in HLG, this is what you may already know as delivery ready HDR video.

13. Scroll down one more time, we can skip the denoising options for now as they can drastically impact exporting time



Simply press ADD TO QUEUE, and this will send you back to the MANAGE VIDEOS menu. You can follow the export process time remaining on a notification that will appear, or on the TASK QUEUE menu.



Congratulations! You've Filmed Your First RAW Video

You've successfully filmed, edited, and exported your first MCRAW! You can find the result in your gallery if your device supports the codec. You will notice a significant improvement in the overall output quality compared to standard video apps—this is the power of MotionCam's control.

Try the Direct Log Mode for a Faster Workflow

If you are interested in a faster workflow, you can try Direct Log mode. This mode performs all the encoding steps we've just done, except on-the-fly during shooting, so you don't have to use the in-app editor afterward!

Adjustments: Adjustments are made in the Direct Log encoding menu *before* you shoot.

A Note on encoding: Using Direct Log is a **commitment**. The data is "hard baked" and irreversible because it discards much of the sensor data to save a significant amount of space. You will not produce a MCRAW file, and the editor will not work on these files.

Viewing Your Files

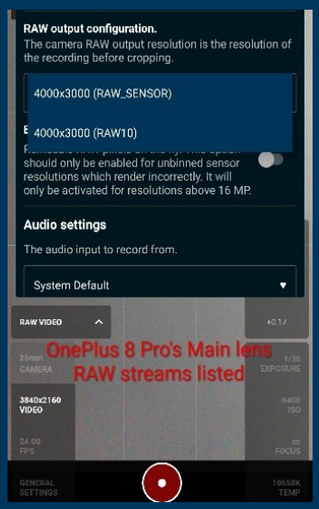
Not all codecs and formats are viewable with default players. To open specialty files like ProRes, APV, and AV1, you can use a dedicated tool like VLC or a video editor.

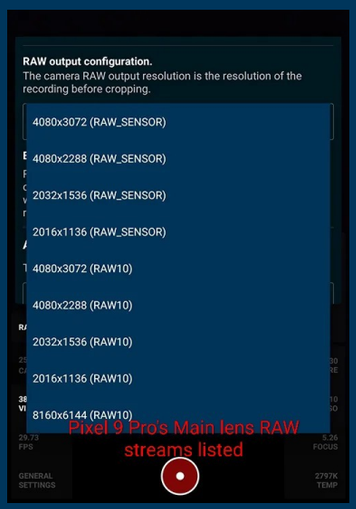
Explore RAW Output Configurations

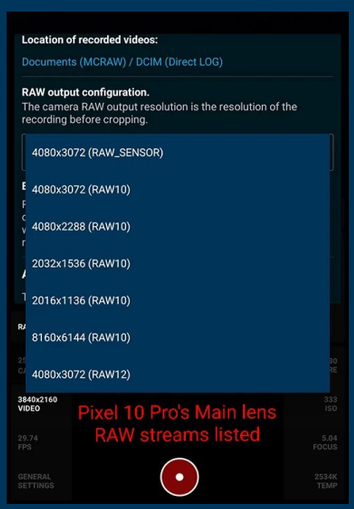
Before we dive into the user interface, it's crucial to understand the **RAW output configuration setting**. You'll find this in the RAW Video, Direct Log Video, and Timelapse modes, below the resolution crop settings (scroll down in sub-menu).

Source Selection: This setting tells the app which RAW stream to use (e.g., "give me 12MP RAWs"). It does not in itself crop your video (cropped streams do exist, however).

Explore & Test: Available streams will vary greatly by device. Explore and test different options. Some may unlock higher frame rates or special features like 12-bit modes







[4] USER INTERFACE & FUNCTIONS

We have arrived at the powerful user interface and inner workings of the app. From this section onwards, I will unfortunately have to be far more technical and specific about the details and functions at play, as it's critical to define what each function does to extract their maximum potential. My recommendation is to go through this section on-demand rather than in a linear fashion (unless of course, you're looking to learn about every literal aspect of the app, that works too!)

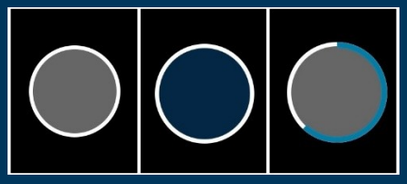
We will now provide comprehensive detailing of what each button, function and tool in the app achieves. We fill first start by covering the Interface of the main modes and their respective overlays during capture. Also covered afterwards will be the sub-menus of the main modes for capture screens.

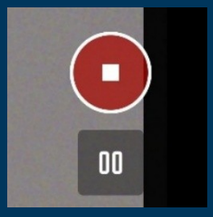
We will then cover the Manage Videos (Video Manager) Interface as well as we will also explain and detail functionalities of the Video and DNG renderers.

Lastly, we'll cover some important functionality aspects as well as terminology. For additional information on further workflows and handling of exported outputs, skip to the next section ahead (Chapter 6 – MotionCam Workflows: Find your Persona).

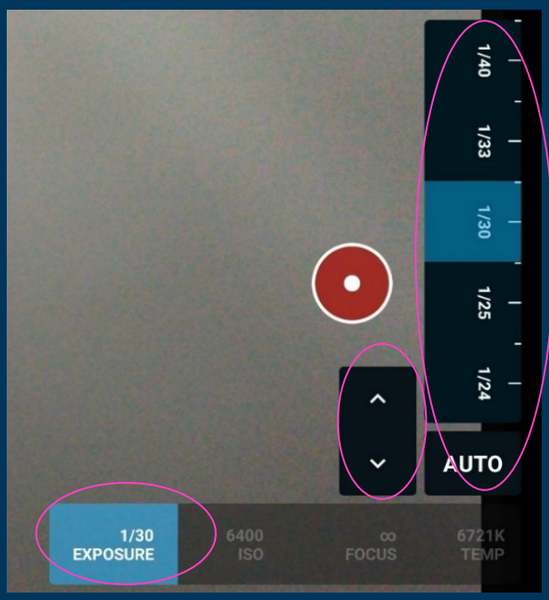
[4.1] Capture/Record Button: Tap to Capture with universal function across all modes. Starts/Ends recording sessions during Video modes. Press and Hold to initiate Buffer Recording mode (captures video only within RAM buffer space to avoid storage bottlenecks, however will only consume RAM/Memory Usage allocated until deleted, ending the recording, page 57 [4.27.1])

A Red button indicates a video mode. A Gray button indicates photo mode (will flash blue when pressed in Photo modes to indicate successful capture and will fill blue circle during night to indicate ongoing exposures





[4.1.1] Stop Recording + Pause/Resume Button: A Square inside red circle indicates an ongoing recording session. An icon nearby showing ⏸ / ▶ will appear for video modes during active sessions. Press to Pause or Resume the recording sequence for cuts without stopping the session. Press the Record button to end recording.

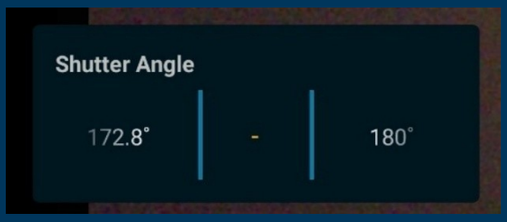


[4.2] Exposure/Shutter Speed: Shows the current Exposure Length/Shutter Speed in use. A white value indicates AUTO mode is active and the Camera2API automatic exposure driver is in control. Tap to open/close manual control interface.

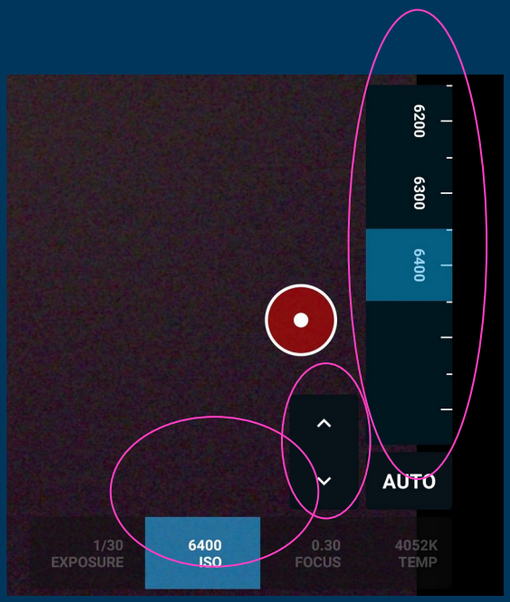
Holding the Exposure value will turn it yellow and manually lock it. Be advised, as of this current version, there is no manual Exposure Priority mode available, full auto or full manual only therefore.

Scroll up/down on slider values to manually adjust Exposure with linear precision. Tap up/down on the arrows to increase/decrease Exposure in fixed values. AUTO button restores Exposure selection back to automatic.

Note, if the exposure value turns red, it indicates that your currently selected shutter speed has gone below the framerate you've selected (will shoot below target FPS)



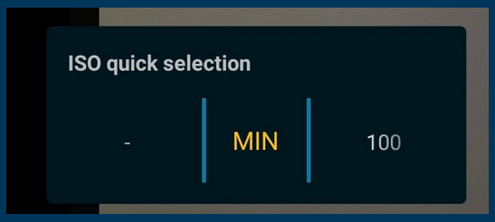
[4.2.1] Shutter Angle: Upon pressing on the Exposure value, a special control will appear on the bottom left side. Slide left and right to change Shutter Angle relative to current framerate selected. A "-" value indicates the mode is not engaged. Press AUTO to any selection



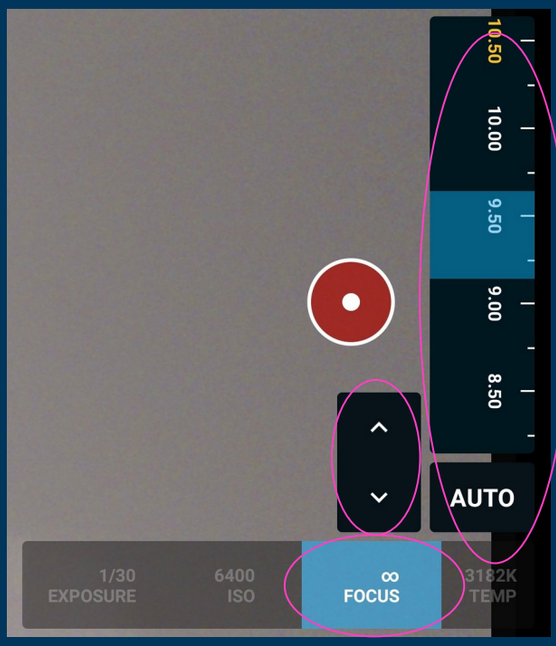
[4.3] ISO/Sensor Gain: Shows the current ISO in use. A white value indicates AUTO mode is active and the Camera2API automatic exposure driver is in control. Tap to open/close manual control interface.

Holding the ISO value will turn it yellow and manually lock it. Be advised, as of this current version, there is no manual Exposure Priority mode available, full auto or full manual only therefore.

Scroll up/down on slider values to manually adjust ISO with linear precision. Tap up/down on the arrows to increase/decrease ISO in +/- 100 increments. AUTO button restores ISO selection back to automatic



[4.3.1] ISO Quick Selection: Upon pressing on the ISO value, a special control will appear on the bottom left side. Slide left/right to change between ISO presets in +/- 1 stop increments. A “-“ value indicates the mode is not engaged. Press AUTO to undo any selection.

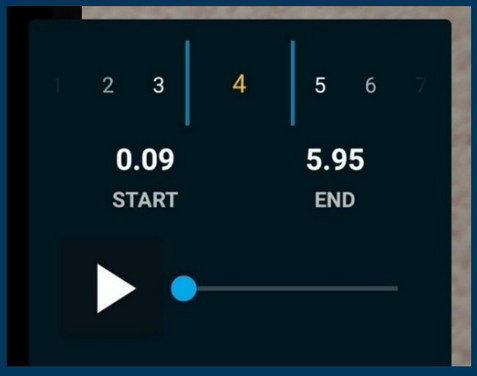


[4.4] Focus Ratio/Distance: Shows the current focus ratio, as defined by Camera2API. A lower value indicates farther focus distance and a higher value indicates closer focusing distance.
A white value indicates AUTO mode is active and the Camera2API autofocus driver is in control. Tap to open/close manual control interface.

Holding the Focus value will turn it yellow and manually lock it. A figure eight "∞" value indicates infinity focus range, however exercise caution as it may not always be the correct infinity calibration on every device.

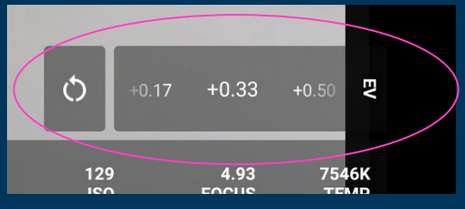
Yellow values on slider rack indicates extended focus ranges that may not be supported (out of spec), however may provide closer focusing capabilities than reported by Camera2API parameters.

Scroll up/down on slider values to manually adjust Focus ratio with linear precision. Tap up/down on the arrows to increase/decrease ratio in +/- 0.01 increments. AUTO button restores Focus back to automatic. Ratios are measured in diopters (1/meters = ratio), may be uncalibrated however.



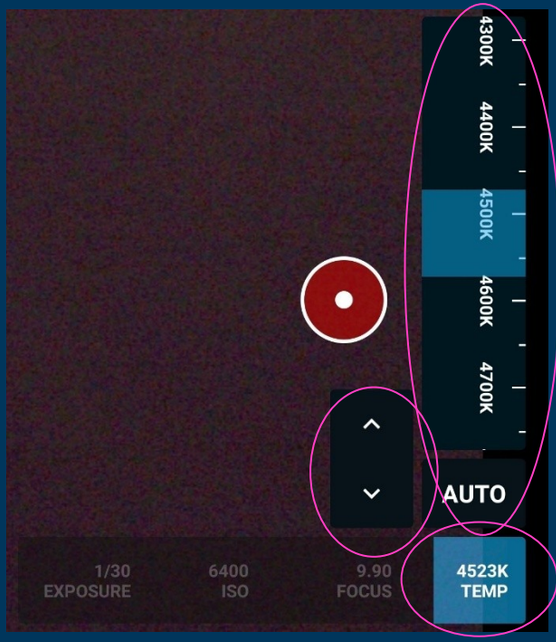
[4.4.1] Focusing Rack Controls: Upon pressing on the Focus value a special control will appear on the bottom left side providing focus racking controls. To set ranges of focus rack, first set a manual focus range. Long press on START or END values and the current manual Focus value will be defined accordingly. The top number indicates the duration (seconds) of the focus rack change. Press ▶ / ⏸ icon to begin/pause rack.

The blue dot will traverse the line to indicate progress of rack session (left side = START, right side = END). You may also hold the blue dot and manually swipe it left/right to change focus within the defined START and END ranges only as an alternative to the manual Focus slider controls.

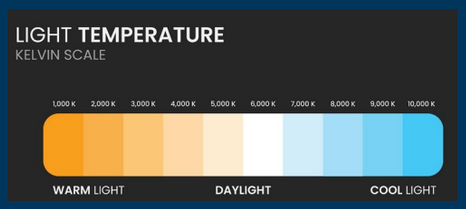


[4.5] Exposure Compensation: Sets the Exposure Value Compensation of the Auto exposure driver. Controls both ISO and Exposure/Shutter Speed to achieve desired Exposure offset. A +0.00 value corresponds to Off state. Slide numbers left/right to increase or decrease compensation in small increments.

Every +/- 1.00 change corresponds to a Full Exposure Stop (1 stop = doubling/halving of light intake). Press circular arrow icon, or return to +0.00 to undo or exit mode. Using manual exposure settings will null current selection



[4.6] Color Temperature: Shows the current color temperature applied in Kelvin temperature scale. A white value indicates AUTO mode is active and the Camera2API automatic color temperature driver is in control. Tap to open/close manual control interface.

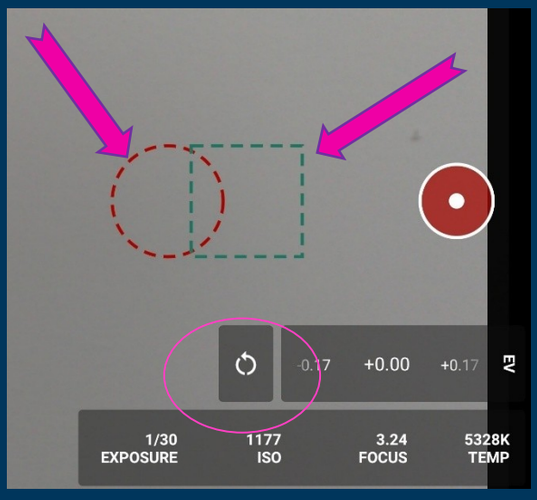


Holding the Temperature value will turn it yellow and lock it. Can be locked independently of exposure controls (doesn't impact automatic exposure driver).

Scroll up/down on slider to select Temperature with linear precision. Tap up/down on the arrows to increase/decrease Temperature in +/- 100 Kelvin increments.

AUTO button restores Temperature selection back to automatic

Does not impact MCRAW or DNG quality as only applied to EXIF data (the attached suggested setting) of RAW formats, however gets baked-in after encoding or compression into Photos, Direct Log videos or exported codec outputs.



[4.7] Spot Metering/Focusing Indicator: When you tap anywhere on the viewfinder, the app displays a circle for spot focusing and a square for spot metering.

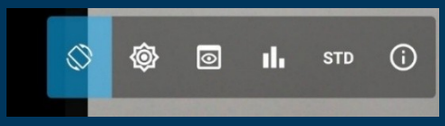
The circle will prioritize the autofocus driver area, while the square will prioritize the auto exposure driver for the area it covers as well.

A red value indicates that the settings are still being adjusted, but a green value means the app has successfully locked on to the target region. Changes within the Square/Circle areas may result in loss of lock-on, so color will switch back to red while the settings hunt again.

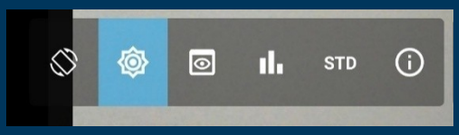
You can lock any acquired settings by pressing and holding the Exposure, ISO, or Focus values on the settings bar, and you can also drag the circle and square independently to different areas each.

Any changes via manual modes (except on Temperature) will exit respective Spot mode individually (Exposure/ISO change = Loss of Spot Metering and Focus change = Loss of Spot Focusing).

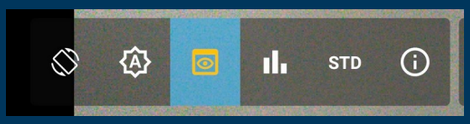
To exit Spot modes completely, simply press on circle arrow icon to undo both



[4.8] Display Rotation/Orientation: Allows for control of the display's current rotation preferences. Tap to alternate between Portrait (Sideway rectangle) and Landscape (Upward rectangle). Press and Hold to unlock rotation (follows system orientation preferences). A diagonal rectangle with arrows indicates unlocked rotation is active.



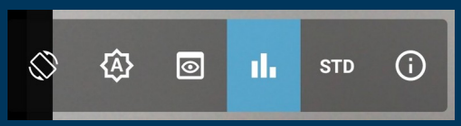
[4.9] Display Brightness: Provides control of the display brightness preferences. Tap to alternate between Full brightness (Indicated by white filled sun) or System/Phone setting (Indicated by Sun with letter A inside)



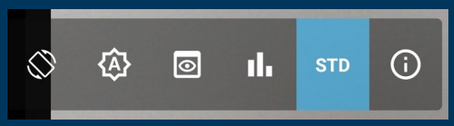
[4.10] Direct Preview Mode (Toggle): Enables/Disables Direct Preview Mode (DP Mode). Direct Preview allows you to see your current Capture settings real-time and bypasses Android viewfinder which can provide inaccurate scene previews that show incorrect image and is subject to ISP processing unlike the captured outputs. A yellow state indicates On, and a white state indicates Off.

Can also be enabled also via hardware button. Press and hold Vol + button to activate. Subsequent Vol + presses will then alternate between Direct Preview, Sensor Clipping and False Color overlays. You may also press Vol - button to exit.

For more information on how to operate Direct Preview Mode, jump to page 94 [4.63] Direct Preview (Usage and Interface)



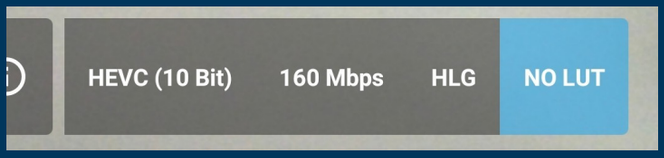
[4.11] Histogram Quick Toggle: Provides quick toggle to Show/Hide the RAW Histogram (Appears below) from user interface. Tap to toggle. White = Off (Hide Histogram), Yellow = On (Show Histogram)



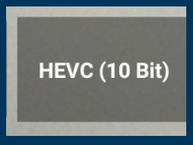
[4.12] Standard/Maximum Preview Resolution Toggle: Provides ability to quickly change viewfinder resolution from current selection to maximum resolution available. Does not impact capture quality however may impact performance. White STD letters indicate selected viewfinder resolution is active, red MAX letters show Maximum viewfinder resolution has been enabled (jump to pages 48 & 49 for more about viewfinder resolution and how to enable this toggle as otherwise it will not appear on the bar)

[4.13] About Us: Opens screen showing App details including Contact Us and Support information, Community groups ands additional app resources.

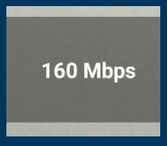
[4.14] Manage Videos (MCRAW Manager): Appears while using RAW VIDEO, TIMELAPSE and BURST modes. Tap to open the app's built-in MCRAW handler (MotionCam RAW video files). For more info on using the MCRAW Manager, jump to page 98.



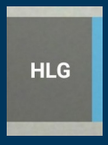
[4.15] Direct Log Video Settings Bar: Appears only during Direct Log Video mode. Shows settings bar with brief summary of the current Direct Log encoding setup.



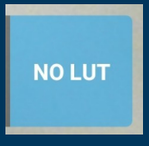
[4.15.1] Codec (Color Depth): Indicates the current encoding codec selection and its color depth inside parenthesis.



[4.15.2] Bitrate: Indicates the current bitrate or encoder quality level selected. Shows numerical Mbps value or letter name depending on codec (Eg. Will show "**LT**" for ProRes LT encoding grade).

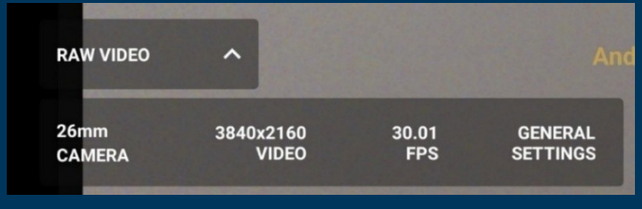


[4.15.3] Transfer Function: Indicates the current Transfer Function in use (Eg. HLG, PQ, Apple Log, Davinci, etc.) Does not show Color Space however.

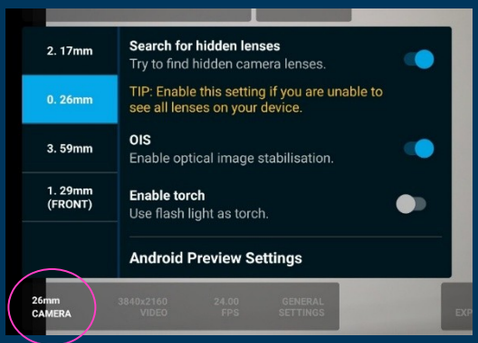


[4.15.4] LUT Applied: Indicates the current LUT selection to be baked-in into Direct Log Video outputs. NO LUT indicates no active LUT is in use.

Tap on any of these settings to open the Direct Log settings sub-menu. Press and hold LUT indicator to quickly enable/disable any recent LUT applied. Jump to page 67 [4.36] for the complete Direct Log Video Settings sub-menu details.



[4.16] Camera & Display Controls: Control bar with options to adjust parameters such as Camera Lens, framerates, RAW stream, cropping, as well as viewfinder. Swipe left/right on Camera ID to quickly swap between lenses!

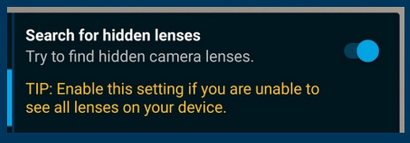


[4.17] Lens & Viewfinder: Tap to open menu for controls of Lens (Camera ID) in use, adjust additional capture aspects, as well as modify viewfinder parameters.

[4.17.1] Camera Lens Selection: Shows the lenses listed as available by the device. Not all lenses shown may work. A blue rectangle indicates the active lens currently selected and in-use. Tap on any lens to switch to it. Selfie lenses will be indicated by (FRONT).

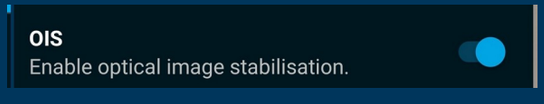
Lenses appear In 35mm equivalent focal length (eg. 24mm = x1 zoom). Please note that the focal length shown may not be exact and is reported as per Camera2API information.

An individual # value indicates a physical ID, and a #/# indicates a logical ID (Eg “2. 26mm” would indicate Hardware ID 2, and “0/2. 26mm” would indicate a Logical ID 0 using physical lens 2). Although uncommon, sometimes duplicate IDs may provide access to different framerates or sensor modes than their main ID, so we encourage you to explore them at least once.

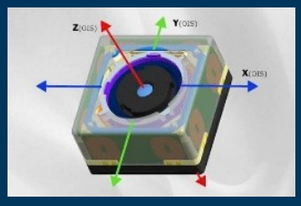


[4.17.2] Search for Hidden Lenses: Enable this setting to force a scan for lenses that the device is not reporting. If you’ve ever seen a horror movie, it’s like when a character asks “who’s there?” before getting killed, except that in this case some Camera IDs may sometimes reply like “Oh, you got me!” then reveal themselves.

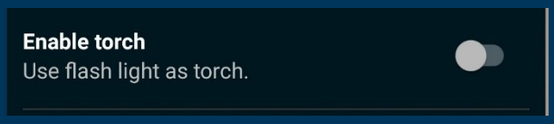
This setting can sometimes find hidden lenses that should appear but don’t. Always try to use it in the event you can’t find a Camera module, however be mindful it can also find broken or duplicate IDs (it’s also not guaranteed to work as OEMs may fully block Auxiliary lens access in some cases)



[4.17.3] OIS (Optical Image Stabilization): Toggle to Enable/Disable Optical Image Stabilization in supported lenses. Moves camera eye module as anti-shake mechanism.

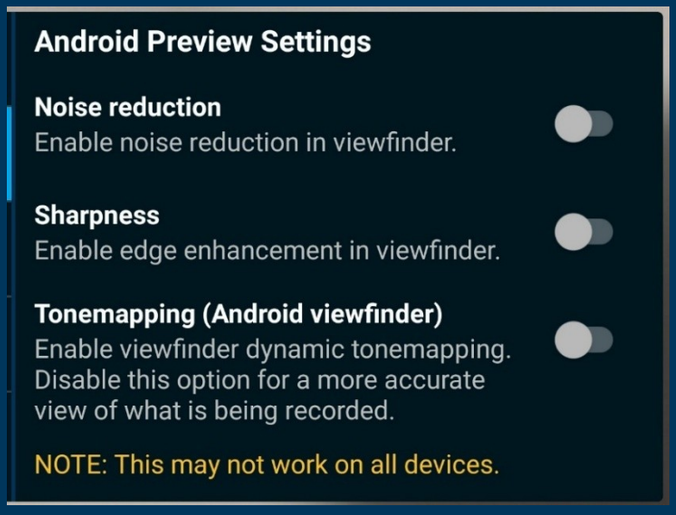


Helps photo and video stability, however should remain off if you plan to use Gyroflow as it will interfere with it. Can potentially hamper gimbal performance as well so use with care.



[4.17.4] Enable Torch/Flashlight: Toggle to turn phone torch (flashlight) on and off in Fill mode to provide additional video illumination. Note: may possibly not work with all lenses.

[4.18] Android Preview Settings: Allows for adjustments to the Android Preview image quality (Jump to page 95 [4.64] for more info on the viewfinder types). These settings do not impact capture quality.

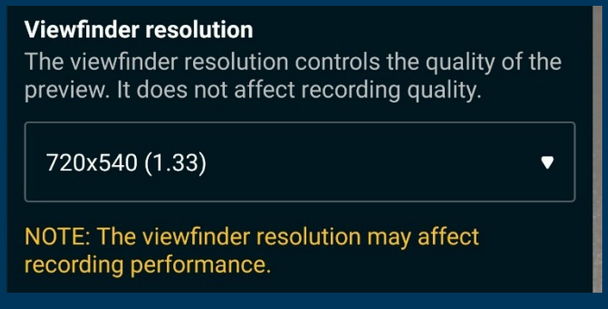


[4.18.1] Noise Reduction: Toggle off to reduce/disable viewfinder noise reduction

[4.18.2] Sharpness: Toggle off to disable viewfinder Sharpening (Edge Enhancement)

[4.18.3] Tonemapping (Android viewfinder): Toggle off to disable preview Dynamic Tonemapping that may show drastically different light and color reproduction compared to capture quality.

Having these options disabled is desirable for more accurate previewing, however do note it can sometimes cause visual glitches or malfunctions in some devices and may not work (preview freezing, screen halves showing differently, broken viewfinder images, and/or app crashes).

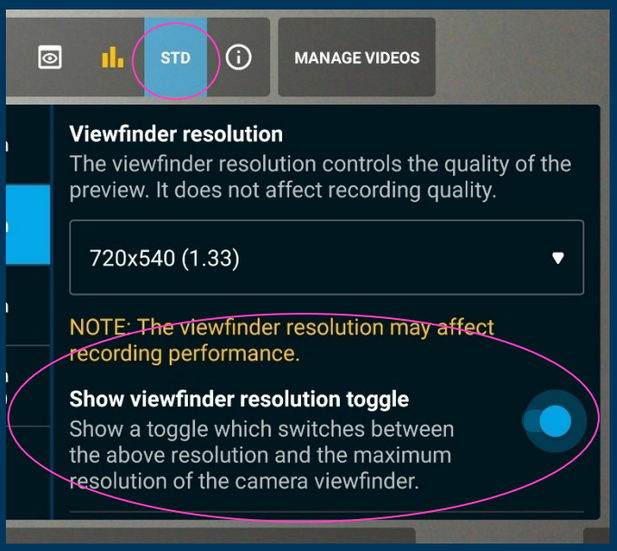


[4.18.4] Viewfinder Resolution: Provides control of the resolution rendered for the viewfinder preview.

Does not impact capture quality however will impact capture performance (a 480p preview is far easier on the CPU vs a 1080p/4K UHD preview which may heavily tax the system performance.

You can quickly toggle between this setting and maximum resolution using the STD/MAX toggle (next page).

NOTE: the ratio of the viewfinder will be shown as a number in parenthesis, in the above example 540 x 1.33 = 720 as an example. BEWARE – If ratio chosen can't fit the current capture resolution correctly, you may possibly warp the viewfinder or no longer be able to see the outermost edges on viewfinder (eg. , however this will not impact capture quality.

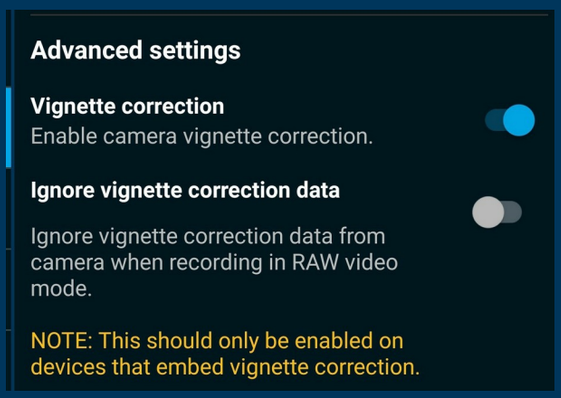


[4.18.5] Show Viewfinder Resolution Toggle: When enabled, will show an option on the top quick settings bar to quickly switch between the current *Viewfinder Resolution* (STD) to the Maximum resolution (MAX) listed by Camera2API.

Be mindful of using MAX viewfinder resolution, although it will not impact capture quality, it will degrade device performance due to rendering a high amount of details for the preview.

Best used to quickly hone into a perfect focus for tricky scenes with a lot of details. Usage of this mode during captures is not recommended.

[4.19] Advanced Settings: Below this area, additional controls that will affect capture quality and lens list are shown and is best left as-is for the most part.

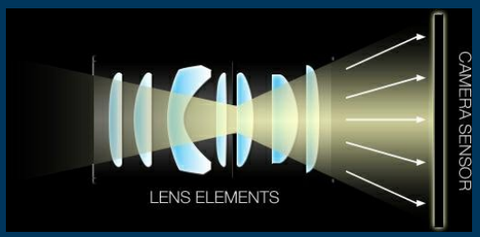


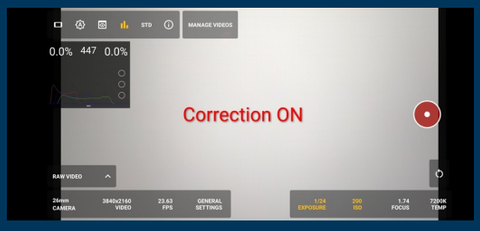
This area should not be modified unless you are having issues or you have duplicate/hidden lenses that may either need to be added manually or removed if they're broken/inoperable

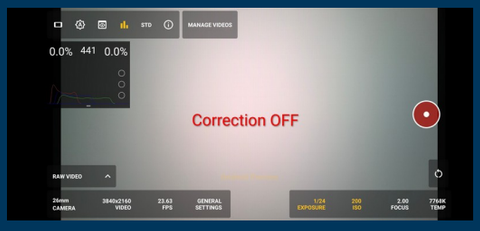
[4.19.1] Vignette Correction: This setting allows you to control the digital correction of the natural vignette that all lenses produce. Toggle On to enable and Off to disable.

Vignetting is a gradual darkening towards the edges of an image, caused by the physics of a circular lens where the corners of the sensor receive less light.

Toggling this setting On applies a digital offset to correct this difference by brightening the corners with a counter gradient. However, be aware that this may also amplify noise in those areas, and in low light, it could introduce a noticeable purple circular pattern resulting from lifting overly dark areas. Note: In rare instances, lenses can exhibit a center pinkish hue with VC



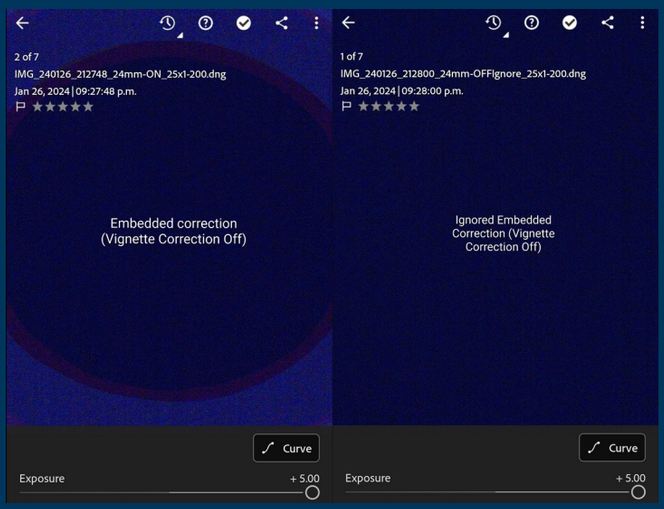




[4.19.2] Ignore Vignette Correction Data: This setting should only be used as a reactive measure, but it's important to understand why. Toggling it on will completely ignore any vignette correction data that the device may generate during capture — see prior page for vignette correction explanation.

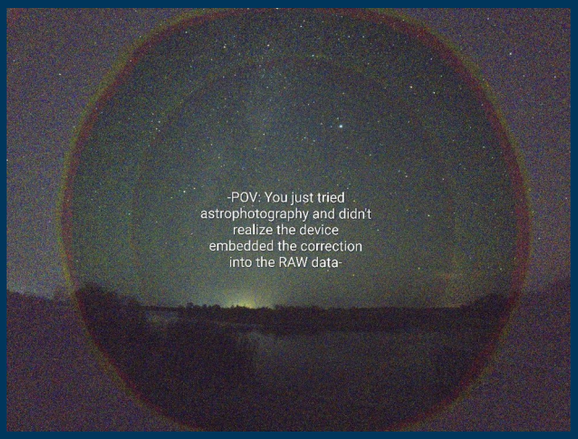
Normal correction works by providing an additional bundle of data with the correction offset info in the RAW file EXIF. This is like when you get fries at McDonald's with a packet of salt—you have the option to use it or not.

The problem is, some devices bake this correction right into the file and don’t give you a choice (I’m looking at you, Samsung and Sony, as a personally traumatized user). It’s like if they added a ton of salt on your fries before you even got them.



While this might sound convenient, it means you're stuck with irreversibly altered data that can have more noise, artifacts (Square looking patches, altered colors in borders of lens), and even purple rings due to the device's sometimes-poor correction.

Another side effect is that embedded data + Correction On can double correct the vignetting, further aggravating anomalies. It’s like adding a SECOND pack of salt to an already salty abomination, only a true brine enthusiast would enjoy it.



Using this option purges the embedded correction before the device has a chance to bake it in, but it also purges the app's own correction data, which is usually a bad idea on normally functioning devices. So, if you're getting weird artifacts, try this option. Otherwise, steer clear.

A NOTE FOR ADVANCED USERS: Vignette Correction contains corrections as well for more than just the gradient mentioned. Due to optical issues with the IR filter elements on some devices, a disabled vignette can produce a radial color shift that looks like a red patch whenever Vignette Correction is off. However newer devices are less affected by this. For solutions if you’re intending to fix this, please jump onto the MotionCam Discord as this subject can get quite complex and beyond the scope of this guide (Heavily dependent on NLE’s and advanced tools).

[4.19.3] Camera Lens Visibility: This section displays the complete list of lenses available to the app, which are shown on the left side of the menu under Camera Lens Selection.

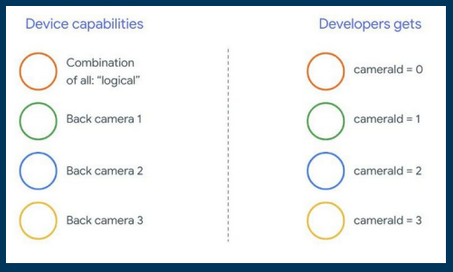
The list you see will vary widely per device, as some camera IDs are duplicates, while others are Logical IDs or Hardware IDs. We encourage you to experiment and confirm which work and which don't.

Toggling the Search For Hidden Lenses (Page 47 [4.17.2]) option will generally reveal additional IDs, but their functionality is never guaranteed.

A blue checked box indicates the lens will be shown in the selection to the left. Tap the box to uncheck then and hide them from the list, or vice versa (after testing them, hide the broken IDs, only show the good or useful ones!)

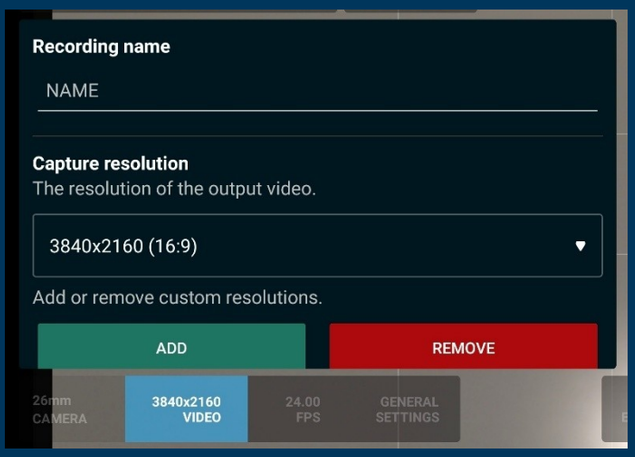
The reason for this confusing list lies in the different ID types. Hardware IDs are the easiest to understand: they are the actual physical lens itself. For instance, if you have a Main, Telephoto, and Ultrawide lens, their Hardware IDs would be 2, 3, and 4, respectively. However, OEMs can arbitrarily name these, so take this only as a conceptual example.

The complexity often comes from Logical IDs, which are denoted by a #/# designation. A Logical ID is a virtual parent ID that acts as an interface. When you ask a Logical ID (say, ID 0) to use a specific zoom length, the phone gets full control of all the lenses and selects the most suitable underlying lens for you.



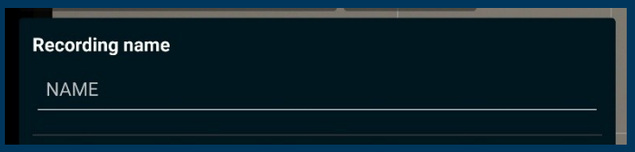
For example, if you have three IDs, using a Logical ID 0 means you are summoning them all on demand; asking it to use the telephoto means you are using ID 0/3. While you don’t need to know this for MotionCam (the app forces them, regardless), understanding this gives you the reasoning behind the madness. If you ever used the stock camera app then zoomed in, yet the telephoto lens didn’t engage, this is why (it’s a feature, not a bug).

Finally, some OEMs introduce Sensor Mode IDs, which is a form of Camera2API malpractice. This happens when manufacturers go full IQ200 and map specific functions to their own hidden ID. This is a worst-case scenario, as it means a Hardware ID 2 might have a hidden 60fps mode or RAW access only available on a separate duplicate ID 11. You are forced to switch to that "fake" ID to access the function. We’re looking at you, Samsung…

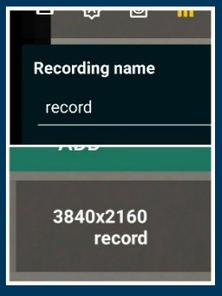


[4.20] Capture Configuration: Shows the current Capture Resolution (Crop) and the name of your video file (Defaults as VIDEO unless a name is added). Appears at the bottom left settings bar showing the Capture and Display settings. Tap on it to open the Capture Configuration Menu.

Here, you can adjust multiple settings including the famous RAW streams we talked about plenty, as well as you can select the capture crop, set file paths as well as mkre options involving footage capture.



[4.21] Recording Name: Use this setting to add a custom file extension name to your captures. Simply press NAME to open the keyboard and type your desired name, which will persist until manually removed.



This custom name will also be reflected on the bottom-left bar below the resolution and applies to both RAW and Direct Log outputs.

The app's file name is constructed using a clear nomenclature: it includes the Year (YY), Month (MM), Date (DD), followed by the Hour (HH), Minutes (MM), and Seconds (SS). This timestamp is then followed by your custom file name, and finally, the Focal length of the lens (#mm).

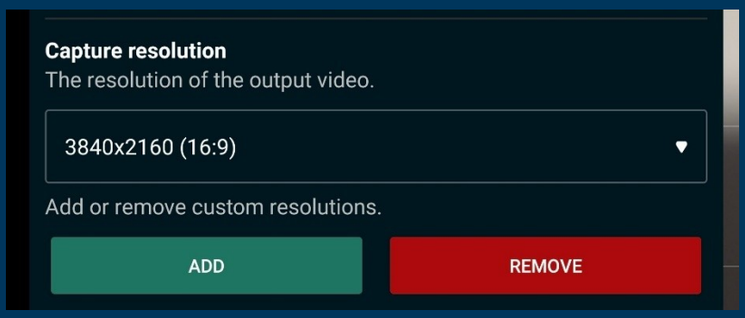
For Direct Log mode, the app automatically adds two extra extensions to the file name to specify the Transfer Function (e.g., Samsung Log) and Color Space (e.g., Rec.2020).

See the examples below for how the files would appear relative to the mode in use…

MCRAW video: YYMMDD_HHMMSS_NAME_#mm.mcraw

Direct Log Video: YYMMDD_HHMMSS_NAME_#mm_Transfer Function_Color Space.format

NOTE: the .mcraw files will not show on filter explorers if under private storage and must be first transferred outside of it. More on page 57 [4.27.2] & page 105 [4.70.3].



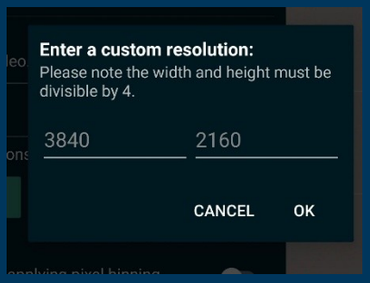
[4.22] Capture Resolution: Use this setting to define the crop resolution for the current sensor stream, which is distinct from the overall Sensor Mode or RAW stream selection.

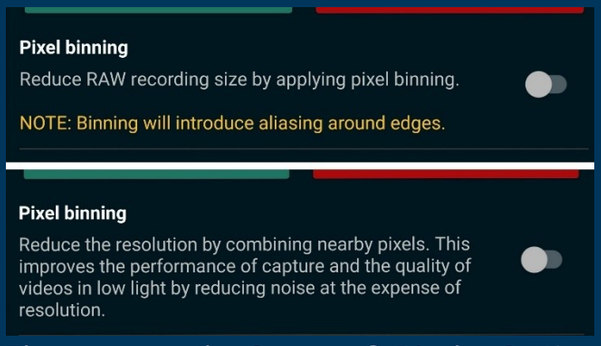
This crop essentially tells the app what final resolution you want to extract from the available stream.

For instance, from a 4000x3000 stream, a setting of 3840x2160 (16:9) will discard all pixels outside that range.

Be aware that a higher resolution demands more device performance, and resolutions exceeding 3840x2160 can sometimes interfere with 10-bit HEVC encoders when rendering compressed video (refer to the FAQs for details).

To use a different crop preset, press the rectangle box showing the current one. You can also create your own by pressing ADD or delete custom ones by selecting and pressing REMOVE.





[4.23] Pixel Binning: Enables resolution binning, which reduces the resolution of the currently selected capture resolution by a 4:1 ratio without cropping (eg. turning a 4K UHD selection into 1080 FHD).

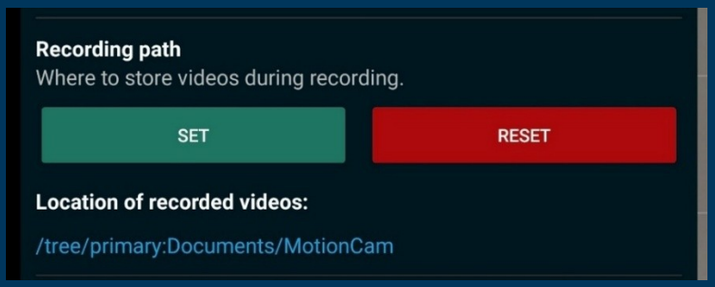
You should use this option primarily to save storage space and reduce the performance tax on your device.

The app uses the Capture & Display Settings button as a visual indicator: a yellow value means binning is active, while a white value means no binning is in use. It's crucial to note that binning works differently depending on the mode:

RAW Video: Binning applied exclusively to save space via lower resolution, but does not resize the data. This can introduce some aliasing (jagged textures noted around fine edges/lines) and impacts micro details, but it drastically reduces the file sizes.

Direct Log Video: The app instead uses super sampling during encoding that leverages GPU accelerated resizing of the data; meaning that as an additional benefit, at the cost of resolution, you get an improvement in low-light performance rather than just saving space.

The description for Pixel Binning will change depending on the mode in use to advise of this.



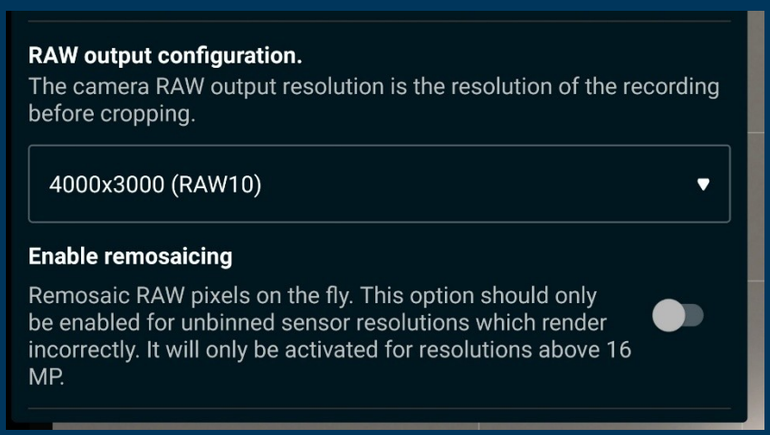
[4.24] Recording Path: This option will allow you to designate the location or folder on your device to use for saving your captured outputs and/or designate External Storage locations like SSDs and such.

Press SET to open the Android file explorer. Any selected location will be used afterwards to store captures.

This option can be undone by pressing the RESET Button.

Any active location in use will be displayed on the bottom part in blue (Location of recorded videos:) and you can observe the file path chosen.

The default location unless set otherwise will be Internal Storage/Documents/MotionCam



[4.25] RAW Output Configuration: This is one of the most critical functions to understand in MotionCam: it allows you to select the specific RAW stream used by the active lens.

Unlike most camera apps, which request a simple video stream (e.g., 3840x2160 30fps from a YUV/Video stream), MotionCam asks for whatever RAW settings the camera can provide. *This is not the final video crop, but the actual sensor output mode.*

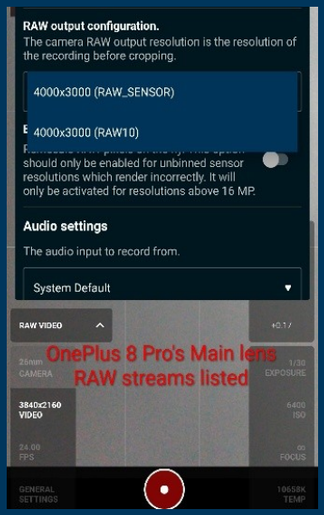
The options available will differ drastically by device. If a device exposes a 50MP RAW stream, you can then crop that for an 8K mode; if it exposes 12-bit RAWs (often designated as RAW12), this is where you enable it. Besides the resolution, you will also see a designation of the stream's container in parentheses.

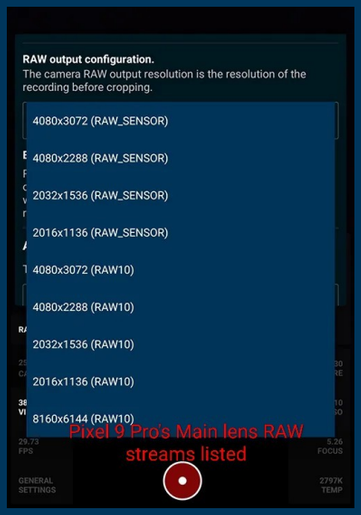
Here is a breakdown of the various stream options you may encounter…

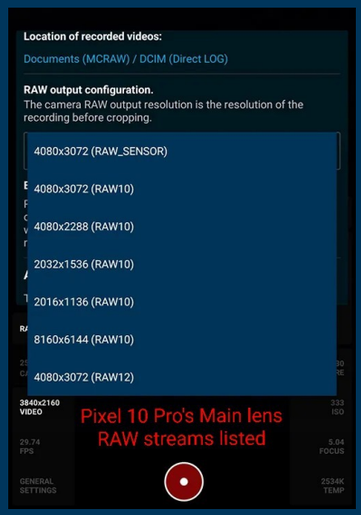
WIDTHxHEIGHT (CONTAINER)

Example…

4000x3000 (RAW10)







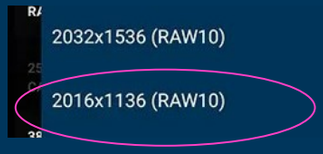
[4.25.1] STREAM RESOLUTION – WIDTHxHEIGHT

OPEN GATE STREAMS: Open Gate implies the full sensor surface area is being used—you are not cropping anything and are getting the maximum area to shoot with. These are the most common streams for MotionCam, often appearing as the only choice. They typically come in 12MP resolution (around 4000x3000, with minor pixel variances).

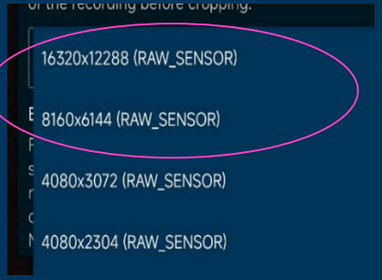
OPEN GATE CROP STREAMS: These streams generally offer around 8.9MP, sufficient for 4K video, but achieve this by performing a full sensor area crop, often to a 16:9 aspect ratio (e.g., cropping a 4000x3000 area to roughly 4000x2200). The key benefit of these cropped streams is that they often unlock higher framerate access. For example, if an Open Gate stream maxes out at 30fps, a cropped stream might allow for 60fps or higher.

OPEN GATE BINNED STREAMS: These are similar to standard Open Gate streams, but they utilize native binning offered by the device itself to reduce resolution. This conserves both performance and storage. While the app can perform binning for you, some devices offer this natively directly from the stream.

OPEN GATE BINNED CROP STREAMS: These streams are a rare combination of binned and cropped modes, often seen on Google Pixel devices. They are powerful because they tend to allow for ultra-high framerate access (120-240fps), though this comes at the expense of both resolution and a cropped stream area.



UNBINNED OPEN GATE STREAMS: These are rare stream types, also known as the full resolution modes. These are the 48/50MP/200MP RAW modes possible with Quad/Nona Bayer sensors. Use these streams to access the highest possible resolution modes (like 8K), but beware: their performance demand is ultra-high as the app gets +48MP worth of RAW data. These modes often require root mods.



[4.25.2] RAW CONTAINERS (RAW##)

This option designates the bit container used for the RAW stream. Understanding the available containers is vital, as using an appropriately sized container will reduce storage and the performance toll on your device.

RAW_SENSOR (RAW16)	16-bit container	The native default, sometimes your only option. **Note:** This does not mean the device shoots 16-bit; it's a large container padded with unused data if your sensor shoots in lower color depth (eg. 10-bit)
RAW10	10-bit container	Preferred choice over RAW_SENSOR if available should your device shoot in 10-bit only, as it significantly reduces storage and performance toll on the device during capture
RAW12	12-bit container	Rare. Appears if your device natively supports 12-bit ADC or DCG modes.
RAW14	14-bit container	A **unicorn**. Extremely rare and would indicate access to DCG16 (The Holy Grail). Never expect to see this.

Sizing Your ‘Cup’

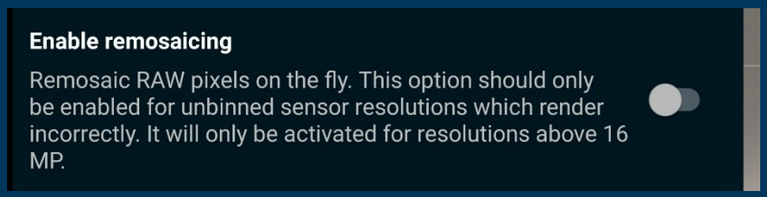
A common question is the difference between RAW10 and RAW_SENSOR. Think of it like this:

Pretend you have a 0.5 Liter Cup (RAW10) and a 0.5 Liter serving of soda (10-bit data)—they are sized perfectly. A 2 Liter Cup (RAW_SENSOR/RAW16) would be great if you had 2 Liters of soda, but if you only have 0.5 Liters to fill it up with, you're now carrying an unnecessarily large cup.

In short, size accordingly. While MotionCam will compress the file, using a container larger than necessary will still tax performance more as the app has to handle more incoming data from the container being padded. If your device shoots 12-bit or 14-bit and only offers RAW_SENSOR, you must use it to harness the higher color depths; otherwise, stick to the smallest container that fits your data. An analogy of RAW10 vs RAW_SENSOR with a 10-bit ADC sensor would be this…

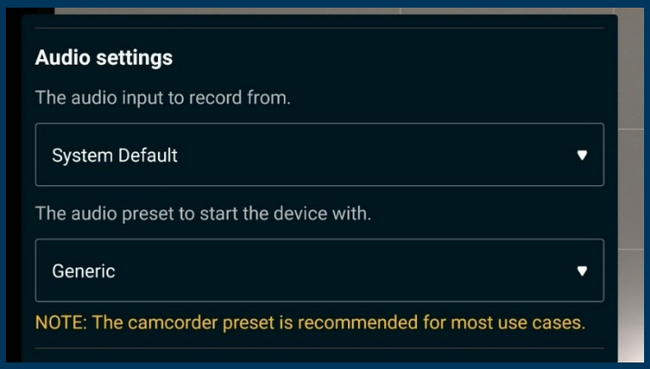
ABCDEFGHIJ000000 vs *ABCDEFGHIJ*

The above example is an easy way to understand the downside of RAW_SENSOR, more space for the same amount of data. For higher color depth, RAW12/14 container options are ideal almost never available.



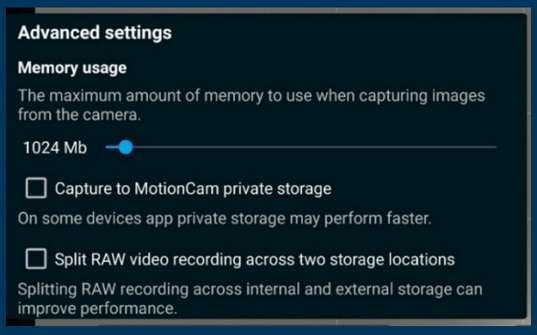
[4.25.3] Enable Remosaicing: This option should only be used if after selecting an Unbinned stream, the image appear purple/magenta and the colors look broken.

Applies a custom app ‘remosaicing’ algorithm that attempts to re-organize the pixel color array into a correct RGGB grid. Only activates once toggled and when a resolution above 16MP is detected.



[4.26] Audio Settings: This area allows you to configure both the audio/microphone input source as well as their preset (the audio processing and filtering, such as background noise reduction applied by the device)

Use the top option to view any external mics connected. NOTE: Bluetooth wireless mics do not work, only wired mics are functional as of now.



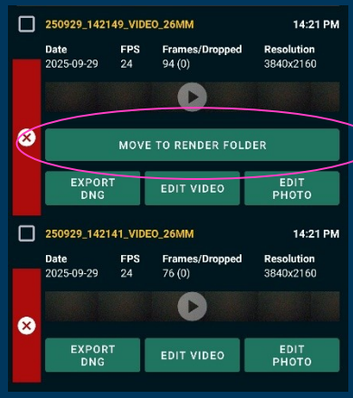
[4.27] Advanced Settings (RAW Capture): This area should be left alone unless you are attempting to extract more performance from the device, whether due to memory or IO bottlenecks you've encountered, or pushing extreme capture settings.

This settig will only appear in RAW video modes!

[4.27.1] Memory Usage: Allows you to increase/decrease via a slider the RAM allocation that MotionCam will use for RAM buffer that is used to store incoming RAW data. A higher value could increase performance during captures, however it may also impact system performance if you request too much.

An overly high value may result in app crashes or noticeably lower capture performance (may also cause longer app boot up times), use with caution.

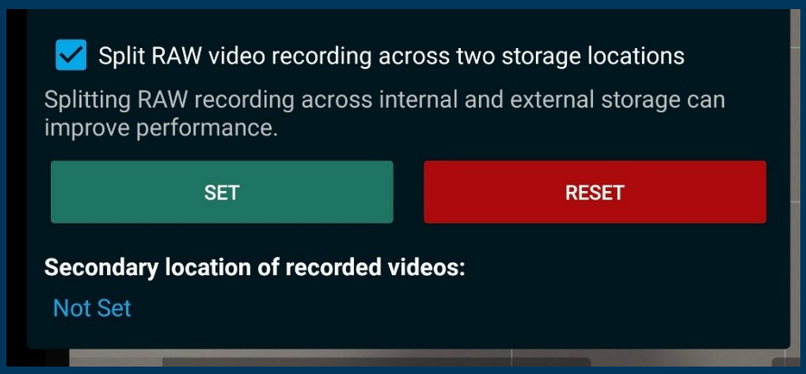
[4.27.2] Capture to MotionCam Private Storage: This setting, which applies only to MCRAW captures, determines whether files are written to your designated folder or the app's 'private' folder (inaccessible at the root/system level).



This option exists to eliminate potential I/O bottlenecks on devices that restrict file write speeds to the standard user-designated folders.

IMPORTANT: Checking this box ON, although it may eliminate bottlenecks, should only be used when absolutely necessary.

If files are written to Private Storage, you must manually transfer them out using the MOVE TO RENDER FOLDER option (circled in purple) in the MCRAW Manager. Failing to do so means the files will not appear in a standard file explorer, and deleting the app will wipe your recordings. See page ## for more info.



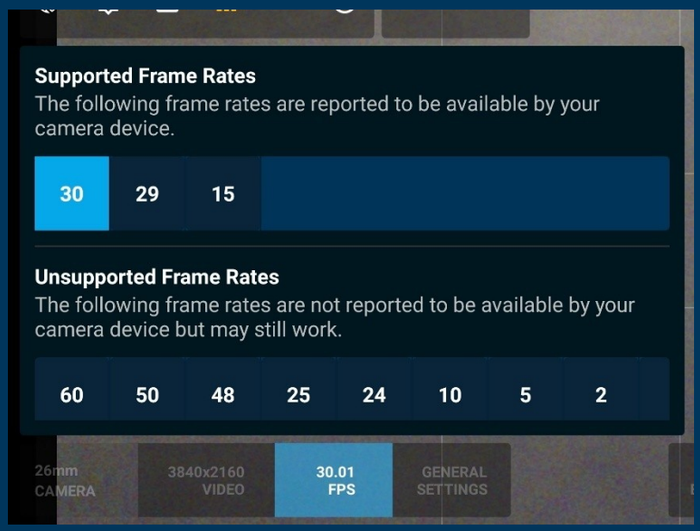
[4.27.3] Split RAW Video Recording Across Two Storage Locations: This option allows you to use two different storage locations (e.g., internal and external) to alleviate write speed issues that may occur with slower internal storage modules.

If checked, an additional section will appear below to set the secondary external location and show the designated path. Uncheck the box to disable the feature, or press RESET to erase the path.

Do note that a designated path is required for this option to be active, even if the box is checked. While this feature was a lifesaver with older storage (pre-UFS 3.1), it’s since become less necessary, except for lower end models.

⚠️ WARNING: Seriously, Don't Do This!

Haha! So you’ve decided to designate an internal location as the secondary one as well? Nice try! It won't be faster and will actually tank performance as you've now made the app write to multiple separate internal locations simultaneously. This will impact performance heavily, so please don't do it... Don’t ask how I know this…



[4.28] Framerate: This area on the Camera & Display Controls bar shows the current framerate being outputted in real-time. It displays with hundredth decimal precision (e.g., 000.00fps).

The value may fluctuate if the device cannot keep a steady constant, known as a Variable Framerate (VFR), but a Constant Framerate (CFR) will remain perfectly still. The app will request an exact number but some devices may give slightly variant ones.

As an example, if you select 30fps, the lens may instead give 30.01, or 30.10, or even 29.96. These are arbitrary examples and the values may vary strictly per each lens and per device, however it's important to account for them in post to avoid audio drifting.

Crucially, if the selected shutter speed falls below your target framerate, the output will change to reflect the current lower rate. For example, selecting a 30fps slot that gives 30.10 with a 1/30s will turn it red due to the slightly lower shutter vs framerate. Additionally, an example shutter speed of 1/16s will cap your FPS counter at ±16.00fps; the shutter speed value will turn red to warn you of these potential framerate overrides.

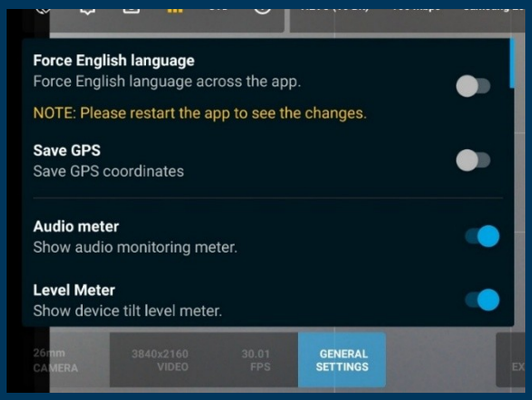
To open the menu and view the available capture options, tap the framerate number on the bottom-left settings bar. The listed options shown will vary by device and are categorized as follows:

[4.28.1] Supported Frame Rates: These are the framerates reported by the device's Camera2API as 'officially' supported. They should work unless the OEM has introduced an issue.

[4.28.2] Unsupported Frame Rates: These are framerates that the device did not explicitly list as available. This omission does not mean they won't work - it simply means the device didn't mention them rather than outright state they won't run. While compatibility is not guaranteed (and some may cause crashes), you are free to try them. Note: You may need a cropped RAW stream to access higher framerates (see page 55 and 56 and the full section [4.25] for details).

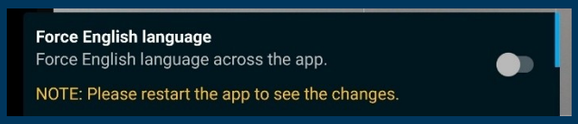
Pro Tip: *Hidden System Priority*

If you try an unsupported framerate that doesn't fully work yet it doesn't crash (e.g., you select 60fps slot but it only runs at 30fps), it's still a win! You've gained a hidden advantage: the higher framerate slot forces the device to prioritize capture resources. This can squeeze more performance from the system and increase the performance ceiling of your capture.



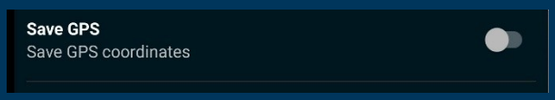
[4.29] General Settings: Tap on this setting in the Camera & Display Controls bar to open up the General Settings menu.

This menu provides options in relation to overall user interface and viewing guides, as well as additional overlay and tool controls.

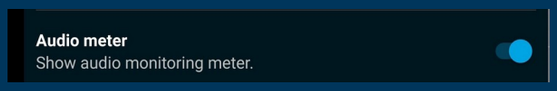


[4.29.1] Force English Language: This setting can be used to force English as the language selection when running MotionCam. If disabled, the app will attempt to use the current system language instead if already added under the supported languages list. If you change this setting, ensure you restart the app in order to apply it.

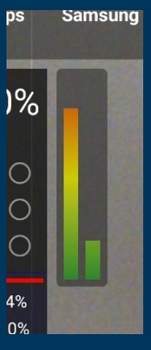
Here's the current list of supported languages: SA Arabic (ar) • DE German (de) • ES Spanish (es) • FR French (fr) • IT Italian (it) • JP Japanese (ja) • KR Korean (ko) • PT Portuguese (pt) • RO Romanian (ro) • UA Ukrainian (uk) • CN Chinese (zh)



[4.29.2] Save GPS: Enable this setting in order to embed GPS coordinates/Location data onto captured MCRAW and DNG files' EXIF information. Please note this option only works if the device location services are already enabled at the time of capture (the app itself can't turn on location services if not on)

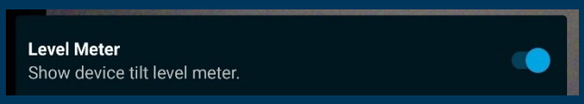


[4.29.3] Audio Meter: Enable this setting to display a real-time Audio Meter on the upper-left side of the user interface, positioned just to the right of the Histogram if that is also enabled.



The meter uses separate bars to indicate the signal strength for both the left and right channels in a stereo arrangement. A lack of bars means no audio is being received, while the presence of a short, green bar indicates a faint audio noise level. As the audio signal increases, the bar's gradient will change towards red to signify a louder noise.

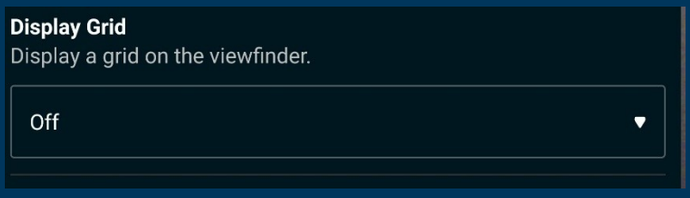
If the bar maximizes out, the signal has blown out or clipped, meaning you have exceeded the dynamic range of the microphone currently in use.



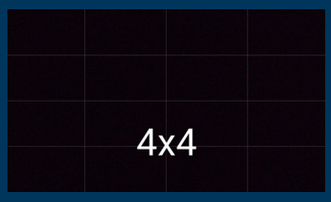
[4.29.4] Level Meter: This setting enables an orientation level meter that will appear on the center of the viewfinder.

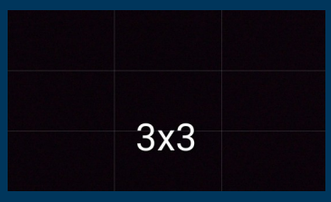
Use this level meter to determine the tilt of your device in degrees. Positive values indicate left tilt and negative values indicate a right tilt with 0.00° being perfectly level relative to the horizon. The Orientation scale used is to the right.

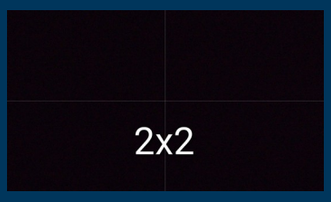
The meter has a central line that tilts with the device showing the degree orientation, and changes with color starting from green to red as the tilt increases more. A left and right squares will perfectly align whenever leveled and the color will become dimmer to indicate a good level orientation. See the blow examples for how it will look.

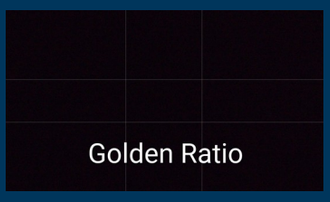


[4.29.5] Display Grid: This setting allows you to enable a display Grid overlay on the UI/Viewfinder. Tap on the rectangle area to open the available selection (Set Off to Disable). Here's the current options…



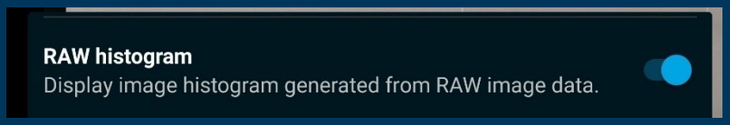






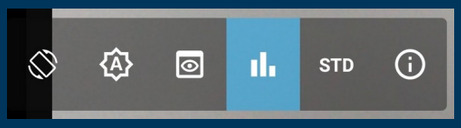
[4.29.6] Lock Exposure/Focus/White Balance on Record: Toggling these options on will make the app automatically lock the current settings of their respective categories once you begin recording,

You can undo the manual locks triggered as per usual once recording, however take note that these toggles will persist even after app is closed.



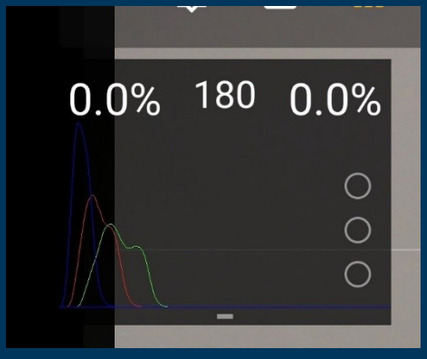
[4.30] RAW Histogram: This toggle will turn On/Off the RAW Histogram on the top left of the viewfinder.

(Note: This option can also be toggled on the top left Histogram Quick Toggle button)



The histogram is one of the most powerful tools provided by MotionCam. It is strongly encouraged to master it in order to fully extract the app's imaging potential.

Practically all other app histograms provide you a JPEG histogram, meaning that you get the information data that would occur after saving to a JPEG. This is problematic as JPEGs are lossy containers operating on an 8-bit color depth, hence not able to properly contain the deep highlight and shadow information that would present itself in a full +10-bit RAW image as most modern sensors will produce.

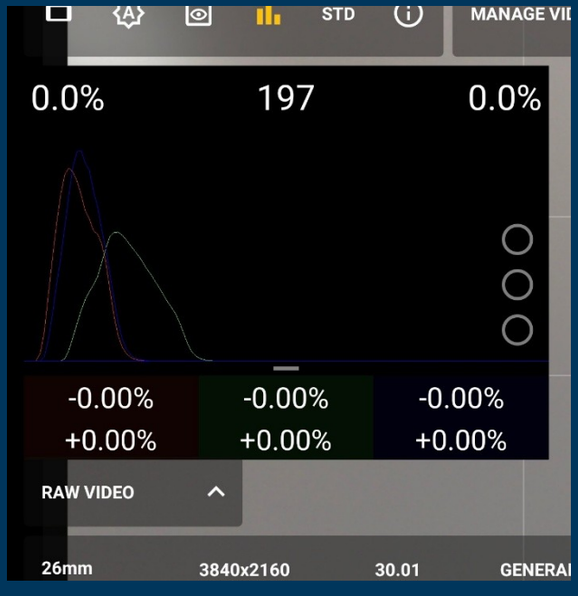


What this translates to is having to guesstimate what the limits of your dynamic range are, without accurate precision (eg. JPEG histogram may show clipping, but the RAW data may still have plenty of space left to keep pushing further).

With MotionCam, you obtain the literal sensor data as would be received in a RAW image container. What this means for you is that the accuracy levels provided by the MotionCam histogram are second to none.

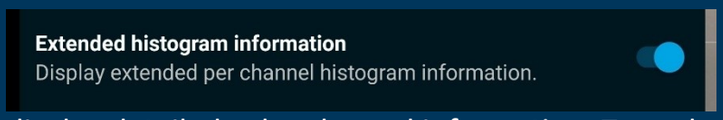
Due to the many complex aspects of the histogram and the data it provides to the user, a full comprehensive section on its usage isn't present in the manual, however it may be added in the future (jump to the next page if you want to also review extended Histogram information).

A bottom bar that runs horizontay through the bottom of the histogram also provides you with a 'Noise Meter' relative to the sensor's light intake. A narrow white colored bar indicates very low noise; as the bar widens, it will start shifting towards red to indicate worsening noise in your image as well!

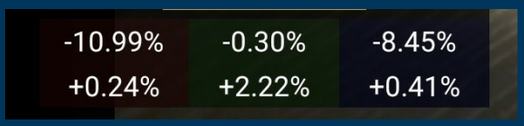


You can tap on the histogram to make it larger (and tap again on it to shrink it).

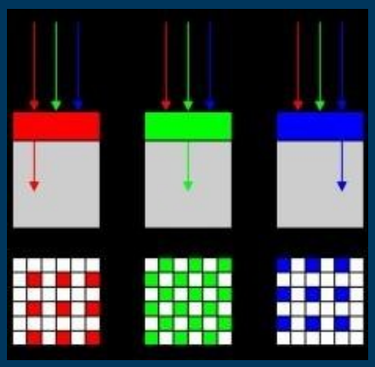
This mode will display the histogram without a see through background to ensure charts are visualized accurately. It will also force on the Extended Histogram Information regardless if the setting is Off (see next page). This override will not carry back once you shrink the histogram back.



[4.30.1] Extended Histogram Information: This toggle enables an additional section of the histogram to display detailed color channel information. To understand your sensor, it's essential to know that it is built on a Bayer pattern where each pixel receives a Red, Green, or Blue filter.



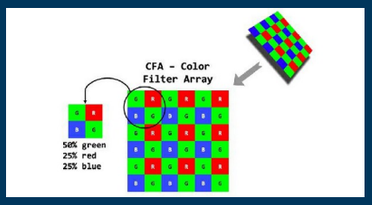
Enabling this option adds a bottom area that shows the individual Red, Green, and Blue channel data, allowing you to view the full range of the sensor's color space.



This feature uses percentages to indicate when separate channels are exceeding their own dynamic range:

Positive values (+) indicate the percentage of the sensor channel that has clipped (highlights that exceed the channel's range and are now pure R/G/B).

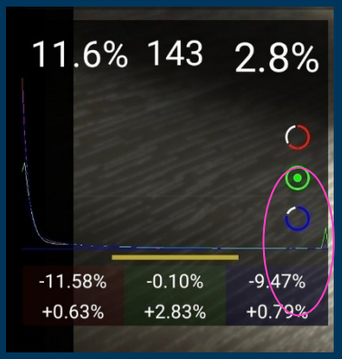
Negative values (-) indicate the percentage of the sensor channel that has crushed (their area is reading pure black and holds no more data).

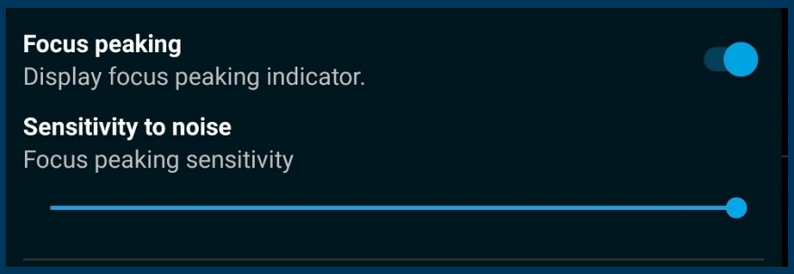


Because different color channels have varying light sensitivities (for instance, green pixels take up 50% of the sensor and often receive more light), they will frequently crush and clip at different thresholds.

This powerful feature allows you to carefully assess the scene with absolute precision and can even be harnessed for highlight recovery/reconstruction by intentionally clipping only individual channels. For a comprehensive explanation on how to best harness this function, see page 81 [4.44.14] for more on highlight recovery).

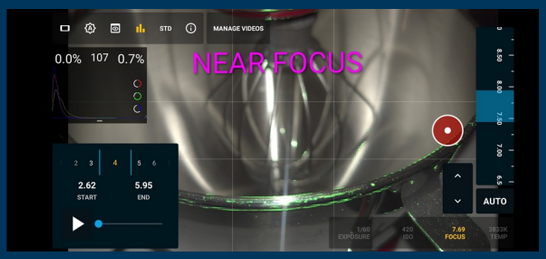
NOTE: The 'traffic lights' that appear to the histogram are a way to quickly review the color channel highlight clipping for an area of up to 1.00% (+0.00% will be light off, +0.01% to +0.99% will begin circling light), anything equal to or beyond +1.00% will be indicated by a fully circled and illuminated light of its respective color channel





[4.31] Focus Peaking: This setting enables a focus peaking overlay that displays green lines around surfaces in the scene that are currently in the sharpest focus.

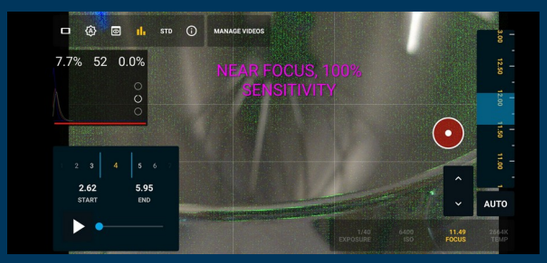
This is a critical tool for fine-tuning focus and can be used with either manual or autofocus modes. The overlay will only appear once you have opened the focus controls by tapping the Focus value on the bottom-right bar and opened its corresponding slider rack.

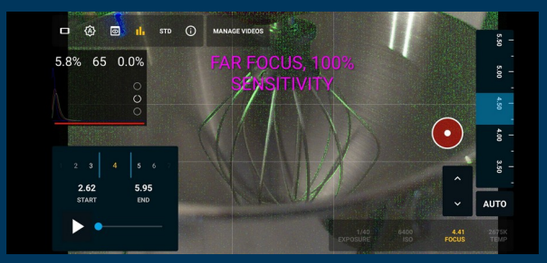


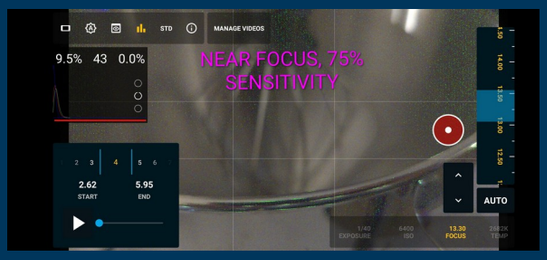
[4.31.1]Sensitivity to Noise: This setting works in conjunction with Focus Peaking to adjust the sensitivity of the green overlay.

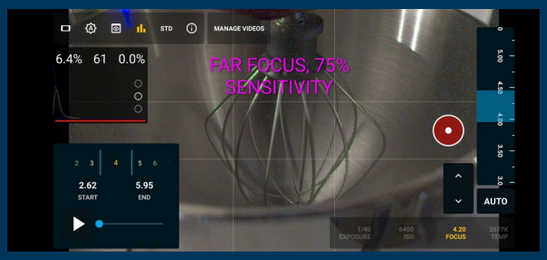
Depending on the Viewfinder Resolution (see page 48 [4.18.4]) and the lighting (especially in low light) sensor noise may induce false positive data by showing peaking patterns on surfaces that aren't actually in sharp focus. You can reduce the sensitivity to mitigate this noise, but at the cost of a less prominent overlay.

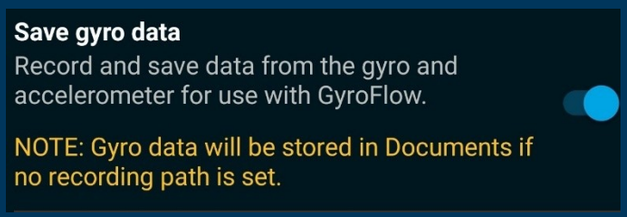
Tweaking is encouraged to find a suitable balance, as too much sensitivity will introduce significant artifacting on the viewfinder, while too little will make the overlay almost non-existent.





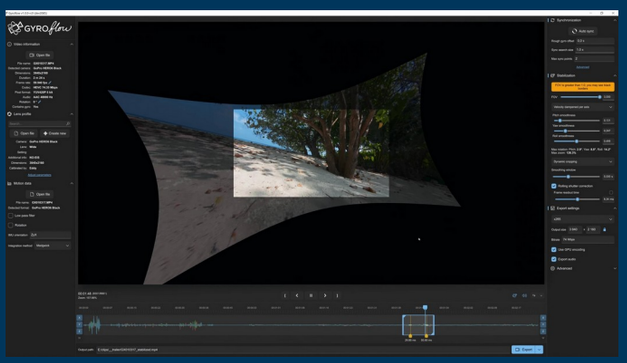






[4.32] Save Gyro Data: Toggle this setting On to have the app record and save gyroscopic and accelerometer data (the movement sensors on your phone's readouts) into a .gcv file required to use Gyroflow stabilization.

NOTE: YOU MUST SHUT OFF OPTICAL IMAGE STABILIZATION IF YOU INTEND TO USE GYROFLOW SINCE THE MOVING LENS ELEMENT IS NOT ACCOUNTED FOR IN THE SENSOR DATA!!!



Gyroflow is a program that can apply superior camera digital/electronic image stabilization than the one otherwise present on phones or full sized cameras.

It works via calibration to your specific camera lens, which allows it to apply things such as distortion correction and de-warping, as well as it can fix things like rolling shutter.

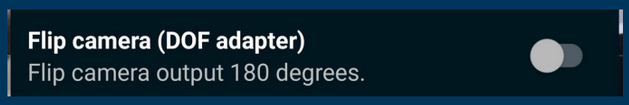
More importantly, this is then combined with its movement based stabilization algorithm that leverages your accelerometer /gyroscopic sensor data to measure the specific scene displacement rather than just visually guessing the movement that occurred.

The resulting shake reduction is nothing short of spectacular, however do note it's generally more effective with faster shutter speeds (180° rule lovers beware) as well as it will work best with wider lenses. As with all forms of digital stabilization, you can expect some cropping too, however this is adjustable.

If you haven't already, I would also encourage to check them out and try it for yourself as its an incredibly effective tool! They also have an amazing community and their developer is super responsive – they're like long lost cousins to the MotionCam community.

https://gyroflow.xyz/

https://discord.gg/6zyUExsXe



[4.33] Flip Camera (DoF Adapter): Use this toggle to flip the viewfinder upside down/by 180 degrees.

This setting is extremely useful for whenever you use an external lens or Depth of Field adapter which will generally provide an upside down image projection; this is also the case for some lenses that have less prism elements and do not correct the light intake. This results in an inverted light input which can make viewing difficult in both cases.

This option does not impact the footage captured and is only applied at the viewfinder level. Also do note that at the moment it does not work with Direct Preview, only Android Preview.

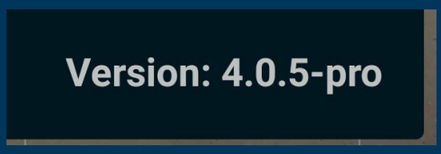
[4.34] Squeeze Viewfinder: This section allows to further adjust the viewfinder squeeze. This is particularly useful for whenever you use external lenses such as anamorphic adapters, which will squeeze the image due to the inherent nature of their distortion. This will naturally warp the preview, use these setting therefore to 'unwarp' the preview.

Any changes done on this area will not impact the capture data and are only meant to assist in the preview. Also, worth mentioning, it will not work with Direct Preview either.

The top slider controls the Vertical Squeeze and the bottom slider controls the horizontal squeeze. Slide left to desqueeze and right to squeeze on both. The lowest/leftmost value is 1.00 representing no squeeze whereas the highest/rightmost value is 2.00, representing a full squeeze by a factor of x2

The bottom numbers provide popular anamorphic lens squeeze ratios such as 1.33, the V represents Vertical and H represents Horizontal to advise you of which aspect is being squeezed by the preset.

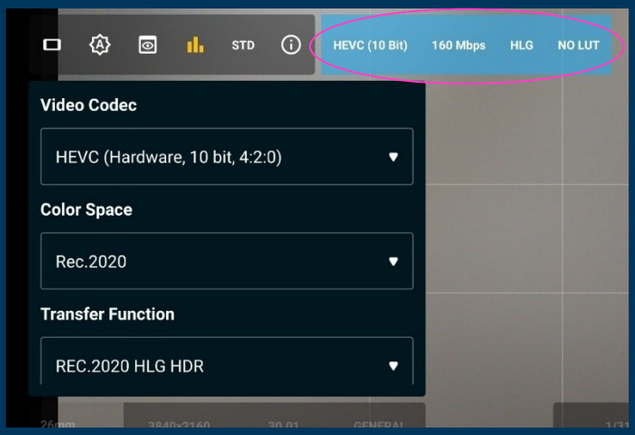
Press on the RESET button to undo any slider or preset changes applied.



[4.35] App Version (Easter Egg - Command Prompt): This area allows you to verify which app version you are on specifically.

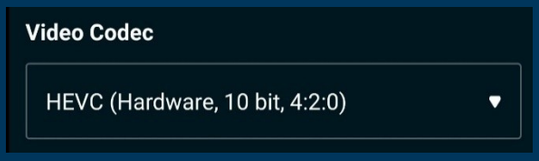
Still reading? Congratulations, you've found a saucy Easter egg! Well then, press and hold the version number to see something cool; the Command Prompt! This feature is still a work in progress. It's hoped one day you'll be able to use it for even more advanced system level app functions! For now, not much to report however… 😁

[4.36] DIRECT LOG VIDEO SETTINGS MENU



Please note that the next following UI Elements are only relevant to the Direct Log Video mode and this settings bar will NOT be applicable nor relevant to the other capture modes – although it does share heavy similarities with the In-App Video Editor (see page 114 [4.73] for more details)!

To open the Direct Log Settings menu, tap on the top bar displaying the encoding settings (any of the 4 categories will open the same menu), the scroll up/down to navigate the available options.



[4.37] Video Codec: This setting displays the current encoder selection, showing its Codec Name, Acceleration Method, Color Depth, and Chroma Subsampling in a clear format: CODEC NAME (ACCELERATION METHOD, COLOR DEPTH, CHROMA SUBSAMPLING). Press the rectangle box to open the full list of options available on your device.

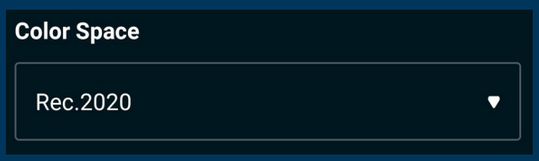
Hardware accelerated codecs (such as APV, AV1, and VP9) will only appear if a specific encoder is made available by your device. However, software accelerated options like ProRes will be available regardless of the device. Be warned that the ability to reliably encode and handle these is entirely dependent on your device's processor and optimizations.

Hardware acceleration implies a specialized physical encoding chip for the codec shown is present on your device which provides the best performance available.

Software acceleration implies that your device doesn't have a specialized chip available for the respective codec shown, and will brute force the encoding via sheer CPU horsepower – vastly more demanding.

NOTE: Even the HEVC encoding used here is not the same as the one used by typical video apps, so your performance mileage will vary significantly. Standard video apps rely on the device's highly efficient Image Signal Processor (ISP) and video stream, which are optimized for speed at the cost of image quality. MotionCam, however, uses its own brute force pipeline to capture RAW images then process them in real-time for maximum fidelity. DO NOT USE *the performance of classic video apps to measure the expected performance in MotionCam, as this quality-first approach is vastly more demanding and completely different.*

Additionally, if you do not see the 10-bit option available for HEVC, it may be due to ROM or OEM limitations imposed on your device. See FAQs for more information on this!



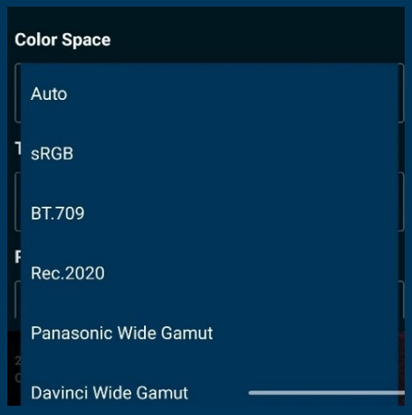
[4.38] Color Space This setting allows you to select the color space to be used in the encoding container.

For those unfamiliar, the color space defines the abstract range of colors that will be captured—not necessarily how the colors are shown. This selection is directly impacted by the color depth in use.

On the accompanying chart, you can observe how the selected color space (represented by the triangles) compares to the full range of human vision (the colored area). For example, a massive space like DaVinci Wide Gamut extends even beyond human perception in some areas.

Do note that no sensor can capture the full vision range of the human eye. In fact, many camera sensors capture colors unperceivable by humans because the wavelength filtering characteristics of their color filter arrays (CFA) can never truly match the human eye's response, not to mention issues like different crosstalk behavior (crosstalk being the spillover of light from one color channel's sensor well into an adjacent color channel). However, these outlier colors are often only achieved with artificial LED colored light.

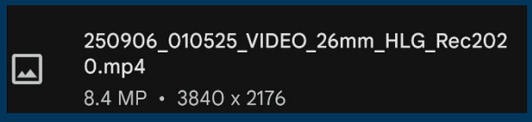
When choosing, you must select an appropriate color space relative to your needs and your Transfer Function (see next section).



A larger color space is not always better, as it requires more data to store and will run into complications if the viewing display cannot properly reproduce it. Therefore, wider gamuts above Rec.2020 primarily benefit log video captures, allowing you to retain more color data in certain intense conditions with super deep colors that may challenge the sensor, or helping you match your log captures' native color space requirements (e.g., Panasonic Wide Gamut + Panasonic VLOG).

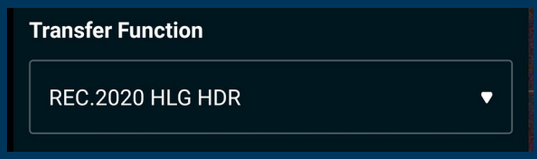
Crucially, to obtain a final delivery-ready video, only BT.709 or Rec.2020 HLG/PQ are recognized options. If you encode using a camera-specific color space like S-Log3's native gamut, it will not be labeled as such in the video metadata, and standard video players will misinterpret the image.

The Auto settings will default to the most appropriate color space (Generally BT.709 or BT.2020). The selected color space will also be stated on the output file's name extension to ensure you do not forget the selection when working with the file..

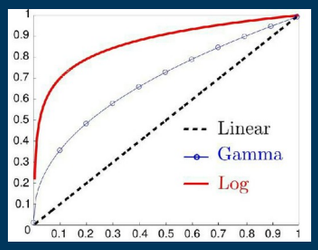


The current list of Color Spaces available in the app is as follows...

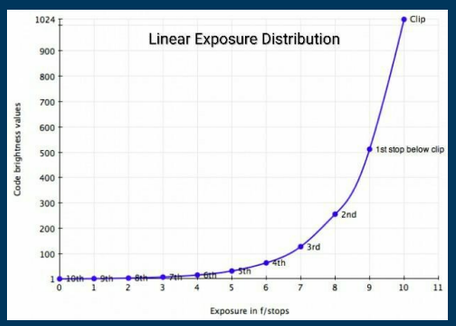
sRGB – BT.709 – BT.2020 – Panasonic Wide Gamut – DaVinci Wide Gamut – ACES AP0 Canon Cinema – Sony S-Gamut3 (Cinema)



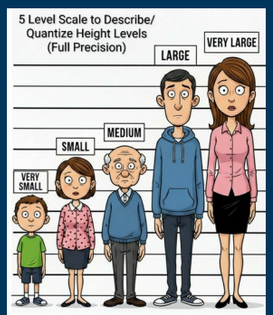
[4.39] Transfer Function (Gamma/Log): This setting controls the Transfer Function applied during encoding, determining how light information is recorded. It's essential for Direct Log as it maps the scene's dynamic range to your final file. If this sounds like gibberish, stick with me—it's key to unlocking this option!



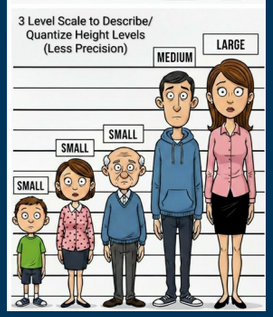
A linear function (the starting point for RAW data) records light precisely. Why is this system terrible for encoding? In 10-bit (1024 total values), roughly 512 values (half of what's available!) are used for just the single brightest stop of light before clipping. The next stop down uses half the remaining precision, and so on.



This drastically under-prioritizes shadows, leaving barely any numbers for dark detail. 'But Ragu! What the hell does this jargon mean!' I hear you asking. Check the top right illustration to see the 'linear' light representation.

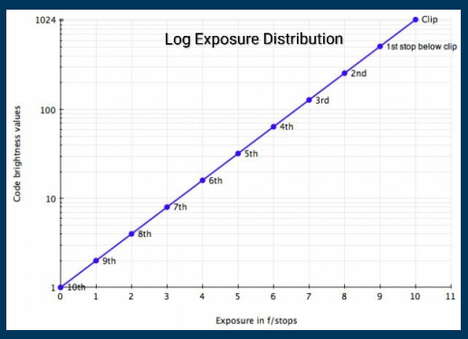


Because of this extreme allocation, Linear data is prone to rounding errors and precision loss in the shadows after lossy encoding, which is why transfer curves are applied: to protect those details from being crushed into pure black during compression.

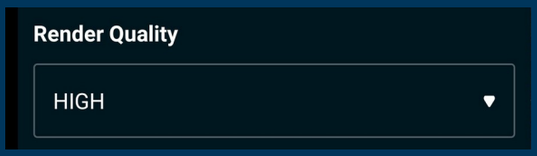


The familiar (Gamma) Rec.709 curve is designed for SDR displays and does not clip highlights by design, that's a myth! The real issue is that Rec.709 is often paired with 8-bit precision. Since 8-bit uses only 256 possible values (vs. 1024 in 10-bit), rounding errors become much more visible. Rec.709 is fine, but its combination with limited 8-bit precision makes it unsuitable for heavy grading, often leading to color banding.

This is where Log functions come in. They are logarithmic—yes, that math! The Log curve mathematically re-distributes the linear values for efficient storage. It intelligently increases precision for shadow information (protecting data) while slightly decreasing it for highlights. Look at the right graph now: Log's dynamic range warping provides a much more equal and discernible representation in the container, giving us better shadow detail (worth the slight highlight compromise!). In other words, the exposure distribution of linear is logarithmically bad, and the exposure distribution of log is linearly good.



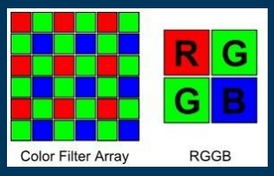
For editing, Log is your best choice for maximum data retention (e.g., S-Log3 + ProRes LT/Standard or 10-bit HEVC + HLG). Remember: if the math is right (big IF) the transfer curve's purpose is only efficient storage. Converting linear data to Rec.709 results in the exact same image as converting linear data to Log, and then to Rec.709. This only concerns transfer functions, though; color space transforms are a whole other complicated topic for later!



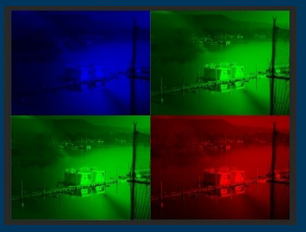
[4.40] Render Quality: This setting controls the quality of the demosaicing algorithm used exclusively in Direct Log Video mode.

You have two settings: Standard, which applies a lower quality algorithm for better performance, and High, which uses a superior algorithm for quality at the cost of processing speed.

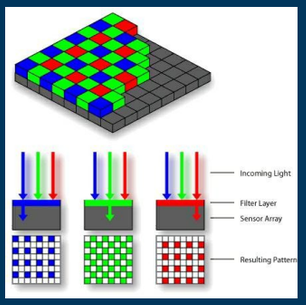
This choice is irreversibly hard-baked into the final encoded output, regardless of the codec or transfer function, and will not affect the quality of RAW or MCRAW files.



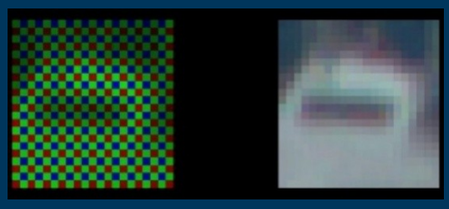
To grasp this, you have to think about your camera's sensor. It uses a Color Filter Array (CFA) in a pattern of Red, Green and Blue within a cluster of 4 pixels (like RGGB, BGGR, etc.) where neighboring pixels never share the same color.



This is a clever trade-off to capture a color image with a single sensor, also giving the Green channel a higher distribution because the human eye is more sensitive to green light. The catch is that every pixel is only capturing one color, creating three incomplete images (see the right illustrations)

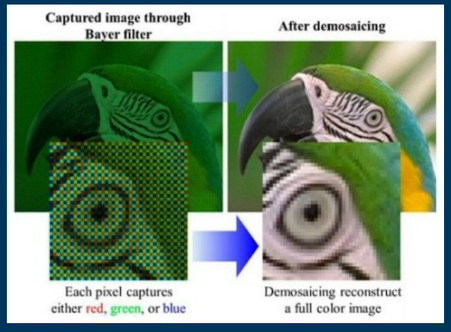


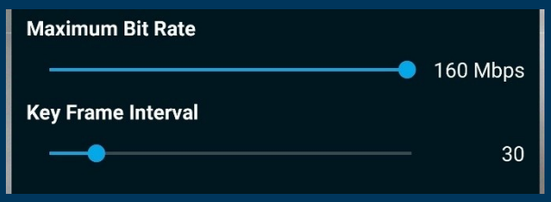
This is basically like when you put on those old 3D glasses! Your amazing brain combines the incomplete red and Blue images from each eye and renders the full color regardless.



Demosaicing is the sensor version of that process. It’s the algorithm that interpolates (or “guesstimates”) the full color of a pixel by using the information from its single-colored neighbors. The better the algorithm you choose, the better the final color reproduction!

All RAW files contain the full Bayer data, which is why editors can use newer, better demosaicing algorithms later. Since you're encoding directly here, your demosaicing choice is critical as it’s irreversible on encoded output files! While this hard baking won’t impact some codecs as badly, it remains a quality factor to consider!





[4.41] Maximum Bitrate: This slider defines the maximum Bitrate allowed during capture for codecs that offer granular selection. Bitrate is the data density per second after compression is performed, measured in Mbps (Megabits per second). Slide right to increase and left to decrease it. The chosen value is shown to the right.

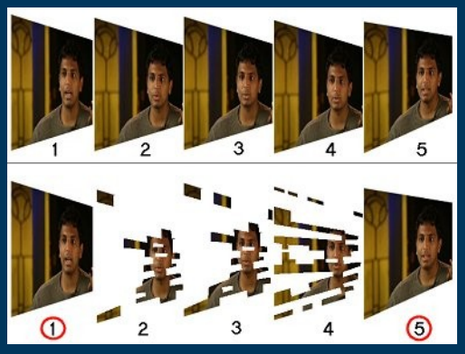
Please note that bitrate allocation is generally not constant, so this option designates the maximum value; the captured output may fluctuate depending on the scene's complexity and the device encoder parameters (firmware level).

A higher bitrate will likely tax performance further but attempts to preserve more data, so you must balance performance against quality. Noisier environments or higher resolutions will inherently require higher bitrates to store the necessary data adequately.

[4.41.1] Key Frame Interval: This setting will only appear alongside codecs providing Bitrate selection and allows you to select the interval between Key frames and the intermediate delta/predictive frames. Slide right to increase and left to decrease it. The chosen value will appear to the right. Although devices won't always respect this setting, it's an important tool in compression.

Key frames (or I-frames) are essentially images in the video sequence under which the full data of the image is encoded. The frames that follow (delta frames) only encode the partial data that may have changed since the last Key frame, recycling the rest of the image. This is a crucial effective method to reduce file size.

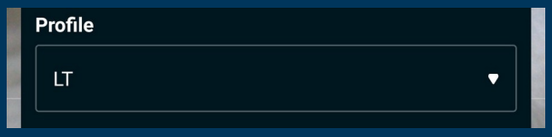
You can observe this efficiency in the image to the right:



The top row uses a Key Frame Interval of 0 (no gaps/intervals). Every image is encoded in full. This is inefficient in scenes with low motion (like a static talking head) and wastes bitrate.

The bottom row has an interval of 3 (3 delta frames between key frames). You can see how only parts of the image that changed between frames are encoded, saving data.

This process significantly saves space, but if the scene is too dynamic (high motion), a high interval can negatively impact quality; there may not be enough data encoded in the delta frames to fully capture all of the new movement. Finding the right balance is key!



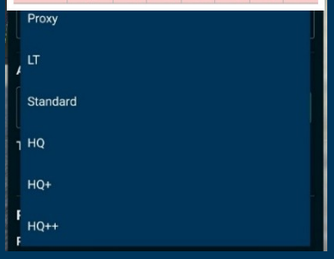
[4.42] Profile (ProRes only): This setting will only appear if you select ProRes under the available codecs. Profile selection replaces both the Bit Rate and Key Frame Interval options that would otherwise appear. Press the box to open the list of available ProRes quality profiles. Key frame selection does not apply in ProRes because it is an intra-frame codec, meaning all frames are encoded in full, independently.

ProRes is offered in multiple profiles with varying levels of data density per frame. This is its key difference: Unlike other codecs which are given a maximum Bit Rate per second and work under that regardless of frame rate or resolution, ProRes profiles designate a specific target Bit Rate that adapts on the amount of frames and resolution received.

NOTE: The actual Bit Rate captured will remain variable, but it will try to work around the guideline target of the profile, this allows the encoding to adapt more predictably based also on the scene. Additionally, light or noise may drastically impact the size given the lack of ISP noise reduction (eg. iPhone ProRes). Remember, Noise = Data!

ProRes target data rates

Dimensions	Frame Rate	ProRes 422 LT		ProRes 422		ProRes 422 HQ	
		Mb/s	GB/hr	Mb/s	GB/hr	Mb/s	GB/hr
UHD 3840 x 2160	24p	328	148	471	212	707	318
	25p	342	154	492	221	737	332
	30p	410	185	589	265	884	398
	50p	684	308	983	442	1475	664
	60p	821	369	1178	530	1768	795



A higher quality profile simply targets a higher bit rate that scales proportionally with your given resolution and frame rate. In other words, increasing resolution and/or frame rate will increase the target bit rate to match the added information being recorded. For example: 4K ProRes 422 will provide a target bit rate of around 589 Mb/s at 30fps, compared to 492Mb/s at 25fps. The target will also increase or decrease proportionally with changes in resolution.

You have multiple ProRes Profiles to choose from…

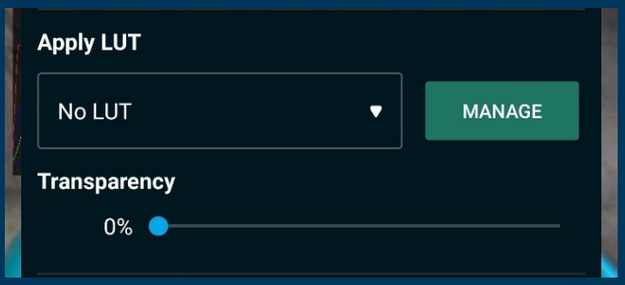
ProRes Proxy: Originally meant for editing proxy purposes. Not recommended due to its very low target bit rate. While it's easier to edit than HEVC because it's compressed far less densely, it provides much less data relatively speaking. Best avoided, if possible.

ProRes LT (Lite): This profile provides a good balance between file size and strong performance. It is a great starting choice and the recommended baseline profile for ProRes usage in order to obtain quality comparable or superior to 10-bit HEVC.

Prores Standard: Although the "Standard" nomenclature simply refers to the fact that this is the normal 'ProRes 422' without compromise, it is the recommended option if you have extra processing power to spare. It provides great data density with less compression than the performance-prioritizing LT profile..

ProRes HQ (HQ+/HQ++): The highest quality of ProRes 422, this version provides the least amount of compression in a visually lossless sense. It is best reserved for the highest visual requirements as it is very resource-intensive. This profile is only recommended if your device has a state-of-the-art, flagship-grade processor and excellent thermal performance. For lower-end devices, pixel binning mode can be a great compromise to allow reliable ProRes HQ captures. The HQ variants (HQ+ and HQ++) simply unleash the target bit rate more and more to increase quality potential.

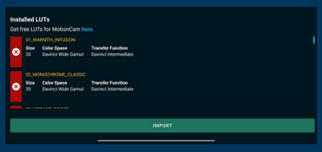
BEWARE: if the target Profile becomes higher than your device can handle (raison d'être of + and ++ increments), this will likely result in audio syncing issues for the capture. Testing beforehand is strongly advised to ensure no capture issues with your specific model.



[4.43] Apply LUT: Use this box to select the LUT (Look Up Table) to be baked into your Direct Log capture footage. Tap on the box to expand the list of LUTs already added.

As of this current release, be mindful that having a LUT in the active selection **hard bakes** the Look into the footage, so disable them before capture if you intend to use this for preview only!!

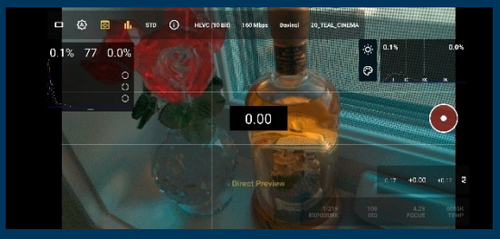
Note: Applying a LUT will immediately override the Transfer Functions and Color Spaces applied before hand and match them to the LUT functions selected upon initial import – you can re-override it again however.

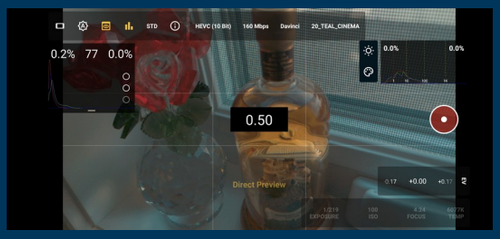


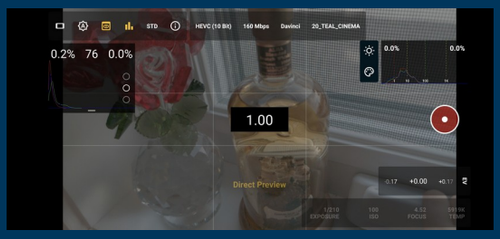
[4.43.1] MANAGE: Press on the green MANAGE button to open the LUT manager and scroll through/view/add/remove LUTs.

[4.43.2] TRANSPARENCY (Blending): Use this slider to adjust the intensity of the LUT overlayed/baked into the image. You can essentially create a hybrid image without completely integrating the LUT into the capture and adjust it to taste. Slider right to increase and left to decrease.

The numerical percentage shown on the left indicates the Transparency amount, with 0% indicating the selected LUT is being applied fully with no transparency, 50% indicating a half-LUT and half-native transfer function image fusion, and 100% showing the LUT is completely transparent and not visible in the image.

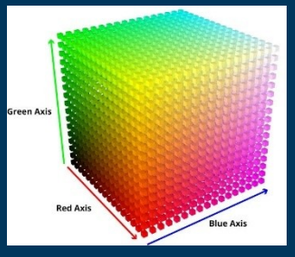






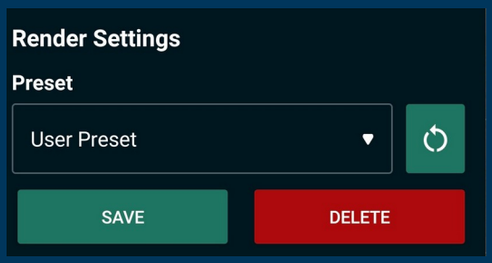
[4.43.3] LUT Summary, for the uninitiated

A Look-Up Table (LUT) is a pre-set conversion guide; a mathematical instruction sheet that transforms every pixel's color, brightness, and contrast to manipulate the output into a desired look.



If you think of grading like adding sauce and seasoning to food, imagine making your own sauce (you mash up a basket of fresh, raw tomatoes, add vinegar, etc). – a LUT is the equivalent of a ready-made bottle of ketchup instead: it instantly and consistently transforms the raw ingredients into a final, flavorful, and palatable product (like vibrant Rec.709 for example).

LUTs are great for monitoring and/or getting an immediate look. That said, the best colorists make their own ketchup from scratch sometimes (custom grading) rather than relying strictly on a 'pre-made bottle' only, as a rigid LUT applies the same transformation regardless of the unique data in your scene. They are excellent tools but should be used wisely.



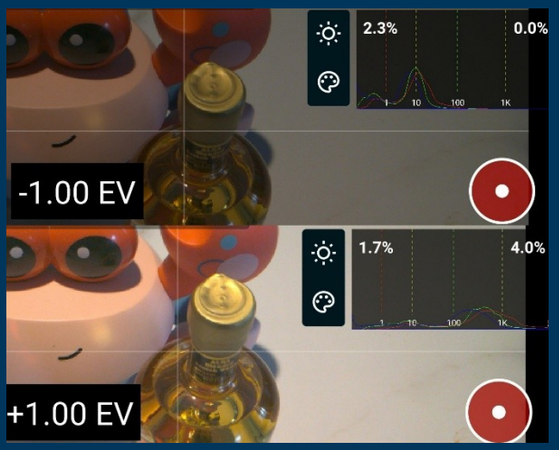
[4.44] Render Settings: This section allows you to alter the encoding pipeline and how the Direct Log outputs are rendered visually. It provides you with a large selection of image adjustments.

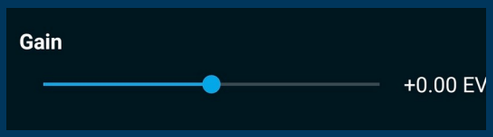
Any changes applied can be undone by pressing the circle arrow. To save a set of changes, press the SAVE button; a prompt will appear allowing you to write the name of your preset. To delete this preset, select it on the drop-down box list and press DELETE.

All changes applied in this list can be viewed and adjusted real-time while using Direct Preview mode (on the respective sliders, hold and slide them to transition into real-time adjustment mode).

Additionally, use the encoder histogram to identify their impact at the data level for maximum precision; you can see an example in the right image comparing +1 EV vs -1 EV (RAW data and capture are identical, however the encoded data is being treated completely different).

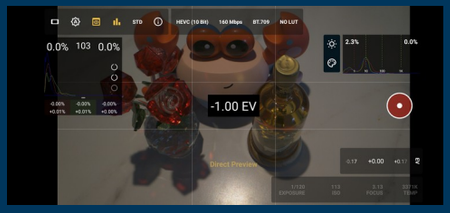
We will touch more on this later as this will be important while capturing using Log transfer functions in order to ensure the full container tonality is being leveraged

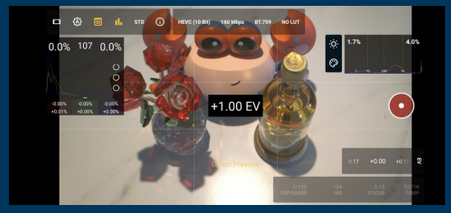


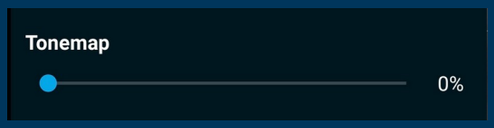


[4.44.1] Gain (Digital EV): This slider provides you with control of the encoding digital exposure compensation. It allows you to boost or reduce the exposure of the raw data in order to better accommodate it into codec containers which may have limitations in their dynamic range for how they store the data (unlike RAW).

Of all the sliders, Gain control is perhaps the most important one as it plays a significant role when capturing in Log (more on this later, jump to page 97 [4.65]). A value of +0.00 EV is the default state and does not alter data. Slide right to increase and left to decrease.







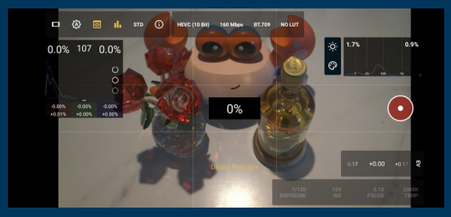
[4.44.2] Tonemap: This slider provides you with the ability to control the tonemapping amount applied during encoding.

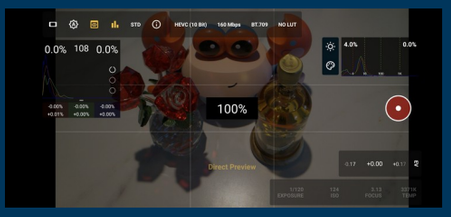
A 0% value is the default state and indicates the setting is not engaged, while a 100% value indicates the maximum tonemapping available that can be applied. Slide left to increase and right to increase.

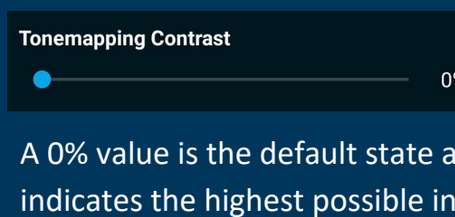
This setting will impact Direct Log performance as it requires more resources to run it.

Tonemapping is essentially the ability to compress a wide dynamic range into a smaller one for better viewing (Eg. 10-bit captured data into an 8-bit SDR), it can particularly useful to properly present a scene that may otherwise look flat if the dynamic range is exceeding the available tonality and can bring back the proper look and feel of a scene

Note: the Tonemap setting must be active (A value of ±1%) for the next two settings (*Tonemapping Contrast/Saturation*) to also work and the intensity applied will directly impact their effect as well.



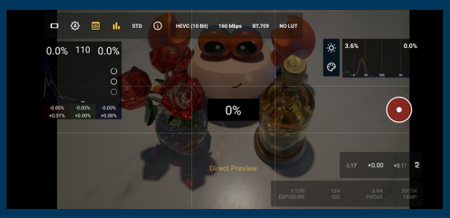


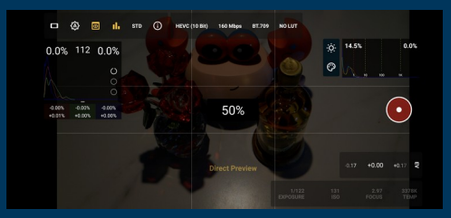


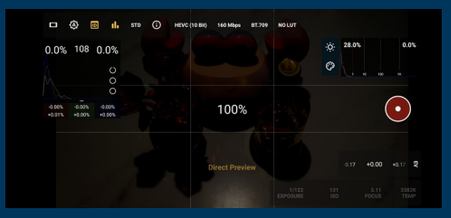
[4.44.3] Tonemapping Contrast: This slider allows you to control the contrast intensity levels applied when Tonemapping is enabled.

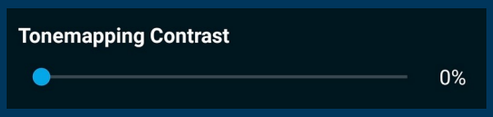
A 0% value is the default state and indicates no changes to the tonemapping contrast levels while 100% indicates the highest possible intensity is applied.

Assuming Tonemapping is On, this option will adjust how the contrast levels are handled on it. The below examples used the Tonemap setting at 100% to show their maximum effect







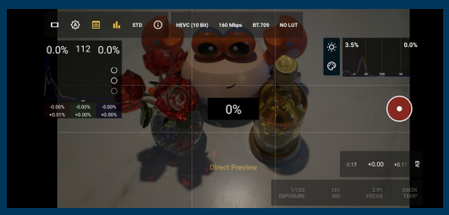


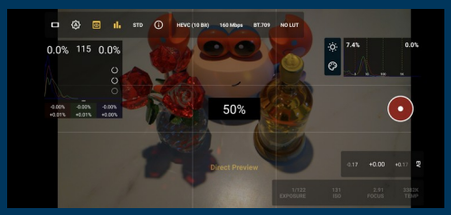
[4.44.4] Tonemapping Saturation: This slider allows you to control the saturation levels applied when Tonemapping is enabled.

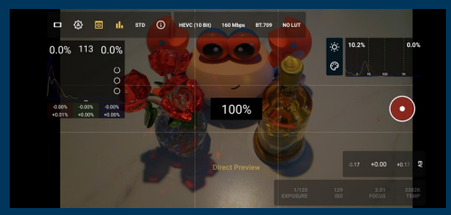
A 0% value is the default state and indicates no changes to the Tonemapping Contrast levels, while 100% indicates the highest possible intensity applied.

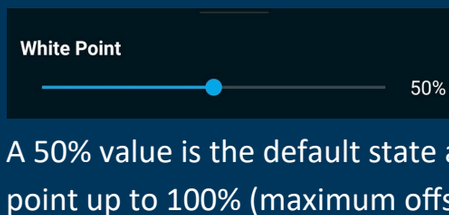
Assuming Tonemapping is On, this option will adjust how the saturation levels are handled on it. The below examples used the Tonemap setting at 100% to show their maximum effect.

NOTE: This Saturation setting is independent of the encoding Saturation itself, and will potentially compound on its effect if both are altered in-tandem. This saturation slider pertains to the tonemapping engine itself.



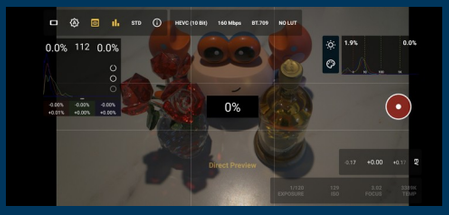


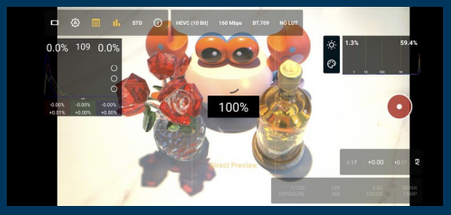


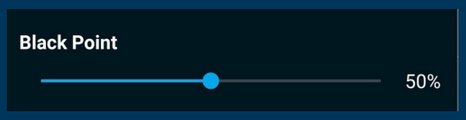


[4.44.5] White Point: This slider let's you control where the 'absolute white' point is and will impact how brighter tones are handled.

A 50% value is the default state and indicates no changes to the White point. Slide right to increase the white point up to 100% (maximum offset) and 0% to decrease it (lowest offset).

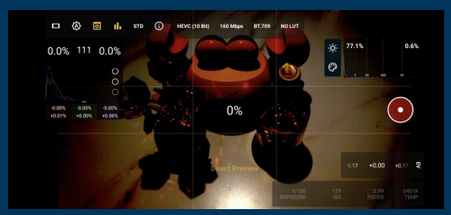


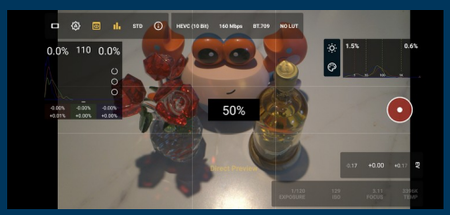


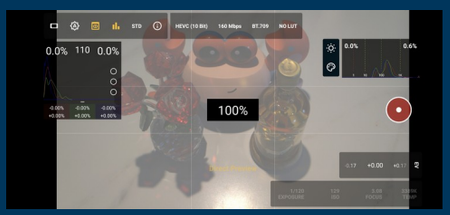


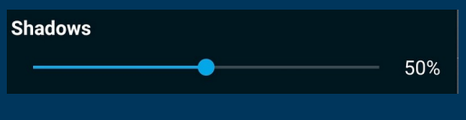
[4.44.6] Black Point: This slider let's you control where the 'absolute black' point is and will impact how darker tones are handled.

A 50% value is the default state and indicates no changes to the Black point. Slide right to increase the black point up to 100% (maximum offset) and 0% to decrease it (lowest offset).



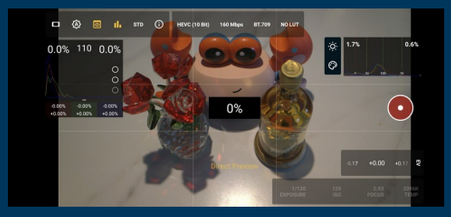


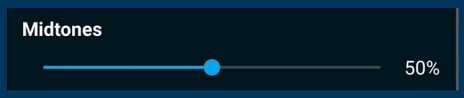




[4.44.7] Shadows: This slider let's you control how shadow details are rendered, allowing you to boost or suppress them accordingly

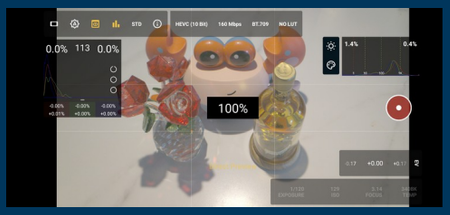
A 50% value is the default state and indicates no changes to the Shadows point. Slide right to boost shadows up to 100% (brightest) and 0% (dimmest) to decrease them.

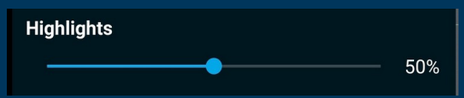




[4.44.8] Midtones: This slider allows you to shift the midtones (middle region of data that's neither dark nor bright where the majority of light data lies) and how they're rendered.

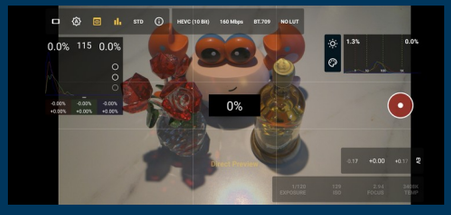
A 50% value is the default state and indicates no changes to Midtones. Slide right to increase the Midtones up to 100% (maximum offset) and 0% to decrease them (lowest offset).

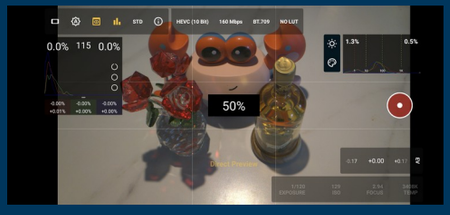


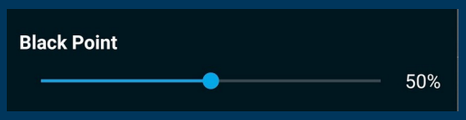


[4.44.9] Highlights: This slider allows you to adjust the intensity of the highlights captured and how they're rendered.

A 50% value is the default state and indicates no changes to the Highlights. Slide right to brighten highlights up to 100% (maximum offset) and 0% to flatten them (lowest offset).

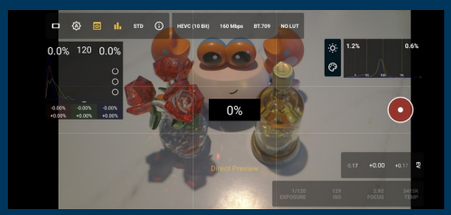


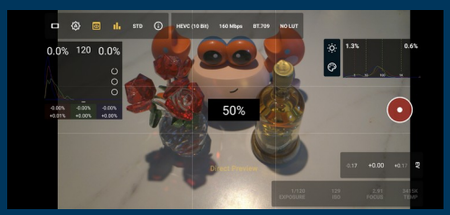


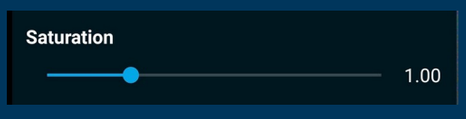


[4.44.10] Contrast: This slider let's you adjust the contrast ratio of the image captured and increase/decrease it.

A 50% value is the default state and indicates no changes to the Contrast. Slide right to increase the Contrast up to 100% (maximum contrast) and 0% to reduce it (lowest contrast).

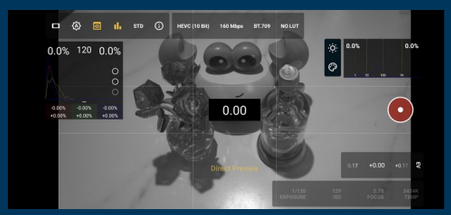


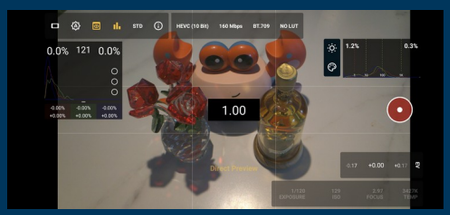


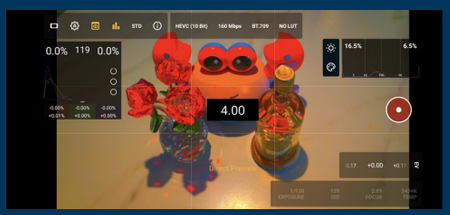


[4.44.11] Saturation: This slider let's you control the image saturation rendered which will impact how intense colors look.

A 1.00 value is the default state and indicates no changes to the saturation as per the image matrix chosen. Slide right to increase the Saturation up to 4.00 (strongest saturation) and slide left to as little as 0.00 to decrease it (0.00 will become gray and retain no colors/becomes gray-scale).





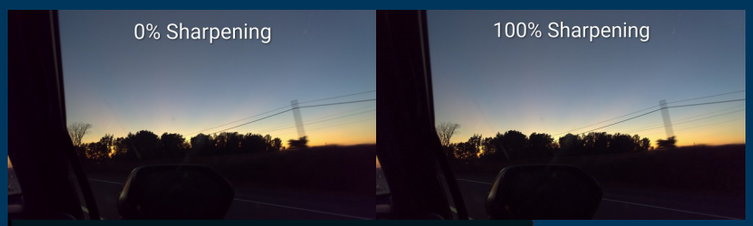


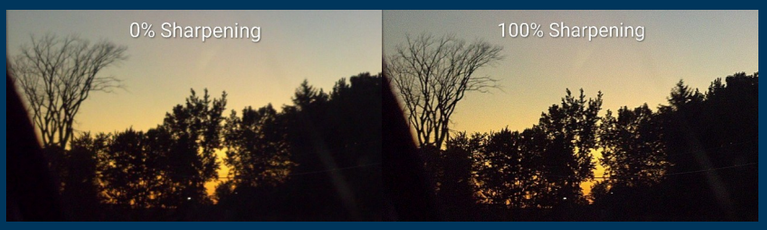
[4.44.12] Sharpness: This slider let's you control the level of image sharpening applied to the rendered output

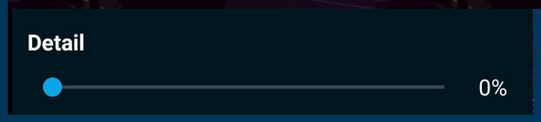
A 0% value is the default state and indicates no sharpening is being applied. Slide right to increase the sharpening level up to 100% (highest sharpening possible) and slide left to decrease it back. This setting will impact performance and requires more resources to apply it when enabled (±1% value).

Sharpening adds contrast around the edges of surfaces and textures in the image, creating the illusion of a sharper, 'more detailed' output.

It looks for variances between pixels or sudden changes (which may indicate a different texture or object) and amplifies the contrast between them. Beware, however, as overdoing it will create an unnatural look with added noise and odd artifacts in edges, to name a few.





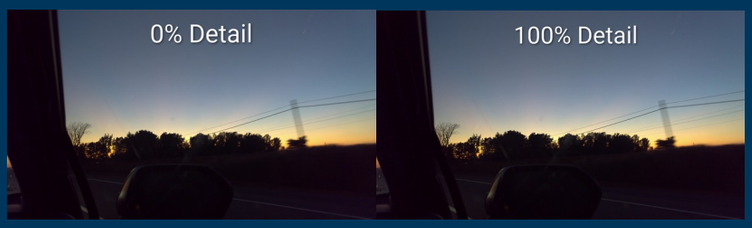


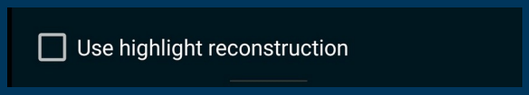
[4.44.13] Detail: This slider let's you control the level of image Detail enhancement rendered which will impact sharp finer details look.

A 0% value is the default state and indicates no Detail enhancements are applied. Slide right to increase the Detail up to 100% (Maximum enhancement) and slide left to reduce it. This setting will impact performance and requires more resources to apply it when enabled (±1% value).

Detail enhancement, similarly to sharpening, attempts to also increase contrast between pixels that show variances, however it more so targets smaller surfaces and textures, providing with a more granular sharpening control that is less aggressive on bigger textures and surfaces but may amplify noise. Best used to enhance finer textures like landscape details and such.

Use with care as too little will result in only larger surfaces being sharper and insufficient micro detail enhancement, too much will result in significantly amplify grain and produce artifacts.





[4.44.14] Highlight Reconstruction: This feature allows you to potentially reconstruct or recover clipped highlights by leveraging data from the color channels that have not yet been fully blown out.

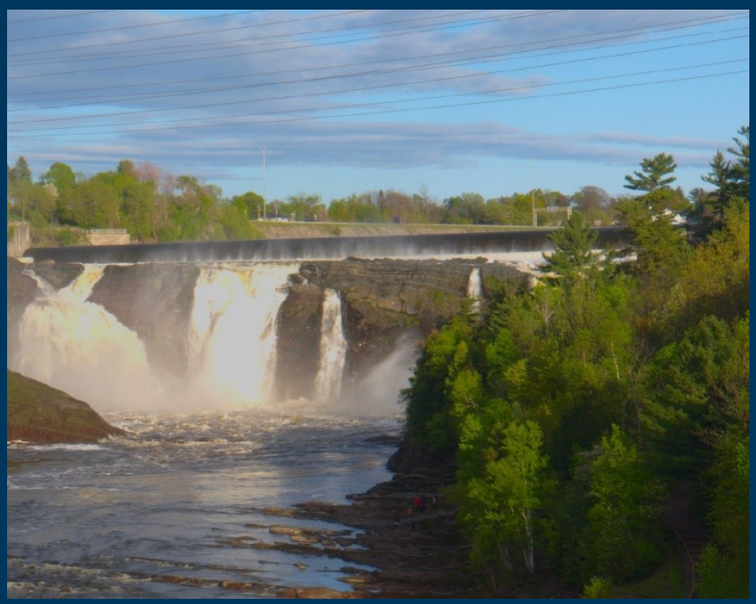
WARNING: This option will heavily impact performance. Check the box to toggle On.

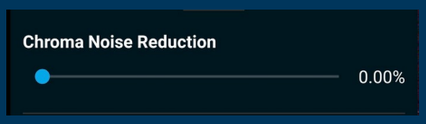
We will cover the in-depth use of this option later for DNGs (see pages 110 & 111), but the principle is simple: you intentionally use the Histogram to controllably clip one or two channels of the sensor's color channels (Red, Green, or Blue). You then use the remaining, non-clipped channels' data to mathematically generate the missing information for the clipped channels, un-hiding the area that would otherwise show pure white.

By leveraging this additional dynamic range, you can push exposure further into otherwise unusable ranges to reduce noise and enhance shadow data.

This technique is best used for specular highlights or surface lights. The reduced color accuracy from using incomplete color channels will impact color fidelity; in other words, do not use this to recover blown highlights on people/skin tones or anything that requires good color reproduction—otherwise, they will possibly look... interesting...

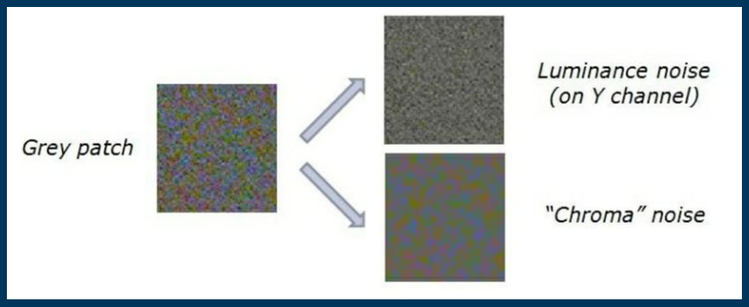
Additionally, although we'll cover them it later as mentioned above, MotionCam offers visual overlays to effectively indicate clipping areas and the amount of channels clipped, outside of the histogram alone. This makes this option quite a powerful tool, particularly for ETTR enthusiasts!





[4.44.15] Chroma Noise Reduction: This slider let's you control the level of Chroma Noise Reduction applied on the rendered video data. This setting will slightly tax performance but can provide basic denoising.

A 0% value is the default state and indicates no Chroma Noise Reduction is active. Slide right to increase in 12.5% increments up to 100% (Maximum out of 9 total steps) and slide left to reduce it. This option will begin to impact performance and demand lightly more resources to apply it when enabled (±1% value).



Sensor image noise can generally be broken down in two categories, the Chromatic noise (aka – Chroma or Color Noise) and Luminance Noise (aka – Luma noise or image grain). This Setting attempts to isolate and target the random Chroma Noise which is the more visually damaging type if left unchecked.

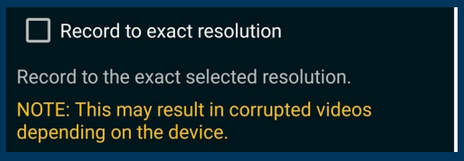
This random noise or color data provides some useful data in its variance, however it can often become overwhelming in noisy environments and can start detracting from the image.

There's multiple noise reduction methods, however this setting's specific one attempts to target the random chroma noise in every single individual frame and tries to mitigate it.

Color denoising comes with a slight risk of suppressing the colors particularly in micro details, however it can also help to produce more usable low-light content.

Although adding more light should always be the first noise reduction method, when that's not available, this makes a great tool without impacting performance drastically.

[4.45] Advanced Settings



[4.45.1] Record to Exact Resolution: Sometimes, your selected resolution might be slightly padded with extra pixels to ensure the container resolution is a multiple of 64. This is not a random effect! As the video is being processed, the device hardware often requires the resolution to be divisible by 64. If it isn't, the encoder can't work correctly, potentially leading to corrupted outputs, artifacts, or video anomalies (and you don’t want that).

When this setting is disabled/unchecked, the state, the app takes care of this by automatically adding a few extra pixels on the border areas as needed, respecting the hardware's rule.

We aren't going to touch on the deep programming logic behind this issue here; just know that the easiest path is to respect the rule and allow the padding.

For example, if you shoot in 3840x2160 (4K UHD), you might expect an output of something like 3840x2176 instead. This will vary on your specific resolution crop.

As for how this extra padding shows up: you can generally observe it right on the edge of your video frame, usually as a very minor mirror effect of one of the border areas. You can simply crop this off in post (or ignore it if it’s minimal).

How noticeable the mirroring looks will depend completely on your resolution and device.

Enable this option to force encoding with the precise resolution crop you chose, skipping the padding. This can be great for workflow since rendering with the pixel padding isn't always necessary and all devices react differently.

That being said, testing beforehand is essential, as forcing an exact resolution can potentially break, corrupt or create anomalies on your output! Feel free to experiment!

[4.45.2] Buffer Size (Direct Log only): This setting allocates the size of the RAM buffer the app will use for capturing Direct Log video. Slide right to increase and left to decrease.

When capturing, the RAW frame data is placed into a memory queue (a waiting zone) before the device's encoder can process it. An encoding deficit occurs when the incoming data overwhelms the device (waiting zone runs out), meaning the queue builds up faster than the encoder can clear it. Once the queue exceeds the allocated buffer size, the data spills over, and the app can no longer hold new frames reliably, resulting in dropped frames.

Increasing this memory buffer provides the app with more room to hold the data, mitigating dropped frames for shorter performance bursts. It also gives you more reaction time when the device overheats and slows down, as you'll observe the memory usage filling up more gradually rather than being instantly overwhelmed.

You should use a higher setting if your device has sufficient RAM, though it is not mandatory. Be aware that if the size is pushed too high and the device has insufficient total memory, it may lead to app instabilities.

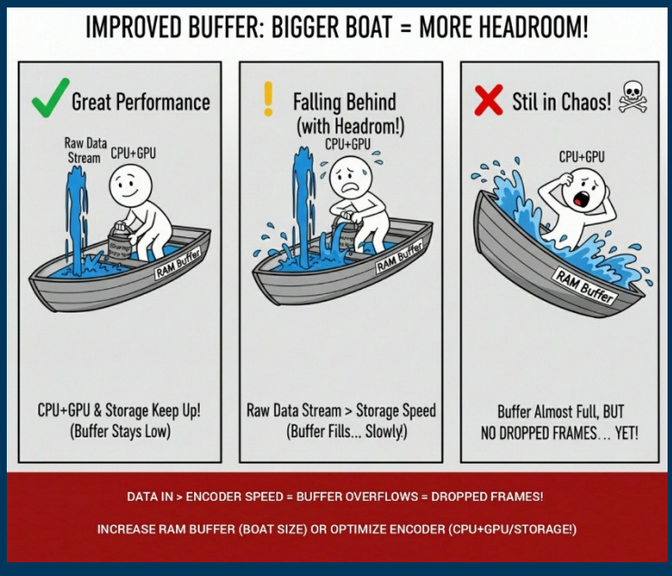
💾 Buffer Size Explained Simply

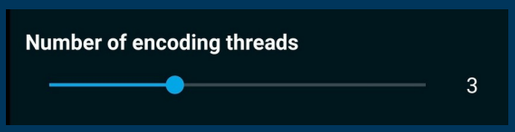
The RAM Buffer (The Boat) is your holding area for raw video data waiting to be processed.

If the Raw Data Stream (Water Inflow) is faster than your Encoder Speed (CPU+GPU Bailing & Storage Writing), the boat fills up faster than the encoder can ‘bucket’.

Bigger Boat = More Headroom, Increasing the buffer delays the moment your RAM runs out, effectively preventing dropped frames for longer.

The buffer is especially useful when your encoder slows down due to heat (Thermal Throttling) as it buys the system critical time to keep pushing data.





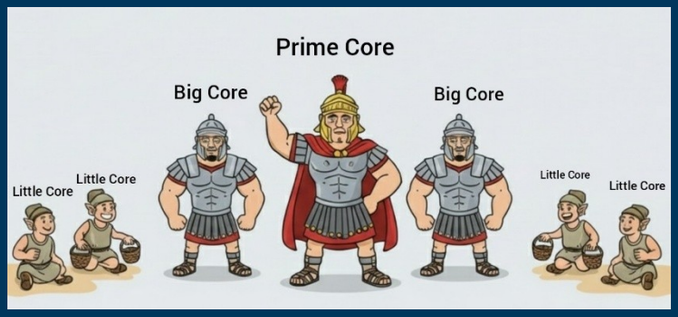
[4.45.3] Number of Encoding Threads (Software Encoding): This setting lets you control the number of CPU threads used when encoding a codec that relies on the CPU for software acceleration (e.g., ProRes). This helps optimize the data load on the processing side. Slide right to increase and slide left to decrease the count.

Note: More isn't always better! Modern CPUs often use a mix of powerful Performance (Big) cores and less powerful Efficiency (Little) cores. Depending on your device's CPU architecture and how it groups these cores, engaging too many threads might accidentally slow you down by involving the slower Little cores meant for power efficiency in the intense encoding task. Your mileage may vary significantly therefore!

We strongly recommend experimentation. You might find better performance by only engaging a lower number of threads (to focus the work on the high-speed Big cores) rather than maxing out the setting and engaging every core, Big and Little, regardless of suitability for the task.

⚔️ Encoding Threads Explained Simply

The Encoding Threads setting determines which CPU cores (Roman figures) are assigned to the encoding task (bailing water, yes we're going back to that).



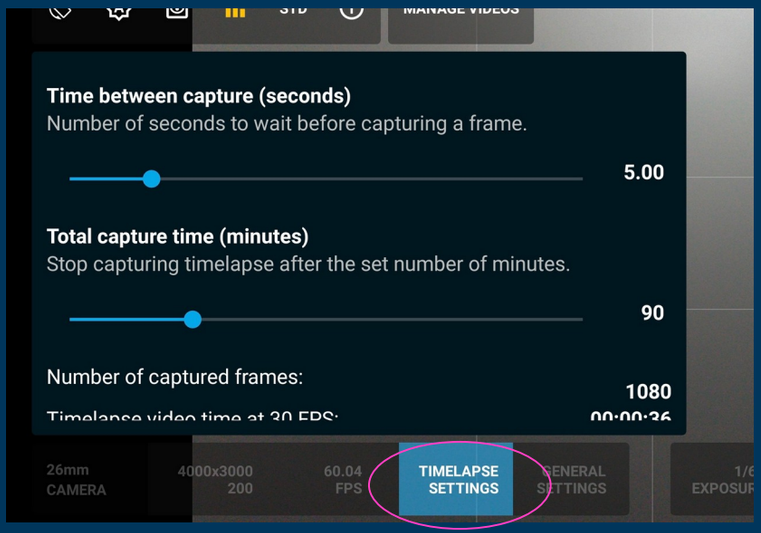
Modern mobile CPUs have Prime/Big Cores (Soldiers) which are powerful but consume a lot of power, and Little Cores (Villagers) which are slow but efficient.

Optimal Threads: Assign the task to only the Prime and Big Cores. They are fast and powerful, best suited to keeping the boat afloat (maximizing encoding speed). We do not care for efficiency in this case as we want max performance.

Too Many Threads: Assigning too many threads often forces the use of Little Cores. They are too slow for intense video encoding and introduce overhead, causing the boat to fill faster than the soldiers can bail (slowing down your performance).

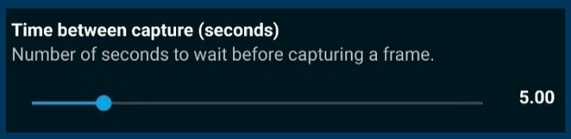
The Goal: Find the right thread count to use only your fast cores, avoiding the inefficient little cores entirely to achieve the maximum, sustained encoding rate.

[4.46] TIMELAPSE SETTINGS MENU



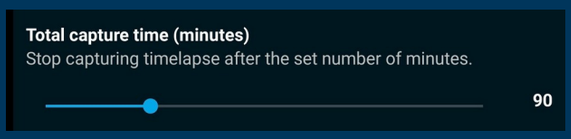
Please note that the next following UI Elements are only related to the Timelapse Mode

To access them, select the TIMELAPSE recording mode and select on the TIMELAPSE SETTINGS button on the bottom left, this will open the capture parameters menu for your timelapse session!



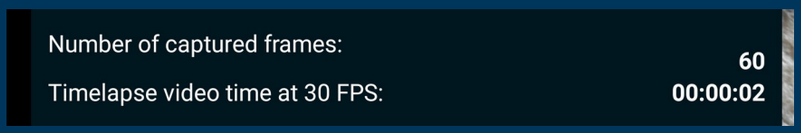
[4.47] Time Between Capture (Seconds): This slider allows you to define the number of seconds/intervals that the app will wait for to space out the saved frames in the session. Any frames captured in-between the gap selected will be discarded to save space.

This value will provide for selections between 0.25 seconds to 30 seconds. Slide left to decrease and slide right to increase. The selected interval will appear to the right with quarter second precision.



[4.48] Total Capture Time (Minutes): This slider allows you to define the duration of the timelapse session in minutes. After the selected time lapses, the app will stop Recording.

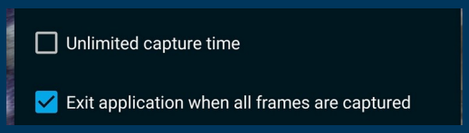
The selection can be adjusted in 5 minute intervals, with the shortest capture time being 5 mintues and the longest being 360 minutes (or 6 hours). If you require longer times, it is possible to override it with one of the following settings below, however (Unlimited Capture Time checkbox)



[4.49] Number of Captured Frames: The value to the right can be used to indicate the amount of frames the app expects to capture assuming appropriate shutter speeds and framerates are selected (that will respect the exact timelapse settings of choice). The following calculation is used…

(60 Seconds ÷ Time Between Captures in seconds) * Total Capture Time in Minutes = Number of Captured Frames

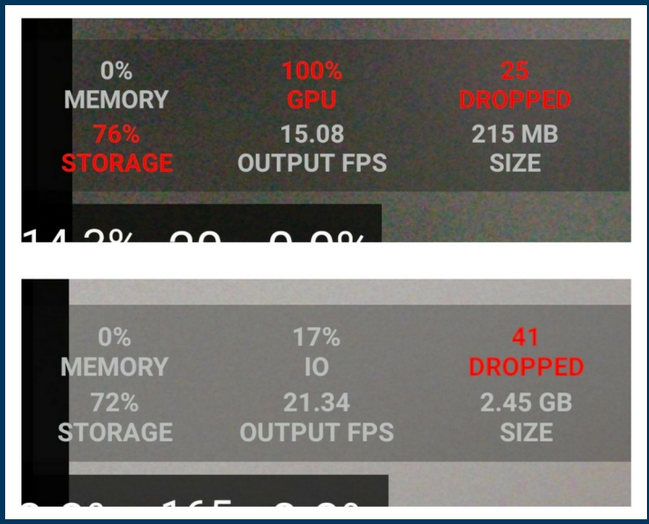
[4.50] Timelapse Video Time at 30 FPS: This timer will indicate how long a video playback would last for the the above timelapse settings chosen, assuming the sequence was viewed at 30 FPS. Use it to determine if the session is sufficient or too short, although it's only a guideline.



[4.51] Unlimited Capture Time: Check this box in order to override the above Total Capture Time selection and force the app to capture indefinitely and override all related sessions duration settings. Using this setting will make the app no longer provide accurate estimates for the timelapse however.

[4.52] Exit Application When All Frames Captured: This setting will make the app exit upon recording all the frames as per the above Number of Captured Frames calculation estimate. This is a great way to automate the session and allow it to end autonomously, however it may also be overridden with the Unlimited Capture Time option.

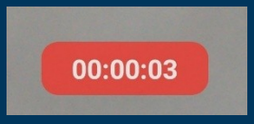
[4.53] Recording Session Performance Interface



This following section will dive into the Recording Session Performance Interface, or otherwise, the section that appears when recording begins and indicates details including the storage used, frame drops, processing bandwidth available, and more.

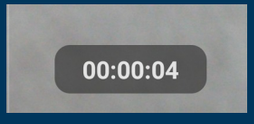
The following elements will only appear once a capture/recording session has begun and will not appear otherwise. Use them to evaluate how your device is handling the requested capture parameters.

[4.54] Recording Duration Timer: When you begin recording, a session time counter will appear in the center upper region of the display. It will change depending on the situation to inform you of the current state and length of the session. Will appear in HH:MM:SS (Hours:Minutes:Seconds) format.

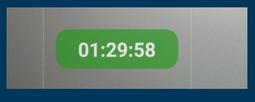


[4.54.1] Normal Mode Timer: Shows how long you've recorded in the current session.

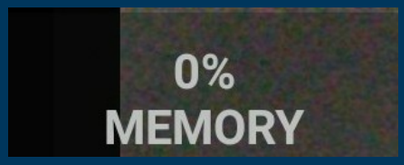
[4.54.2] Buffer Mode Timer: Shows how long you've recorded in the current session and indicates Buffer Mode is in use.



[4.54.3] Paused Timer: Will show the timer in a grayed out status whenever you use the pause button to indicate the session is paused. Will remain static until you press resume and it will then return back into the Normal Mode.



[4.54.4] Timelapse Countdown: The normal timer will switch to green and into a countdown mode instead to indicate you are now shooting in timelapse mode and the time left until the session ends. It may behave like the Normal Mode timer however, if you selected the Unlimited Capture Time setting.



[4.55] Memory Usage Percentage: This value displays the live usage of the RAM buffer allocated to the app. During RAW video or Direct Log recording, the incoming RAW sensor data is first placed on this memory buffer before the CPU or GPU can process it into the final format (like MCRAW or a Direct Log codec).

This concept was explained in the Direct Log Buffer Size section as bailing water off a boat with a leak.

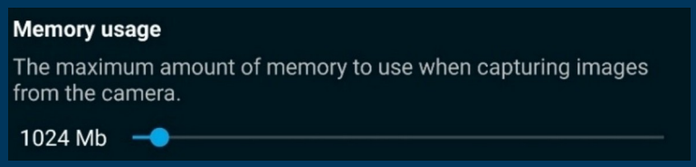
You can use this value to instantly assess your device's encoding efficiency. If the value sits near 0%, your device is successfully processing the RAW data faster than it comes in—you're stable!

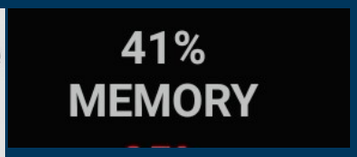
If the value starts climbing, however, it signals that the performance required exceeds what your device can output; the buffer is building up, and the device will eventually become overwhelmed.

When this value hits ±50%, it will turn red - this is your critical alert for an imminent Memory Overrun. This is your immediate signal to stop recording before your session is suddenly bombarded by frame drops.

While performance surges and drops are normal, this buffer provides crucial breathing room to mitigate drops in such event. If the cause for the performance bottleneck isn't controlled, the Memory will eventually reach 100%, resulting in frame drops.

Use the Memory slides (pages 57 [4.27.1] & 84 [4.45.2]) and manage this for better stability or performance.



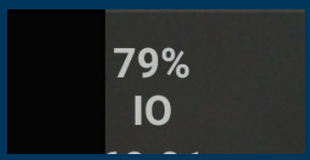


[4.56] Storage Amount Used: This value displays the percentage of storage currently used in your device's selected recording location (Internal Storage or an External SSD). A 0% value means the location is empty, while 100% means you have no room left to record.

The value will turn red at ±76% to alert you that storage usage is reaching high levels.

It is strongly discouraged to fill up any device above 85%. Doing so often results in a performance impact and can cause storage writing speeds/bottlenecks as the system begins throttling down the write speed.

This risk varies by capacity; for example, 20% remaining on a 1TB device is far less critical than 20% left on a 128GB model. Never fill your internal storage to 100%, as this can create file management anomalies and severely slow down your device or even crash the Android system OS.

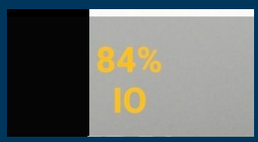


[4.57] Input/Output Capacity (RAW Video Mode Only) : This value displays the percentage of available IO (Input/Output) currently used by the recording session. IO is your storage's ability to manage the flow of incoming RAW data against its ability to write that data out (as MCRAW).

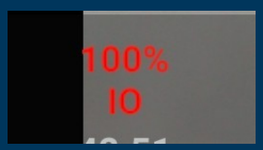
A 0% value indicates no IO is in use, while 100% means you've maxed out your storage's writing speed, whether due to slow speed, throttling, or other bottlenecks. This percentage is a real-time gauge of your device's storage performance and session health.

Lower IO values are always better, but as long as you can hold a stable value below 100%, performance is considered sufficient.

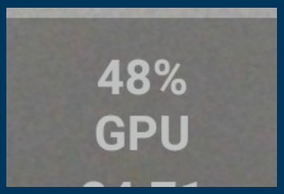
The value provides critical warnings: it will turn yellow for moderately high usage (80% to 89%) and turn red for critical IO levels (90% to 100%).



Reaching 100% will begin overflowing data into the backup Memory Buffer and result in massive frame drops once that gets overwhelmed too.

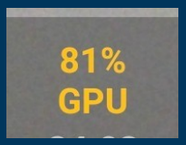


Note: This setting will only appear during RAW video capture. Direct Log Video will show GPU percentage performance instead.



[4.58] GPU Percentage Usage (Direct Log Video Only): This value displays the percentage of available GPU power being consumed during Direct Log Video captures, allowing you to gauge the performance demand on your specific device.

You ideally want to keep this percentage as low as possible for best stability, but it will naturally fluctuate based on the scene, lighting, codec, container, and selected rendering settings (which have a big impact).



A higher GPU usage not only lowers potential stability but also generates significantly more heat. The percentage shown will adapt to reflect the system resources available, as thermal throttling itself reduces the GPU's available power.

A 0% value indicates no GPU usage or that the device isn't rendering. A 100\% value indicates you've exceeded the available processing power, forcing the device to immediately fall back on the recording buffer to absorb the shortfall. This will result in immediate frame drops once you exceed the buffer allocation.

The percentage turns yellow from 81-90% to indicate higher than desirable consumption, and red from 91-100% to signal that you are about to, or have already, maxed out your processing power.

To summarize, lower is generally better, but this will be highly dependent on your device's power, actual performance and stability. You should also balance the usage against additional rendering capabilities that may be desirable to increase overall quality.

[4.59] Output FPS/Frames Per Second: This indicator allows you to see how many image frames the app is able to capture and process into a container based on your chosen settings.

Ideally, you want this number to be as close as possible to the framerate you've chosen for capturing. Use this value to assess the actual rendering performance the device is providing.

The actual output fps is generally going to be slightly lower than the framerate which is normal, however it can also drop way below depending on actual shutter speed in use, as well as based on the rendering intensity.

Very demanding settings that push the device beyond its limits will result in a drastically lower output, which will result in frame drops

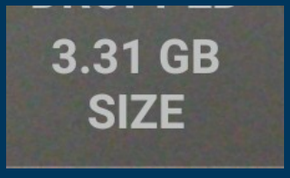
[4.60] Dropped Frames Counter: This area displays the total number of dropped frames within the current capture session.

Dropped frames occur when the frame or image data given by the device cannot be successfully captured, resulting in a "gap" within the frame sequence. This failure manifests as a stutter or jaggedness in your final video.

A low amount of drops can be tolerable, but a high amount will severely impact viewing quality and discard large amounts of incoming image data, making it highly undesirable.

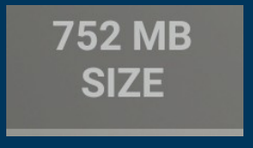
This indicator will show "-" to indicate no drops have been detected. Any value from 1-10 will show in white to indicate nominal performance, or otherwise an acceptable level of frame drops within nominal variance. Any values from 11-20 will show in yellow to alert you about a rising number of drops that may begin to become noticeable. Finally, any number over 21 will be shown in red to indicate a possibly critical level of frame drops.

Ultimately, you will be the final judge of what is acceptable or not. For instance, a very long recording with widely spread-out drops will be far less noticeable than one which is shorter and has a large cluster of dropped events. This indicator is only a simple, quantitative way to measure them.

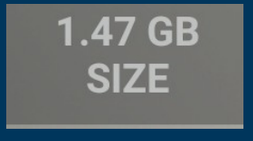


[4.61] Real-time File Size : This indicator provides you with the real-time size of the file you've accrued during the current capture session.

The size indicator measures file size in Megabytes (MB), typically from 0 up to 1023MB. Once the file size exceeds 1023MB, the measurement automatically switches to Gigabytes (GB), rounded to the nearest hundredth decimal place (e.g., 1.02GB).



This value will generally keep climbing as you record. If the size indicator remains static while you are actively capturing, it signals a problem preventing the data from being written (a rare event).



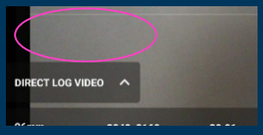
[4.62] CPU Throttling Indicator: This thermal status indicator, located at the bottom left, provides a real-time assessment of your device's thermal/heating state and will appear regardless of whether you are actively recording.

The app receives alerts directly from the Android system regarding the thermal state, which is handled according to your Device Manufacturer's defined specifications on your specific device.

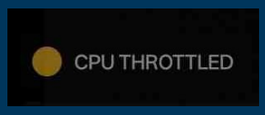
While some devices throttle more aggressively than others, this indicator alerts you to specific temperature thresholds and the corresponding power reduction to expect. In optimal conditions, the app is allowed to utilize the full peak power of the CPU. However, as the device heats up, the Android system begins to moderate this available power at specific, undescribed thresholds defined internally as THERMAL_STATUS_######.

Just because you see this alert pop up doesn't mean you must stop recording; it simply tells you that your available peak performance power is being reduced. If your current capture settings require performance within the now unavailable power envelope, your session may no longer be sustainable. If your session doesn't require the full capacity, this is merely a heads-up that your headroom is shrinking.

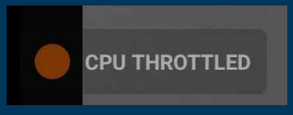
The app will warn you accordingly of the current thermal state with four distinct thermal states:



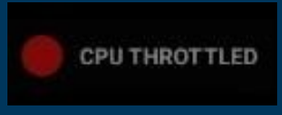
No Alert: Defined by the Android status THERMAL _STATUS_NONE, this shows the System is not under throttling and full power/performance is available for capturing.



Yellow Throttling Alert: Light throttling where Android UX is not impacted, however peak capabilities are mildly reduced. Mostly harmless however indicates heat buildup is beginning to increase. THERMAL_STATUS_LIGHT

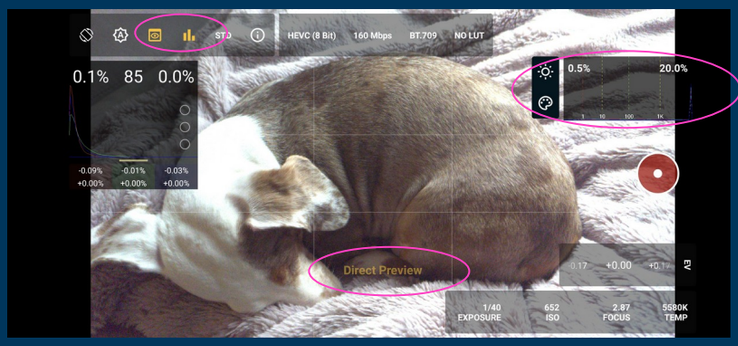


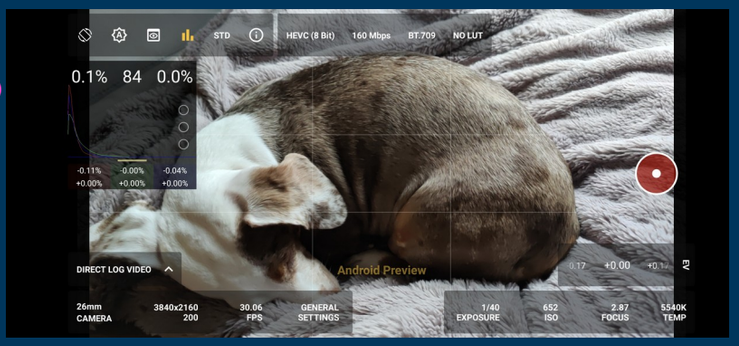
Orange Throttling Alert: Moderate throttling where Android UX is not largely impacted but apps like MotionCam in this case will begin suffering due to heavier processing restrictions. Your device will be noticeably warm and you'll begin to notice potentially less stability and more frequent frame drops. THERMAL_STATUS_MODERATE



Red Throttling Alert: Congratulation! You dun' did it now – you're cooking! This is the last indicator showing severe throttling where a massive performance reduction will be seen across the board. Expect your capturing capabilities to go down the toilet at this point. You should attempt to reduce strain or cool down the device if possible since otherwise, if you keep pushing, the Android system itself may begin to intervene further (may put the device on limp mode, shut down the device or begin throwing OS warnings). THERMAL_STATUS_SEVERE.

[4.63] Direct Preview (Usage and Interface)





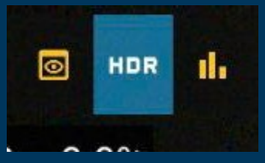
Upon pressing and activating the Direct Preview Mode, the bottom left bar will hide and a secondary dedicated Encoder Histogram to the top right will appear. Two buttons beside it to toggle the alternative Direct Preview Instrument Modes will also present themselves (A Sun and Paintbrush)

A prompt will also appear to indicate the Direct Preview Mode.

To summarize it again, Direct Preview is the app's unique mode allowing you to see the actual image content you are about to capture without any of your device's Image Signal Processing Features interfering or giving you an inaccurate, baked image that the app intentionally avoids. Without direct preview, what you are seeing is closer to what the stock app or standard video settings of your device would shoot.

When you turn on Direct Preview however, the app proceeds to process and show you the settings you've chosen, exactly as they would be as per the RAW stream source alongside any other settings for rendering you've altered. This is one of the app's most powerful and unique abilities, allowing for unprecedented precision and interpretation of the capture data well before you even need to shoot.

NOTE: You can also get to see your content using the HDR mode (if your device supports it and runs Android 14 or above), further elevating the capture previewing experience by taking advantage of your display's HDR capabilities



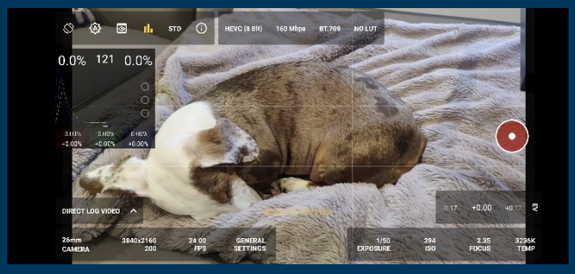
To compliment these capabilities, additional tools and modes leveraging the same control and purity of the source output are given to the user: a dedicated *Encoder Histogram* as well as *False Color* and *Sensor Clipping* overlays!

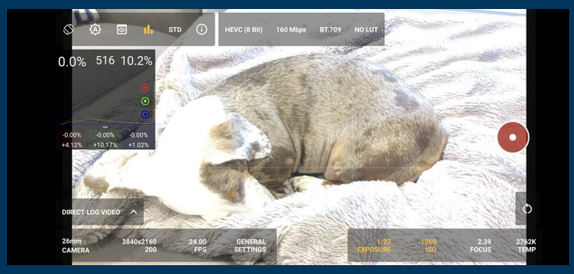
[4.64] Direct Preview Modes: When you start the app or open a mode, a pop-up toast will appear to indicate what viewfinder mode is currently shown on the preview image.

As of this release, you will have 5 modes. The below image columns will all be the same, with the left samples as normal exposure scenes and the right ones as overexposed samples for immediate comparison of the modes under different exposures.

[4.64.1] Android Preview: The standard viewfinder given by the Android system as per the Camera2API. The least resource intensive viewfinder mode, however will generally have processing applied to it.

Although it's possible to diminish the processing applied towards the Android Preview (see page ##) it will still not generally provide you with a high fidelity/accurate preview and doing so may also show an image that's way darker or brighter than the actual capture – in some cases it may even crash the app due to system restrictions.

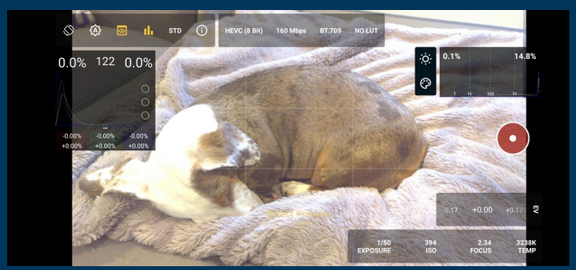


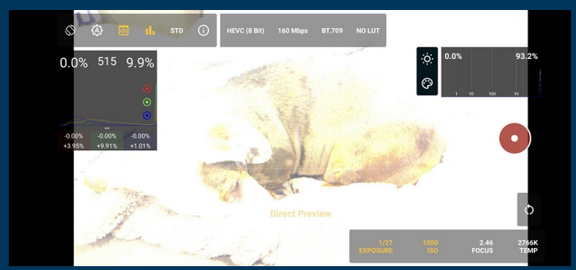


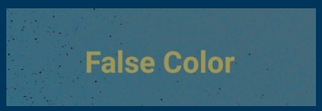
[4.64.2] Direct Preview (Direct Log Mode): Defines the Viewfinder Direct Preview Mode is active with no additional overlays and will show an accurate Preview of your current Transfer Function (gamma/log) or LUT.

[4.64.3] Direct Preview (sRGB/PQ) – Photo/RAW/Timelapse Video Modes: Will appear when capturing outside of Direct Log Mode.

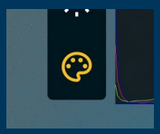
As the RAW data is technically linear, depending on whether you selected the preview to be in HDR or not, sRGB/PQ gammas will be applied respectively to the image to provide normal interpretation of the data. This gives a better idea of how it's being handled at the RAW level.







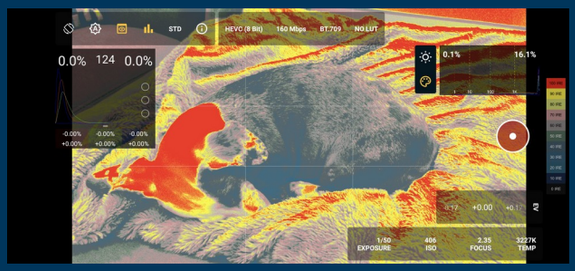
[4.64.4] False Color: This pop up will appear when you press on the painting palette icon and indicates the False Color mode is engaged.

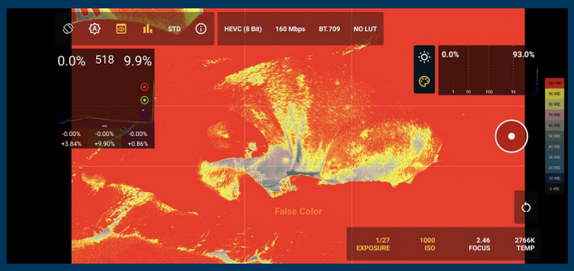


False color is an advanced exposure monitoring tool that emphasizes a scene's brightness levels by mapping them to specific, assigned colors.

This visual system helps you quickly identify contrast ratios and analyze underexposed, properly exposed and overexposed areas, in a more pronounced manner, effectively ensuring correct exposure for recording is achievable regardless of the scene.

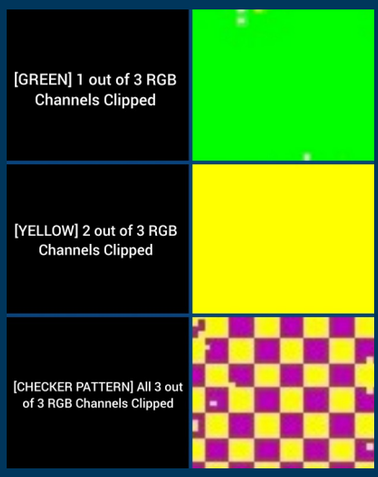
An additional IRE (Institute of Radio Engineers) scale will appear at the right hand side of the display to allow you to gauge the lighting and contrast ratios in a quantifiable and measurable way. A higher IRE value indicates a much brighter area and a lower one will represent deeper shadows until clipping eventually occurs.





[4.64.5] Sensor Clipping: This pop-up appears when you tap the shiny sun icon. This mode is unique to MotionCam. It leverages the app's RAW data handling to give you a visualization of exactly where and when the individual separate RGB channels are clipping in the image.

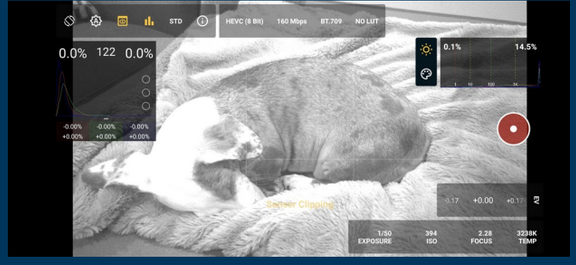
When active, the image appears in grayscale (black and white) to clearly accentuate light intensity, and when clipping begins, colored textures appear on the image.

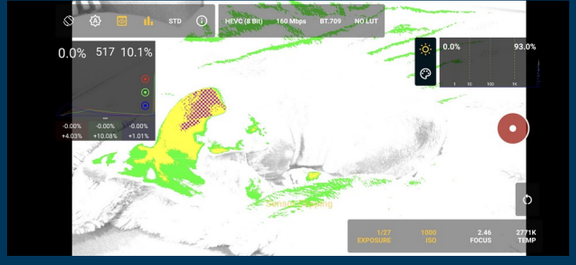


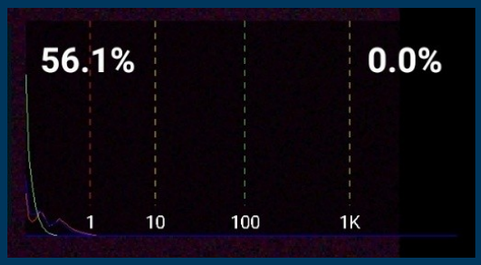
Green areas indicate where one channel has clipped, yellow areas show two channels have clipped, and a checker pattern tells you that complete three-channel clipping has occurred (undesirable, unless that area is not important to your scene).

This mode is designed ONLY to evaluate clipping of highlights and will not measure shadow clipping or crushing.

Use this mode for precise and controlled ETTR (Expose To The Right) when paired with the Highlight Recovery feature, a method capable of using partial sensor clipping (ideally one or two channels only) to help potentially extend your sensor's useful dynamic range by letting you eat further into the highlights! This can help drastically reduce noise as well.







[4.65] Encoding Histogram (Direct Log + Direct Preview Only) :

When using Direct Log Video mode and activating the Direct Preview Mode, a secondary Histogram will appear.

This histogram's primary purpose is to indicate not how the sensor received light, but rather how that light will be packaged into your selected rendering settings (Transfer Function, Tonemapping, Contrast, etc.).

This histogram dynamically adapts based on your rendering choices. For instance, boosting the exposure gain will show the lighting increase accordingly, shifting both shadow and brightness details. Different transfer functions will also demonstrate different affinities in how the data is packaged and which areas of the container are utilized.

The two percentage indicators at the top are critical for measuring potential data loss:

•The top left percentage indicates the shadow clipped percentage—the amount of data expected to turn into absolute black after processing and will lose all detail.

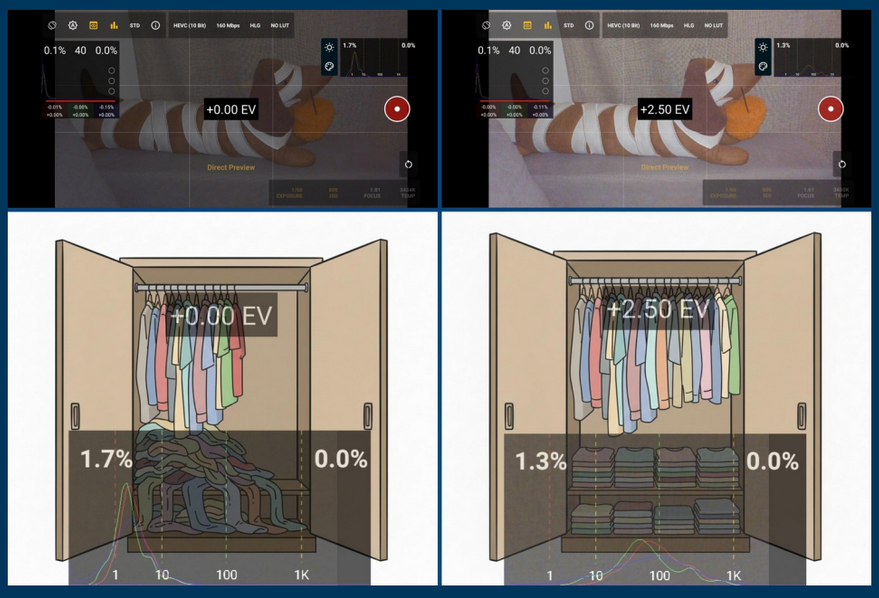
•The top right percentage indicates the highlight clipped percentage—the amount of data expected to turn into absolute white after processing and will lose all detail.

Any value higher than 00.0% in either indicator shows that clipping has occurred (right for highlights, left for shadows).

As is with Histograms, the Red, Green, and Blue lines indicate the amount and intensity of each color channel across the brightness range.

The 1, 10, 100, and 1K (1,000) values along the bottom express the Nit levels (a measurement for brightness) of the scene. The scale shows from 0 to 10,000 with every lined step labeled jumping by x10.

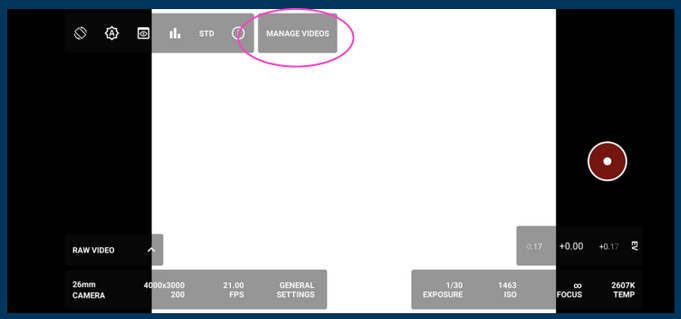
To maximize container efficiency, you ideally want as much of the container used from left to right without any clipping.

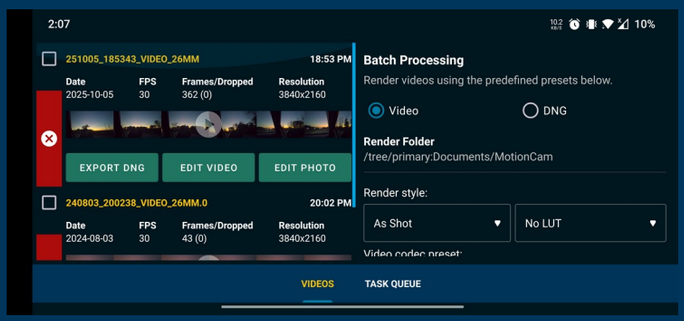


You also want to avoid bunching up all the data in a small clustered region due to the limited tonality available if only partial amounts of the container are used. Think of this container wardrobe and your clothes as the data—you will ideally sort them them using all available space rather than just one side (or else they'll pile up over each other).

This histogram's purpose is to allow you to adjust your rendering settings to achieve this goal, giving you the ability to successfully measure the variance between how the sensor intakes light and how it ultimately compresses it. Harnessing this variance and being able to see what the device has done versus what you need to do is the greatest power of this tool.

[4.66] MANAGE VIDEOS (RAW/BURST/TIMELAPSE Only)





This following section will dive into the MCRAW Video Manager that comes built into the app. Here, you can manage, batch or individually prepare, process and render MCRAW reels into anything from single JPEG extractions, ready-to-go full CinemaDNG sequences, video codecs of your choice, and much more.

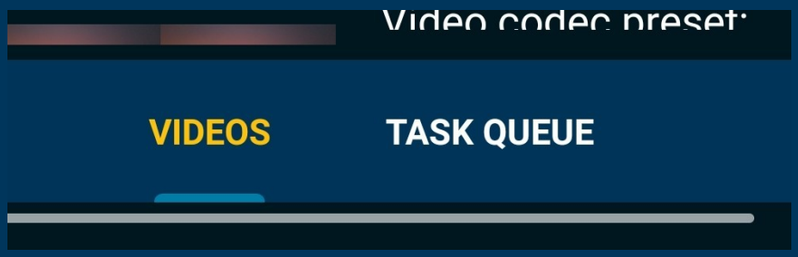
This menu is one of the most powerful app functions as you can effectively handle and process MCRAW format videos, regardless of which device shot them and without the need for an external device or PC.

It's important to note that although Direct Log videos share a similar pipeline to this exporter, their implementation is slightly different; that is to say, in some cases some devices will perform fine with this render but will not take to Direct Log (eg. Samsung Galaxy S20 FE). Additionally, more options such as denoising and additional codecs will be available here that may be too intensive to apply via Direct Log – although some settings are shared.

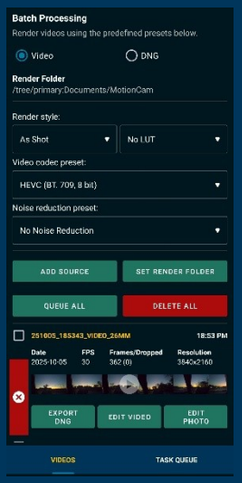
Worth mentioning too is that this interface can work in both Landscape or Portrait modes, with identical functionalities available on both orientations; in case you prefer either layout!

This menu will be split into 2 sections, VIDEOS and TASK QUEUE. The active category will be indicated by Yellow writing as well as a blue underline.

Press on either mode to activate it respectively. VIDEOS will be where you can prepare and initiate the rendering process, TASK QUEUE is where you can monitor rendering time, progress, or finalized projects as well as modify when the app renders accordingly.

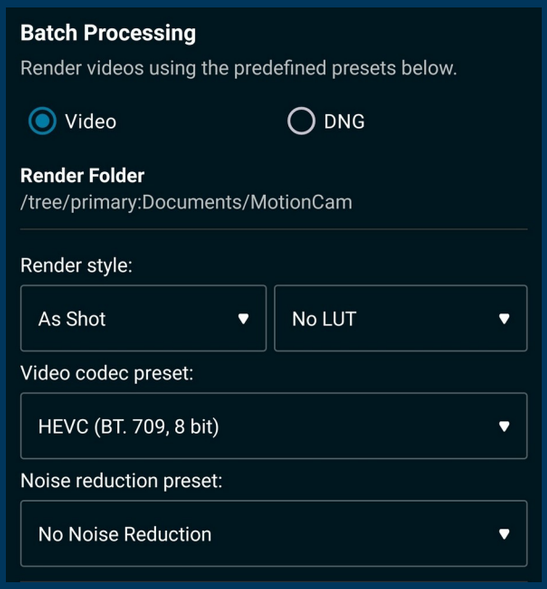


[4.67] VIDEOS Tab (Video Manager): The default start-up mode whenever you open the Video Manager menu. Use this option to display any MCRAW reels found in your designated locations and to also export/render them!



You can scroll up/down to navigate it and explore all MCRAW videos shot with the app.

The app will provide you with the means to either process the reels individually, or as a batch. We will explore both approaches in the following UI element explanations.



[4.68] Batch Processing Interface: The top area of the Video Manager is designed to help you prepare and apply presets quickly and efficiently.

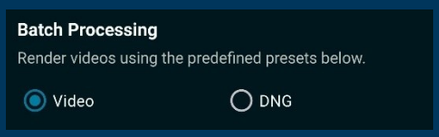
You can then apply these options in bulk to multiple MCRAW clips simultaneously to set them all in the rendering queue with your chosen configurations or adjustments!

It's important to note that the options you see here initially are strictly built-in presets included with the app to get you started.

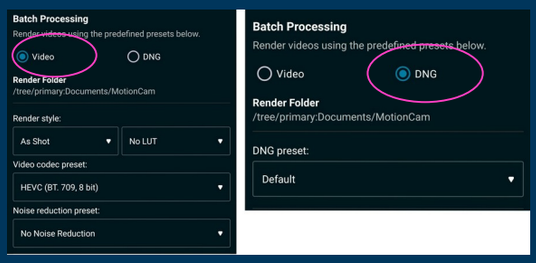
The power of this section, however, lies in your ability to add custom presets of your own!

This includes custom combinations of codec variants and rendering options, such as granular denoising adjustments or even defining how DNG rendered files are interpreted.

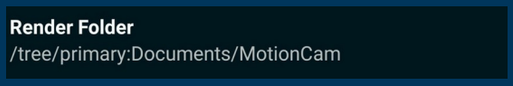
Don't hesitate to move beyond the default options. Explore, experiment, and create presets that further suit your own specific needs as deemed necessary!



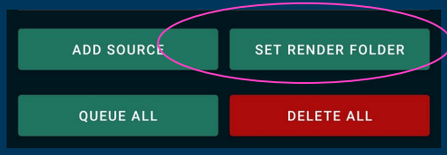
[4.68.1] Batch Processing (Mode): This area indicates whether you've activated the Video or DNG batch preset modes.



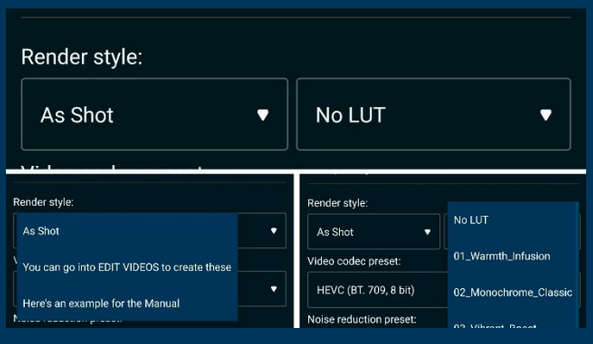
Press on either circle to switch accordingly between video renderer and CinemaDNG sequence rendering Preset options as they will each bring up different options. A filled blue circle will indicate which mode is active.



[4.68.2] Render Folder (Location): This area indicates the file location in which current files the app renders will be saved on.



Can be adjusted via SET RENDER FOLDER option below the Batch Processing Menu area.

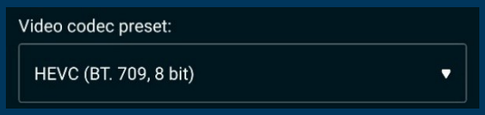


[4.68.3] Render Style: Tap these box areas to quickly and efficiently select presets you've already created under the EDIT VIDEOS editor.

The left box (AS SHOT) is for in-app rendering settings you've altered and saved as presets.

The right box (NO LUT) allows you to select previously added LUTs to be baked-in as well.

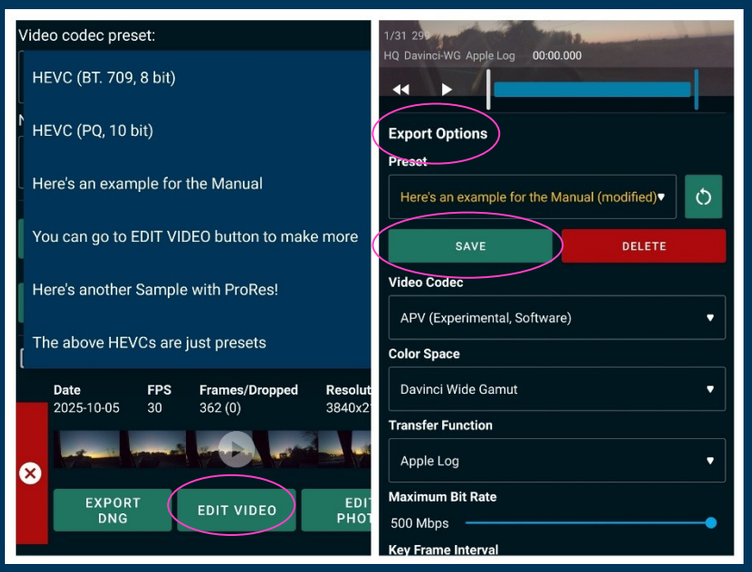
Both options can be used independently. However, keep in mind that if you apply only a LUT its transparency will be applied with 0% by default, meaning it's baked at full intensity and without blending.

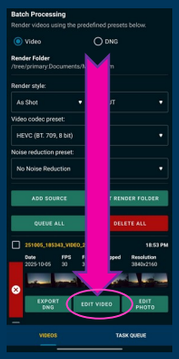


[4.68.4] Video Codec Presets: Use this box to open up the list of built-in app presets as well as the ones you've saved.

This list is often misunderstood and thought to be the only exporting options the app offers, but in reality they're actually just the preset names, not the codecs list itself!

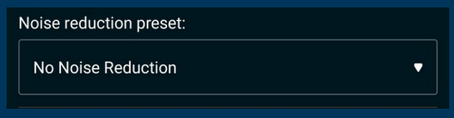
The current preset selected will be shown here. Tap on the box to expand it and see the full list and press on any other preset shown to use it instead.



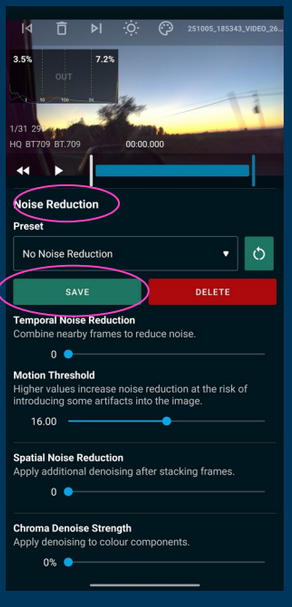


To create more, simply scroll down to view any MCRAW reel and press under any of their EDIT VIDEO option to open the Video Editor. You may then scroll down to the Export Options section to create your presets!

After changing the codec settings, simply press on SAVE and name your new preset!



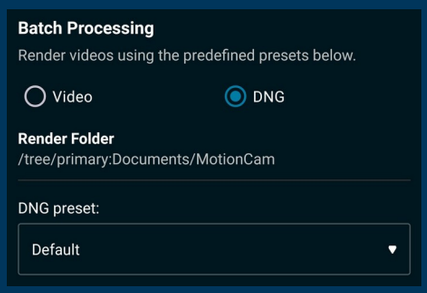
[4.68.5] Noise Reduction Preset: Tap on this box to open the Noise Reduction presets you've already created. Just like the prior option for Video Codec Presets, these are only premade suggestions and you can create your own!



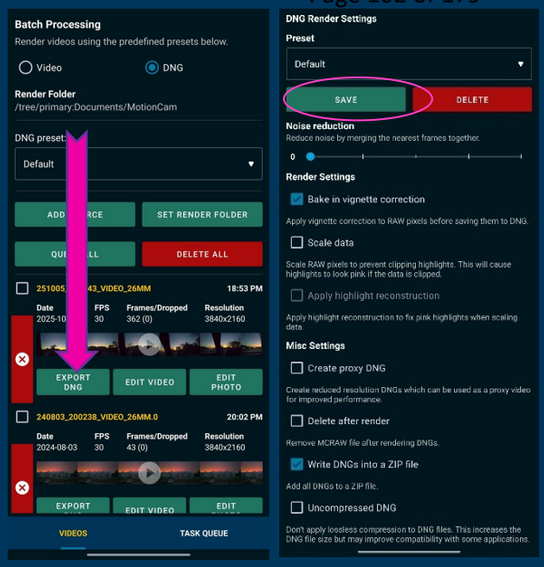
See the above Video Codec Presets steps to also create Presets — however in this case you are looking for the Noise Reduction section. Simply adjust the settings to taste and press on SAVE, then name your new preset!

We will cover each Noise Reduction function later on in pages 119 to 124.

BEWARE: Although using Chroma Denoise Strength is very efficient and quick to render, any usage of the Spatial or Temporal Noise Reduction settings will **significantly** affect exporting durations, so keep that in mind when deciding whether to apply denoising or not.

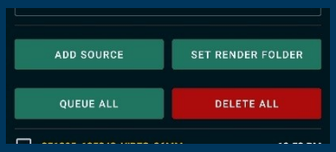


[4.68.6] DNG Preset (DNG Batch Processing Only): Use this box to quickly select any DNG presets for quick batch/bulk exporting of multiple MCRAW reels into CinemaDNG format.

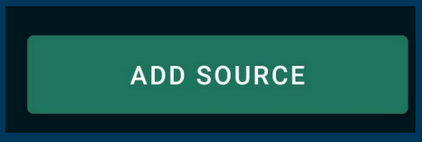


The standard presets are also mostly to apply denoising via frame stacking however you can once again create your own custom ones with further granular DNG data adjustments!

To create your own DNG Presets, simply scroll down to the MCRAW list and press EXPORT DNG under any of them; you will be brought into the DNG Render Settings menu. Simply adjust the settings and frame stacking to taste, and press SAVE to name and store your new preset! We will further explore this DNG Renderer in further detail on page 108 [4.72].

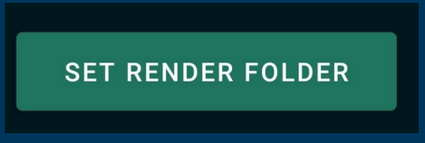


[4.69] File Management: The following button cluster allows you to adjust settings in relation to where the app reads files from, renders them, as well as the batch handling of files.



[4.69.1] ADD SOURCE: This option allows you to designate additional file locations (folders) that the app may look within to scan for MCRAW files (internal device storage or external locations such as SSDs).

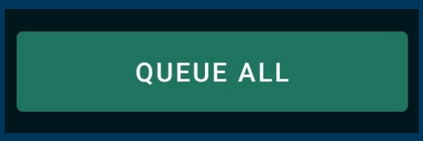
Pressing this option will open up your device's File Explorer. Simply navigate to your desired path of choice (must be a folder).
Note: you cannot use the 'Downloads' parent folder due to Android restrictions however.



[4.69.2] SET RENDER FOLDER: This option allows you to designate the file location under which the MotionCam app will save any rendered MCRAW outputs (regardless if rendered individually or in a batch).

Pressing this option will open up your device's File Explorer. Simply navigate to your desired path of choice (must be a folder).
Note: you cannot use the 'Downloads' parent folder due to Android restrictions however.

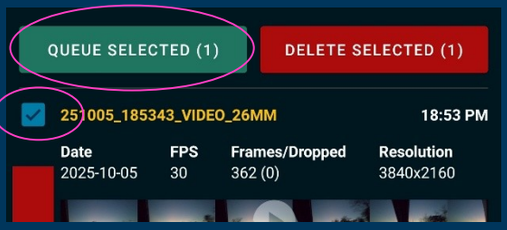
Although Internal storage is the safest option for this, you may also use external storage to save rendered outputs. Do beware however, if you've selected any external storage location that gets disconnected during rendering, this may result in the outputs being corrupted and the session itself will also be interrupted!

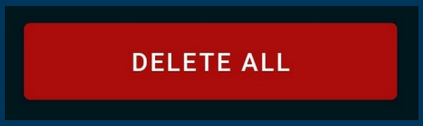


[4.69.3] QUEUE ALL: This option can be used to apply the current Batch Processing presets as selected onto the full list of MCRAW reels that appear in the list below.

This setting should only be used as-is if you intend to batch render everything as it will select all the MCRAW files as found on your designated source locations and will begin a large Rendering Queue sequence.

As an additional function, you can designate which files are batched to be rendered by simply checking the desired reels boxes above them individually. The **QUEUE ALL** button will then change to **QUEUE SELECTED (# OF FILES SELECTED)** which allows you to batch only the selected group.

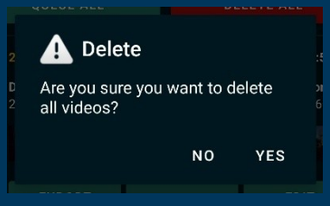




[4.69.4] DELETE ALL: You see the red color? Do not press the red button… Ok, so actually this button is self-explanatory; if you press it, it will Delete All – yes, ALL of the MCRAW files that are currently shown per your designated sources.

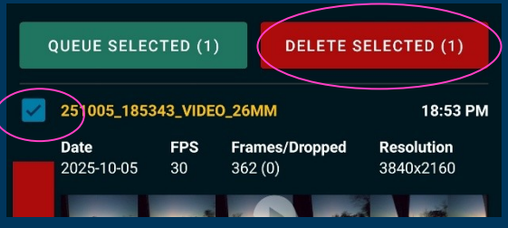
Only use this button if you've ensured that you no longer need any of the MCRAW files shown as once deleted, there's zero ways to recover the MCRAW files.

NOTE: This does not impact any existing rendered outputs, this option is strictly for MCRAW reels detected.



Great! You've pressed the red button… Well, you can rest assured you still have a fail-safe prompt in case you accidentally pressed it. You will be asked for a final confirmation of the deletion intent. Jokes aside, once you accept it by pressing YES, understand that the action finalizes and all MCRAWs will be permanently erased.

Lastly, this option will also change if you've selected specific MCRAW reels by checking their boxes. Simply select the desired group that you intend to delete and upon selection of them, the button will change to **DELETE SELECTED (# OF FILES SELECTED)** to indicate one or more files were individually selected.

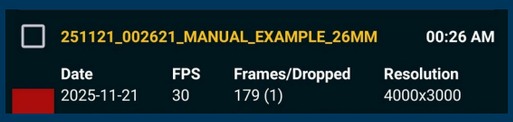


[4.70] MCRAW WORKBENCH: Use this area in the Video Manager to preview and manage individual MCRAW files/reels found in your current file source location.

Each file will be independently shown with its own Workbench allowing you to read details regarding the session and parameters at the time of recording.

You can also use the in-app tools on the lower area to export full CinemaDNG sequences or to open the built-in editors! We will explore these in the area below.

You can also selectively batch MCRAWs by tapping and checking the top left box which enables the ability to select specific reels only.



[4.70.1] Session Review: Each MCRAW file will list the session details. The following information will be shown to quickly identify both the content and when it was shot, as well as some essential capture data.

The file name is the writing in yellow. To understand the app's naming scheme, see page 52 [4.22]. You can also observe the time the file was recorded at to the right of the name in white (AM/PM). You can also see the recording resolution listed in Width*Height values.

There are important things to note regarding FPS (Frames Per Second) and Frames/Dropped counters however.

FPS
30

The Framerate itself will ideally be what you've selected, however the app will round to the nearest stable framerate achieved and doesn't reflect the framerate slot selected, but rather the most stable one.

If you've selected for example 60fps, assuming it runs fine but with many frame drops, you may for example see this counter report 59 FPS in such case. Additionally, using longer exposures that override the framerate may further deviate this value.

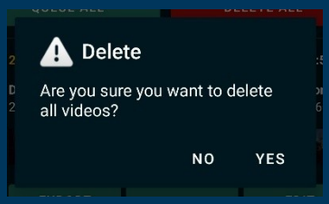
Frames/Dropped
179 (1)

As per Frames/Dropped, the left number indicates how many image frames you captured in total during the session and the number in parenthesis indicates the amount of dropped frames – or in other words, the amount of frames that couldn't be retained and were completely lost.

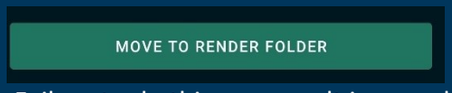
Dropped frames may occur due to a multitude of causes, such as performance or capture issues, bottlenecks , or thermal limits (potentially even misconfigured settings).

[4.70.2] Single MCRAW Delete: We've already explained what the red DELETE buttons do in the Batch settings, however the red X button that shows to the left of every MCRAW reel allows you to specially delete the targeted file that's beside it only.

This will perhaps be the most used deletion method to work with. If you select it accidentally however, don't be alarmed, a final confirmation prompt to verify the intention will appear.



All deleted MCRAWs are permanently wiped, so ensure you've reviewed or confirmed the file first since once you accept, there's no going back.



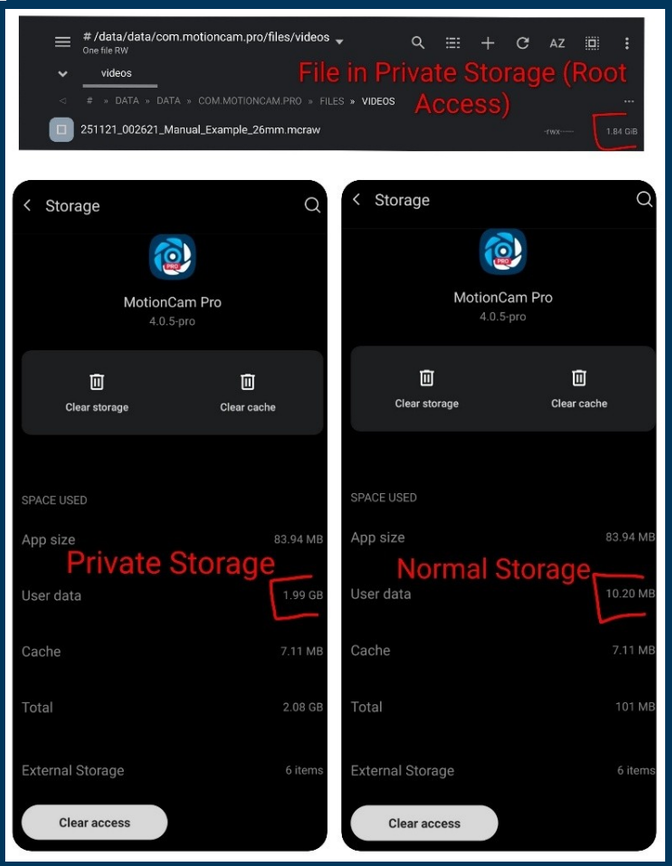
[4.70.3] MOVE TO RENDER FOLDER: Press this option to transfer the MCRAW files out of the app's private storage and to bring into the ordinary designated render folder. Failure to do this may result in complete file loss if the app gets deleted.

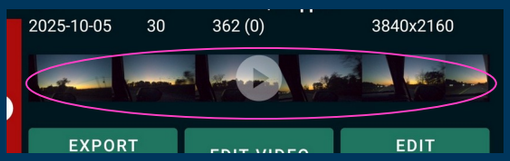
This area will ONLY appear if the app is currently storing any MCRAW files within the private storage area – no, we're not talking about degenerate scumbag folders; Android apps are allocated space within the device root folders, or otherwise, the area where the literal installation and program files are kept.

As discussed in page 57 [4.27.2] (***Capture to MotionCam Private Storage*** setting), the app can be made to instead write MCRAW file captures within this 'private' area at the system level, only accessible otherwise via root privileges and root explorers.

The benefit of using this option will depend per device, however in some cases it will provide faster storage I/O and eliminate system write speed bottlenecks. The downside is you then need to offload them using this button which is not an instant process

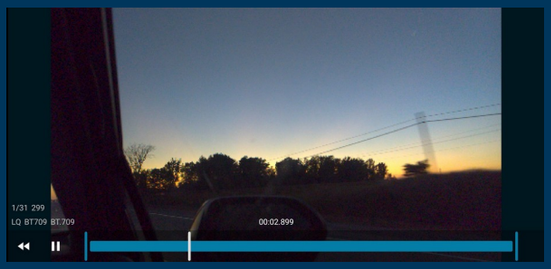
You can observe this by the app User Data consumption if you should use Private Storage; It will quickly make PUBG look like a Nintendo 64 file sizes game.





[4.70.4] MCRAW Preview: This area provides you with 6 image frame previews from your sequence, split in equal intervals during the video to show you what was captured within the clip.

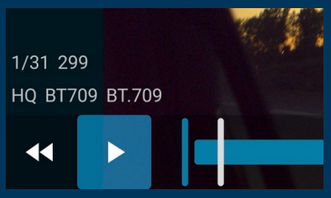
You can use this to identify the specific content at a glance, however you can also tap on the Play icon to open the in-app MCRAW Viewer!



[4.70.5] In-App MCRAW Viewer: Upping tapping on the MCRAW preview area's play button, you will open the app's native viewer.

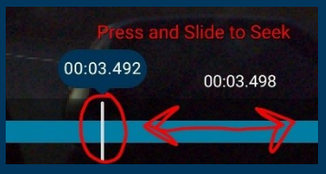
The files as seen here are not inherently at the maximum quality to ensure smooth playback (future app versions will solve this however this is the current mode).

Although they may not fully reproduce the quality or colors actually present in the files, this player will allow you to view MCRAW files as a normal video and play them back to assess them as needed. The preview will be with BT.709 color space and transfer functions both for displaying compatibility.



At the bottom left, you will have a ⏸ / ▶ button respectively to pause or play, and a rewind (⏪) button which you can use to fully rewind the playback of the file.

The top numbers row indicates the exposure/shutter speed to the left and the ISO reading to the right. The bottom row shows the color space and the transfer function applies to the preview, as well as the quality of the image shown (pause to increase to HQ and playing will decrease it to LQ).



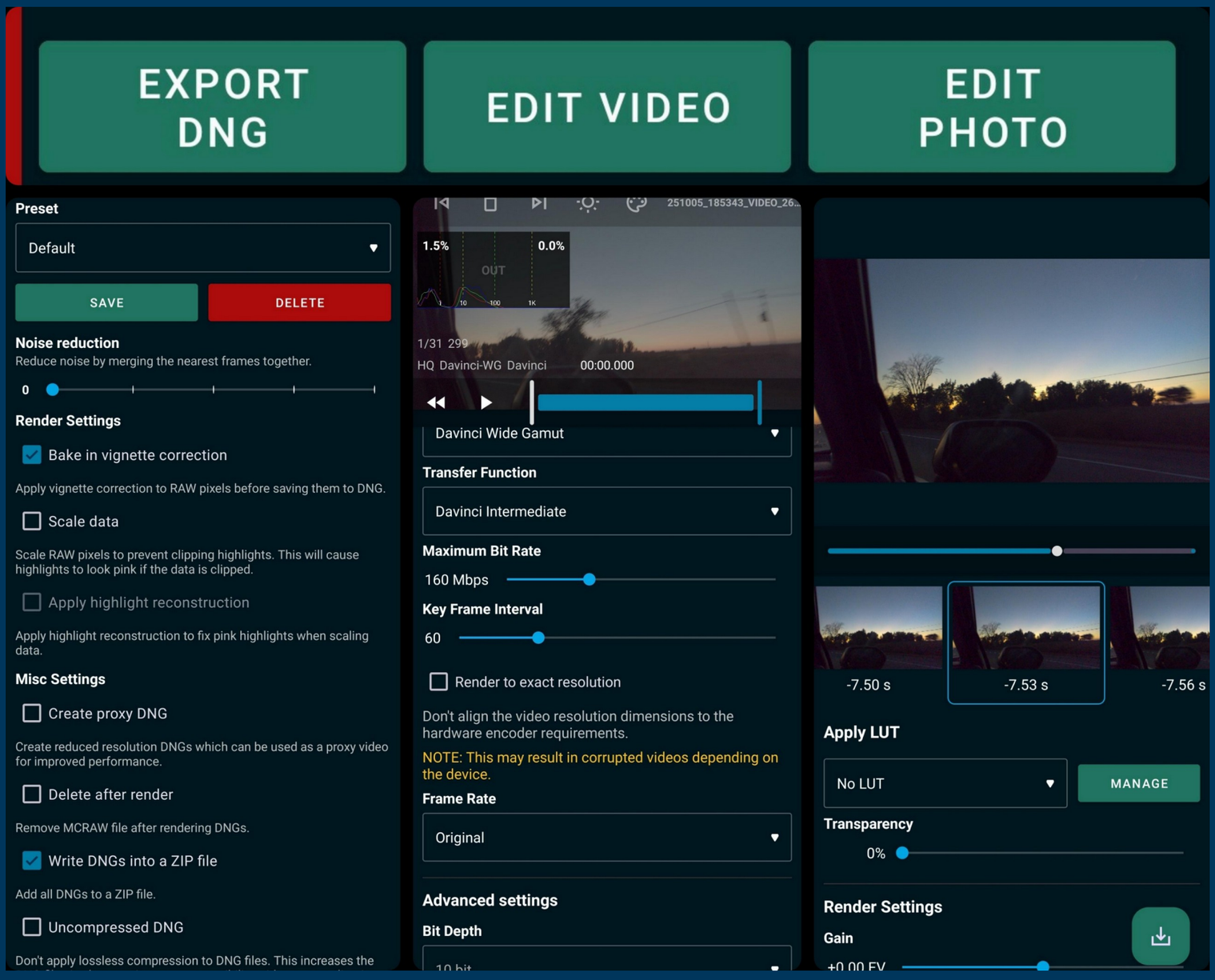
You can also hold the white vertical line on the progress bar and slide it left/right to navigate and seek a special video moment.

Additionally, the specific video time elapsed will show under white numbers and the current image being seeked when moving the white line will also be shown under a blue bubble to allow precise navigation of the footage.

Lastly, minor red indentations will also show above the progress bar to alert you of areas in which the app believes a frame drop occurred.

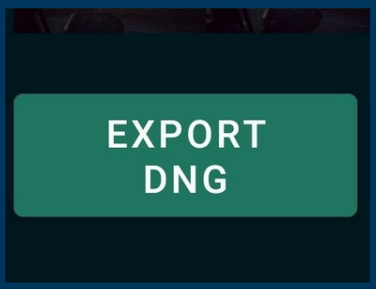
Although the MCRAW Workbench will advise of precisely how many, this tool will show you exactly where they occurred within the footage – a great tool to assess if the frame drops are going to be a concern or if they're spread out/clustered in a specific area/moment in the footage. As an example, if a bunch of drops occurred for any reason, but only during a section that you didn't need, then the file may still be completely usable.

[4.71] IN-APP MCRAW RENDERING AND EDITING



This next section will explore the often ignored, yet ultra powerful in-app rendering tools for DNGs, Videos and Photos (see their respective menus below them in the above example).

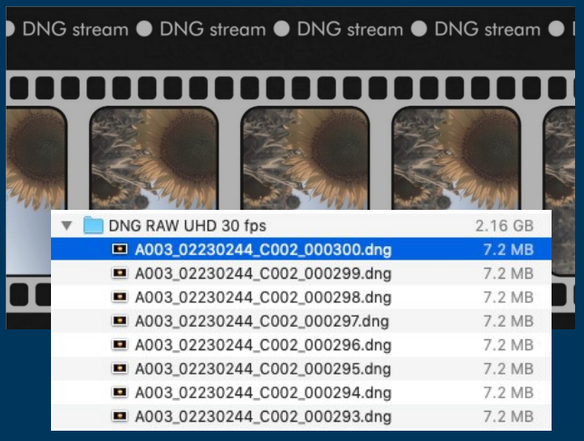
As you've probably learned through the manual, the app captures in .MCRAW format, a data lossless RAW video format. You can choose to offload it for processing at a PC, however if you also prefer a mobile workflow or want to leverage the powerful app rendering engine, the following buttons will provide you with all the tools required to process MCRAWs locally!



[4.72] EXPORT DNG (DNG Render Settings): Use this menu to individually select a MCRAW sequence and the settings it will use before exporting it into a DNG sequence.

You can also use this menu to create presets for batch DNG rendering.

Although it's not explicitly stated, this button could be technically referred to as the 'CinemaDNG' Converter, since it exports a version of the MCRAW file's data into a classical cDNG sequence.



For the uninitiated, CinemaDNG (cDNG) is an open-source RAW video format that was created by Adobe. It consists of simply stitching together a sequence of DNG photos (Digital Negatives, also an open source RAW photo format created by Adobe) into a video, alongside with audio to go with.

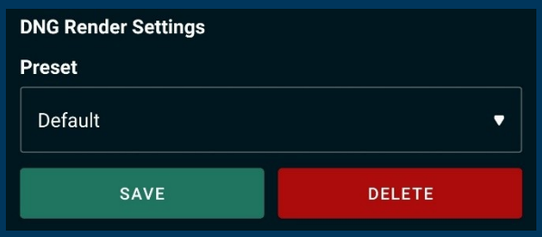
This essentially would look like a folder full of individual RAW images alongside an audio file.

Each image would be labeled in a sequential fashion that lets your editor understand which order they follow and this can then be played as a video with audio… RAW video! This really is like those classical film days, except digital!

Although widely supported, cDNG is rather inefficient as a format (as far as space usage and write/read speeds are concerned) and cannot compress anywhere near what MCRAW can.

It also provides the added bottleneck of having to move over those folders which oftentimes contain hundreds/thousands of DNG files!

Nevertheless, if you want easy compatibility, this will be your route to use! The app's DNG sequence renderer above will give you the tools to simplify your workflow and modify/enhance the data pushed into the DNG containers; we'll explore that in the next few pages!

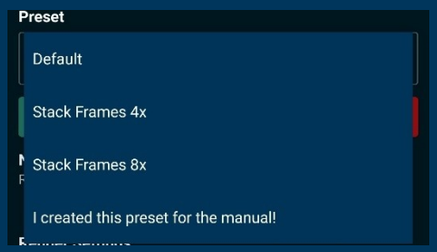


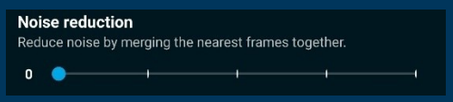
[4.72.1] Preset (DNG Export): This setting allows you to quickly select or create rendering presets to be used for exporting/rendering DNG sequences.

Simply tap on the box to open the presets already created. Two existing frame stacking presets already come built-in, however you may add or delete more.

To save a new preset, simply ensure you've finalized all your settings in the rendering menu, then press the SAVE button above. You will then be prompted to name your new preset!

To delete a preset, simply select it and then press on the DELETE button on the red box.





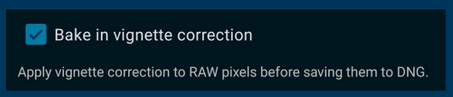
[4.72.2] Noise Reduction (DNG Export): Use this option to apply noise reduction in the form of the app's signature Temporal Frame Denoising algorithm. In short, this option uses a custom algorithm that analyzes the image frame, then proceeds to use the data in the immediate frames before/after it to evaluate what is noise and what is not.

You can keep it at 0 for an unaltered DNG, and you can increase in increments of 4 by sliding to the right. A higher frame merge may potentially induce movement artifacts if the scene is too dynamic however. Additionally, this option will heavily impact rendering times; use as needed since not all scenes require it

What's that? You thought DNGs had to be pure and unaltered data? We've got something to tell you then..!

The advantage of denoising at the DNG level is in having the full RAW data from the frames before/after for denoising rather than compressed images which fundamentally have less precision. This algorithm is quite robust to movement and dates back to the MotionCam app's life as a Photo app only! Indeed, this is a relic of that era. A future upgrade is coming in newer versions however (far less resource intensive as well).

As a rule of thumb, to determine the effectiveness of Temporal Frame Merging, for every doubling of frames stacked, you reduce noise by half – this means it will require exponentially more frames to appreciably improve the effectiveness as you push higher.



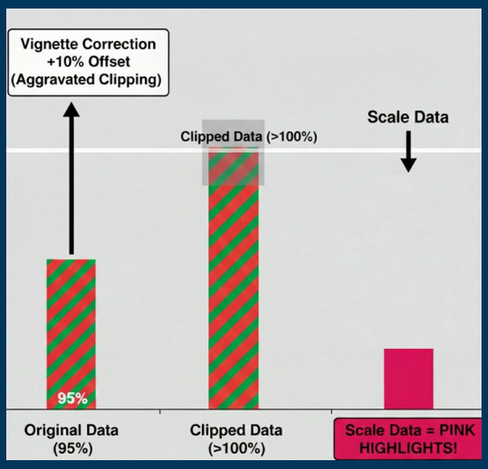
[4.72.3] Bake in vignette correction (DNG Export): This setting allows you to irreversibly bake the vignette correction offset into the RAW data. It is on by default.

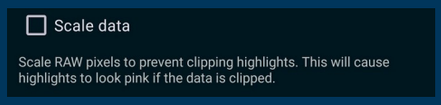
Ordinarily, this correction is bundled into the file as part of the EXIF metadata (for more details, see the vignette correction explanation on page 50).

Baking it in ensures you don't have to worry about the bundled data not being applied correctly by an external editor, or being stripped off accidentally. However, this process comes with two important trade-offs: you can no longer decide when you want to disable the correction if you've discarded the MCRAW, and you also reduce the effective dynamic range of the image, particularly near the edges of the frame. Let's explore why, and the solutions

NOTE – we will use percentages for simplicity's sake, but in reality we're talking about the white levels and their tonality values available to describe brightness. Vignette gradients are also complex things with all sorts of weird oval patterns and such. Offsets aren't merely perfect circles that get darker linearly //////

If you are Exposing To The Right (ETTR), and your sensor reads a data value near the edge that pushes it—say, to 98% of its maximum dynamic range—and the vignette correction then adds an extra 10% boost; you've now digitally pushed the data beyond 100% of what the sensor could capture. This is a form of digital clipping caused by the added offset. This issue leads us directly to the next solution: scaling the data back…



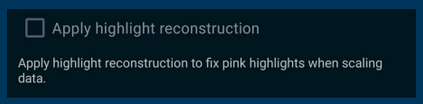


[4.72.4] Scale Data (DNG Export): This setting only becomes available when you check the 'Bake in Vignette Correction' setting above. It allows you to forcibly scale down any data that may have clipped due to the vignette correction while pushing the exposure, and can also be used to further manage sensor dynamic range in extreme scenes.

As we've covered above, your dynamic range can be decreased because the offset correction digitally pushes data that was near clipping into the clipping range. Imagine some pixels are at 95% capacity, and a 15% correction boost pushes them to 110%. Your sensor will clip the data once the digital correction occurs and it touches any value ±100%.

Scaling back the data essentially forces the app to bring any clipped data back down if it observes clipping occurs after offsetting, before the correction finalizes and discards the data. To follow the example: if the app realizes the data was pushed beyond 100% after the correction, the data will be forcibly brought back to, say 80%, wherever clipping may have happened. While this is not quite the original value, it forcibly unclips the data and further it lower too for good measure so you don't completely lose it, thereby restoring some of the useful dynamic range otherwise lost.

Be aware that this option may potentially induce pink/magenta highlights for clipped areas (most observable during ETTR). This is because if one channel clips completely, the remaining two unclipped channels are now unbalanced. For example, if the Green channel clips but Red and Blue remain, the area will be revealed as a magenta mix because the Red and Blue values now dominate and the necessary color information from the third channel is missing. This is a problem – and we like problems here. You can fix this with an advanced editor; or as you may have guessed, we've got something for that too; enter Highlight Reconstruction… (next page)



[4.72.5] Apply Highlight Reconstruction (DNG Export): This setting only becomes available if you've activated 'Scale Data', which itself requires 'Bake in Vignette Correction' options that were both mentioned prior to be enabled.

Use this to algorithmically attempt to reconstruct the clipped highlight data whenever one or two color channels have experienced clipping.

This feature is powerful because it completes the workflow. As we learned, forcing the app to Scale Data back (unclip) when a single channel is blown out creates that undesirable pink/magenta tint because the color is unbalanced.

Applying Highlight Reconstruction steps in at this exact moment, using the remaining data from the two unclipped channels to mathematically generate the missing information for the third (clipped) channel. This effectively corrects the color balance and removes the magenta cast.

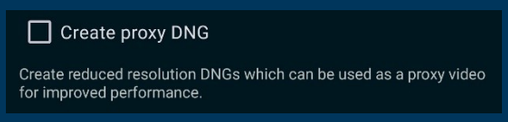
This ability to recover and correct clipped channels allows you to further push the useful dynamic range of the sensor.

By using this trifecta of the three prior settings, you can intentionally expose further (e.g., utilizing ETTR—Expose To The Right) to reduce shadow noise, knowing that this powerful RAW pipeline will manage and restore the highlight detail and color fidelity, even if the clipping occurred due to normal scene conditions.

This highlight recovery method is applied directly at the DNG/RAW data level, distinguishing it from the video-level highlight recovery applied during the subsequent codec compression stage. It is best used to mitigate vignette correction losses unlike the video highlight reconstruction.

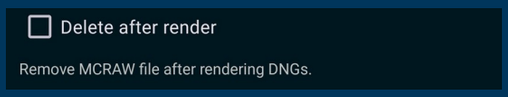
For more on highlight reconstruction you can jump back to page 81.

Misc Settings (DNG File Management):



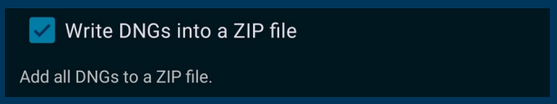
[4.72.6] Create Proxy DNG: Toggle this setting to create 'proxy' versions of your rendered DNG sequence.

Proxy files are mini versions of reduced quality and resolution, intended to be used as temporary stand-in replacements for your main sequence files. This allows you to edit your footage with much lower processing requirements compared to using the full-quality files. Keep in mind this will take slightly more storage space, as you are creating a secondary, lightweight version of your reel specifically for editing efficiency.



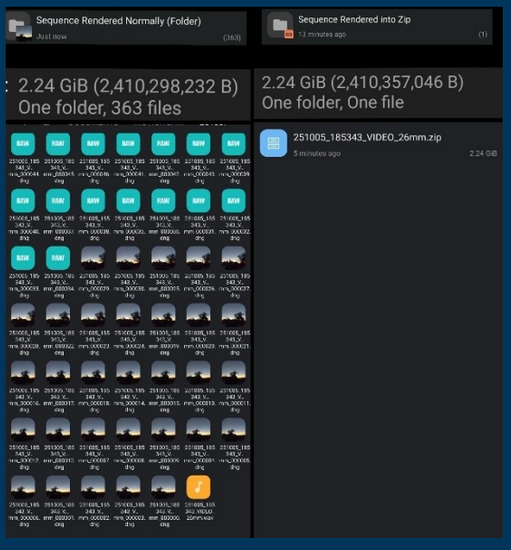
[4.72.7] Delete After Render: Toggle this option to have the app automatically delete the original MCRAW file data upon successful completion of the DNG sequence rendering.

Ordinarily, the app's MCRAW files are retained after rendering, meaning you would have two RAW video versions of your capture. Use this option to immediately reclaim space by only keeping the converted CinemaDNG file outputs. Beware, however—this option irreversibly deletes the MCRAW file, meaning once you've rendered, there is no going back or altering the DNG rendering parameters (they become final).



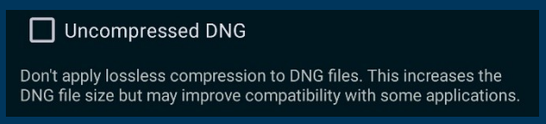
[4.72.8] Write DNGs Into A Zip File: Use this setting to bundle your rendered DNG files into a single ZIP file to allow for quicker file management and drastically improve transfer speeds. On by default.

Normally, the files would appear in a folder containing hundreds or even thousands of individual DNG image files and an audio file, as shown in the left side of the accompanying image. While this method is straightforward for storage, moving around that many small files is significantly slower, due to the way computing devices manage IO (Input/Output), than simply moving one large file.



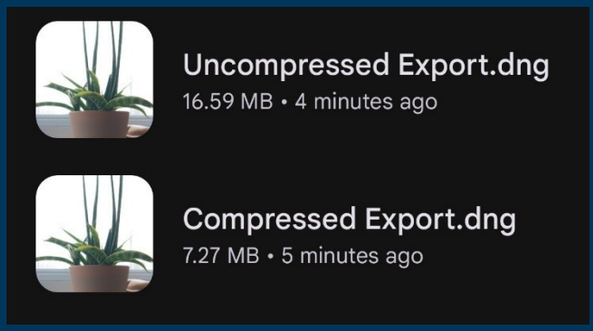
This option mitigates that inefficiency by outputting the entire sequence as a single, tightly packed ZIP file (as seen on the right side of the image). You can then simply transfer this single ZIP file and unzip it on your PC. While this adds the necessary step of unzipping, the time saved during the transfer due to avoiding the IO bottleneck of moving thousands of small files can be substantial.

This option helps to mitigate the inherent clunky inefficiency of the CinemaDNG format, which we must use because most editors cannot directly work with MCRAW files (more on this later).



[4.72.9] Uncompressed DNG*:* This option allows you to export DNG files with no data lossless compression applied. We do not generally suggest using this method, but it exists as a necessary auxiliary option.

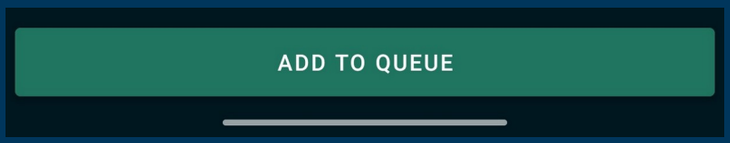
Here is why: Android devices often use DNG containers topped at 16-bit as the maximum (even if the sensor itself shoots at a lower depth, like 10-bit data within a 16-bit RAW_SENSOR container; see page 56 [4.25.2]). MotionCam utilizes data lossless compression to ensure only the actual data is retained. For example, if a 16-bit container only holds 10 bits of sensor data, the wasted 6 bits of empty padded space are efficiently discarded.



Furthermore, the compression algorithm can further shrink the file size lightly based on scene characteristics; the better the lighting and the lower the noise, the more effectively the compression can work. Although nowhere near as efficient as MCRAW compression, it nevertheless gives a noticeable space reduction.

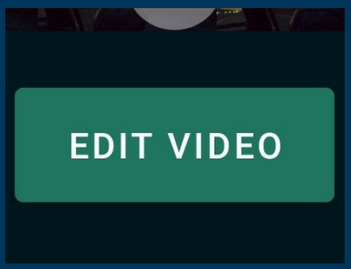
However, some editing/processing software (looking at ya, Adobe Premiere) absolutely detest this type of intelligent compression and requires the full, untouched, and uncompressed container to function correctly. This is the only reason this option is available: in the unfortunate event that an issue like this occurs with your editor of choice, you can provide the fully uncompressed files instead.

To visualize the space savings: If the RAW data is ABCDEFGHIJ but is put into a padded 16-bit container ABCDEFGHIJ000000, MotionCam's compression saves space by trimming off the unused padding (000000). Uncompressed DNG forces the output to include that padding.



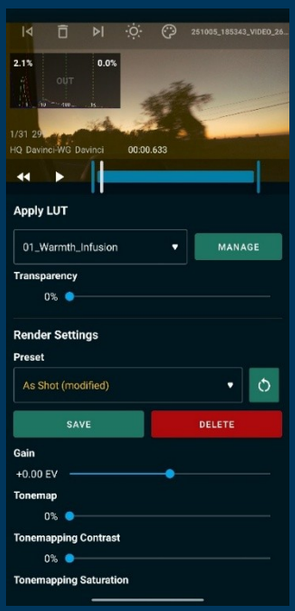
[4.72.10] ADD TO QUEUE (EXPORT DNG): Once all the settings and/or presets have been selected, press this button to send the MCRAW file into the Task Queue for rendering.

Pressing this button will immediately return you to the prior Video Manager menu. You can then monitor the rendering progress and time remaining of the individual reel task (jump to page 128 [4.82] for the general explanation of the task menu).



[4.73] EDIT VIDEO / Offline Renderer (MCRAW Render Settings):

This menu allows you to individually select a recorded MCRAW sequence and define the precise settings it will use before exporting it into a compressed video file. It also provides the ability to create presets for batch codec rendering.



Otherwise known as the 'Offline Renderer', the 'In-App Editor', or the Video Exporter, this is perhaps one of the most powerful and commonly overlooked app functions, hidden in plain sight – this menu provides you with an entry level RAW editor and processor in your pocket!

The core advantage of this feature is it provides a full, entry-level RAW editor and processor in your pocket. Because you are dealing with the file after capture, you are no longer bound by real-time performance. This means all settings will operate at the highest performance available for both rendering quality and speed.

Furthermore, the amazing thing about MCRAW containers is their ability to hold the complete, lossless data as received by the system and sensor—essentially, no modifications have been applied to it beyond the lossless compression. Unlike cDNG (CinemaDNG), which requires certain container accommodations (such as deciding to bake in vignette or reconstruction/scaling of data), MCRAW remains pristine and fully flexible. Therefore, you incur zero quality penalties by using this editor.

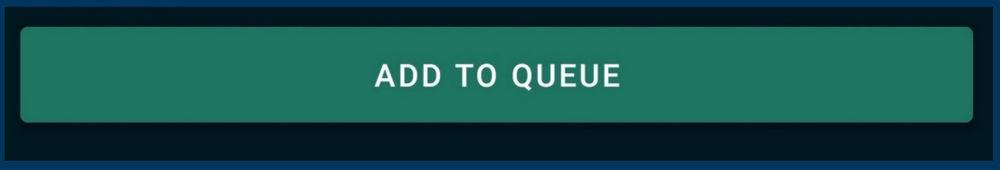
Not only can you view and trim your MCRAW reels, but you can utilize the same monitoring tools (like Histogram and exposure monitoring) found in the Direct Log Video mode. On top of this, you may also observe even more granular settings and additional capabilities otherwise not present for Direct Log mode due to the performance tax they would incur. This includes more advanced codecs (such as 4:2:2 APV and 12-bit options) and additional Noise Reduction/Denoising capabilities (like temporal frame stacking).

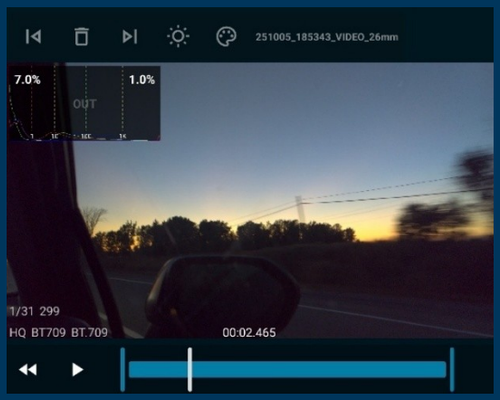
This section will cover the various options provided. Please note that although the rendering pipeline for compressed videos is slightly different than the Direct Log pipeline, they share many similarities. For this reason, and to avoid redundancy, you'll be referred back to the prior Direct Log encoding parameters already documented for repeat settings. Additionally, we would actually encourage you to try any settings that appear here under the Direct Log mode in the Direct Preview viewfinder, as this will provide a superior experience for calibrating presets or visualizing their effect on your capture.

🛑 *CRASH COURSE REMINDER*: If you are still unfamiliar with what encoding or video editing entails (you didn't skip ahead, did you??), you are strongly encouraged to JUMP BACK TO THE FOUNDATIONAL CRASH COURSE OF THE GUIDE before proceeding, as this section will otherwise be difficult to understand.

FINAL NOTE BEFORE WE BEGIN:

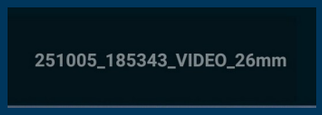
To finalize any of the following changes, crops, edits or codec selections you'll apply, you must navigate to the lowest part and press the following ADD TO QUEUE button!! This will send your outputs to the Queue!





[4.74] Offline MCRAW Viewer (Editor): This area will dominate either the top or left (depending on Landscape or Portrait mode orientation) and will allow you to see the MCRAW file data as rendered in a low quality texture preview.

Although work is underway to allow you to see the content shot in real-time with full quality, this current release does not provide with the actual output under this preview, rather an approximation of it. Do keep this in mind as you edit with the following settings!



[4.74.1] Offline MCRAW File Name: Shows the File Name of the current MCRAW selected.

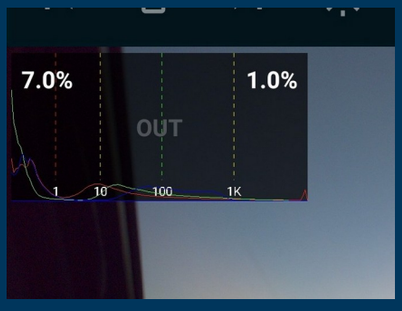
[4.74.2] Navigate Backwards/Delete/Navigate Forward (MCRAW list): Use these options to quickly dispatch and navigate through MCRAW files without needing to exit the Offline viewer.

The Skip Back ⏪ button navigates backwards in the MCRAW list. The garbage 🗑️ button can be used to quickly delete the current MCRAW if you observe it's somehow undesirable. The Skip Forward ⏩ can be used to navigate forward in the MCRAW list.

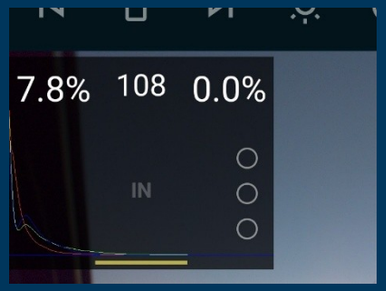
MCRAWs are sorted by capture date on the main list. See the right example to visualize it!

[4.74.3] Offline Viewer Sensor Clipping/False Color Overlays: Tap on the Shining Sun (🔆) icon to change the viewer into Sensor Clipping mode and press on the painting palette (🎨) icon to bring up the False Color overlay mode.

Both of these modes do offer greater accuracy for monitoring purposes versus the output displayed in this renderer. Refer back to Page 88 to learn more about both of these modes in comprehensive detail.



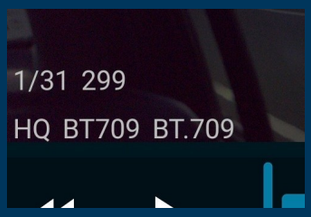
[4.74.4] In/Out Offline Histograms (RAW/ENCODER): You can use the same RAW and Encoder Histograms offered by the app to assess the captured data, and the encoding container allocation thereafter as well.



Tap anywhere on the histogram to alternate between **IN** = RAW Histogram and **OUT** = Encoder Histogram modes accordingly.

To learn more about the RAW histogram (**IN**put) jump back to page 82 [4.30]. To learn more about the Encoder Histogram (**OUT**put), jump back to page 97 [4.65].

Do note however, although the extended RGB channel data is not available on the offline renderer, the other tools on it remain!



[4.74.5] Session Playback Information: This area indicates the current information about the playback

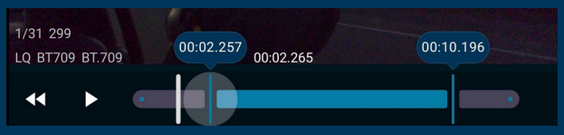
The top row shows the exposure duration on the left (eg. 1/31s) and the sensor ISO (eg. ISO 299) that was used in the image frame being previewed.

These counters are Dynamic and will reflect changes throughout the session if you changed your exposure through the capture (whether you used Auto or Manual exposure controls).

The bottom row shows the preview settings applied. Unlike the standard In-App MCRAW Viewer accessible in the MCRAW Manager menu, this viewer can actually change the settings applied to the preview rendered.

The HQ/LQ designates the quality mode applied. Pausing will apply HQ mode for still frames and increase resolution for better textures; playing or navigating the reel by sliding will revert it to LQ mode to enable smoother playback.

The middle and right values both indicate the Color Space (Middle) and Transfer Function (Right) and will change as per your active selections and the image will visually vary to reflect them accordingly.



[4.74.6] Playback Progress/Trim Bar: The below area of the player is an essential tool to both navigate MCRAW reel content and trim the output you plan to export— great when you don't need the full file and will cut render times!

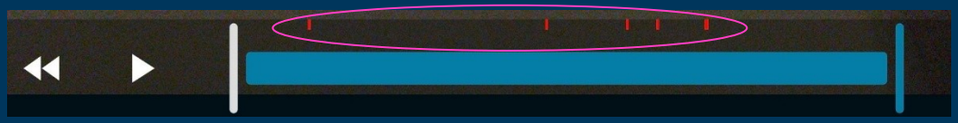
You can hold on the vertical blue lines and slide them across to modify the start time (leftmost bar) and the end time (rightmost bar) of the exported output. You can observe the specific start/end times of the trim selected in the blue circle clouds above the bars themselves, which state the exact time of trim specified down to the millisecond.

Unless manually trimmed, the files will render in full duration/without trim by default.

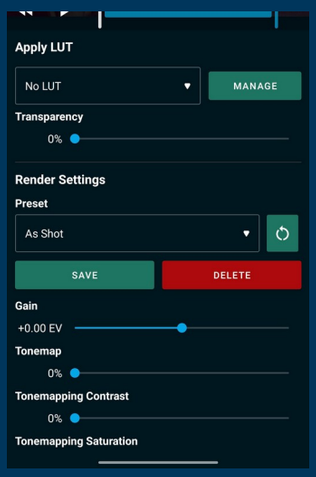
The ⏪ button can be used to rewind the playback to the start of the recording and the ⏯ button can be used to play/pause the sequence playback. The vertical white line indicates the playback progress and the elapsed time in the sequence will be shown on the white numbers above the bars that remain static (also precise to the millisecond).

You can also quickly identify frame drops/dropped frames events with the progress bar as they will be flagged in the form of red indentations above the sequence. This shows the exact moment the app believes a dropped frame occurred; do keep in mind, although it is very accurate, it's not 100% precise – this is to say, in some rare cases you may not have had a frame drop and it may potentially show them anyways. Nevertheless, it remains an excellent tool for quick analysis.

Ultimately, you will be the judge of the impact if any, and whether the footage is usable or not.



of

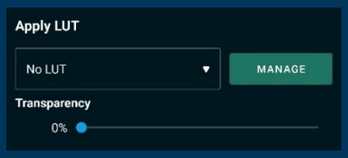


[4.75] Offline MCRAW Editor (Editor): This area will dominate the bottom/right side the screen depending on the display orientation (Portrait/Landscape mode).

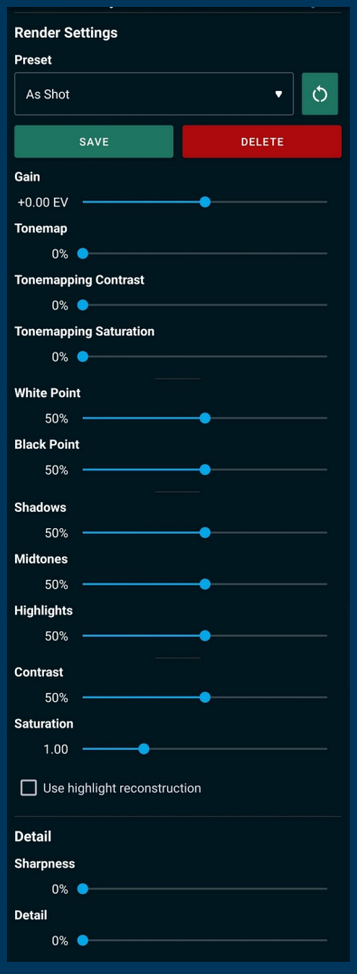
Use these settings to apply quick edits to your MCRAW reel's compressed codec output. Be advised that Render Quality is set to High and will not decrease.

Highly similar to the Direct Log Video mode settings, this mode will also present additional capabilities as mentioned prior, such as additional Denoising and Codec variations, as well you can even modify the playback framerate!

BE STRONGLY ADVISED THAT SOME SETTINGS, SUCH AS DENOISING OR INTENSIVE CODECS, MAY HEAVILY LENGTHEN EXPORTING TIMES!!



[4.75.1] Apply LUT/Transparency (Offline Render): Identical in its operation to the Direct Log mode counterparts as covered on Page 67 to 82, use this setting to apply a LUT and then select the blending/feathering intensity of it that is applied to your video output.



[4.75.2] Render Settings/Presets/Sliders (Offline Render): Also identical to the settings in Direct Log Video mode, use the Presets box to quickly select previously created or customized rendering settings for the below sliders and options available. Please refer to the following pages for all prior options to see comprehensive explanations and illustrations respectively…

GAIN – Page 66

TONEMAP – Page 67

TONEMAP CONTRAST – Page 67

TONEMAP SATURATION – Page 68

WHITE POINT – Page 68

BLACK POINT – Page 69

SHADOWS – Page 69

MIDTONES – Page 70

HIGHLIGHTS – Page 70

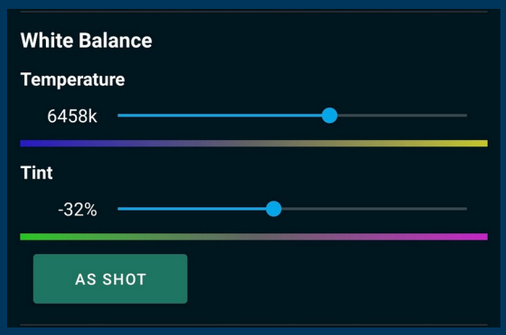
CONTRAST – Page 71

SATURATION – Page 71

HIGHLIGHT RECONSTRUCTION - Page 73

SHARPNESS – Page 72

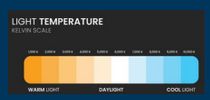
DETAIL – Page 72

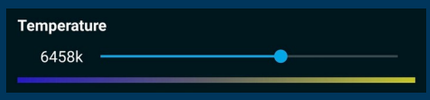


[4.75.3] White Balance (Offline Render): As covered previously, White Balance is often a parameter baked directly into encoded video footage based on the Color Temperature set at the time of capture.

Here is where the inherent difference with the Direct Log Video mode shines through: because MCRAW is not yet encoded, the White Balance is stored only as a metadata guideline. This means it is bundled data indicating what you originally 'set' but is not yet hard-baked into the image data. Consequently, you can tweak this section with complete impunity!

AS SHOT (White Balance): Use this button to reverse the changes below back to the default values as per the captured metadata!

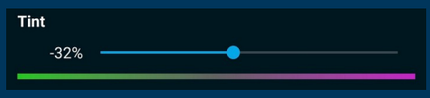




[4.75.4] Temperature/Color Temperature (Offline Render): Use this slider to alter the Color Temperature of the MCRAW export output you will render, running on a scale from 1,000 Kelvin to 10,000 Kelvin.

This slider will default to a starting value of whatever color temperature was set at the time of capture. In other words, even if you alter the white balance during recording, only the initial value used when you hit record will matter as the starting value here.Unlike Direct Log Video, where the Color Temperature itself must generally be hard-baked into the footage and will potentially move around and fluctuate during the sequence, whichever color temperature is applied here will remain static through the clip and it will not vary.

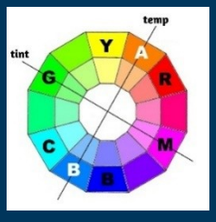
A bottom 'scale' bar appears to give you a reference point to the temperature level applied, and any changes to the bar will immediately be observed on the Offline Viewer section of the screen. The scale runs from Amber (Warm) to Blue (Cool).

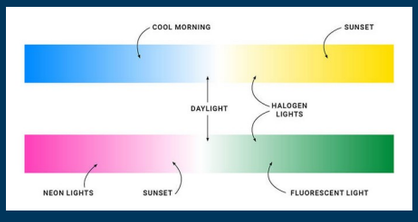


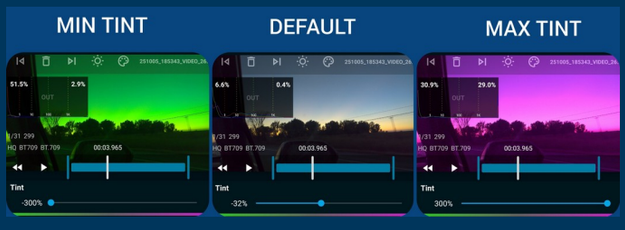
[4.75.5] Tint (Offline Render): Whereas Color Temperature controls the perceived 'Coolness/Blue' or 'Warmth/Amber' you'll perceive in the colors, Tint impacts the Green and Magenta tones.

Together with Color Temperature, Tint further helps maleate the specific tone range that remain unimpacted by Color Temperature alone. What does this mean? Below are illustrations to exemplify it. By leveraging both Axis together, you can combine the sliders at different thresholds to achieve the 'corners' otherwise unattainable by a single axis (color temperature or tint) alone.

Identically managed as with color temperature, the Tint is not baked into RAW and is strictly stored as metadata on the file, only sampled once at the beginning of the video. Any changes will also apply to the Offline Viewer section of the screen.

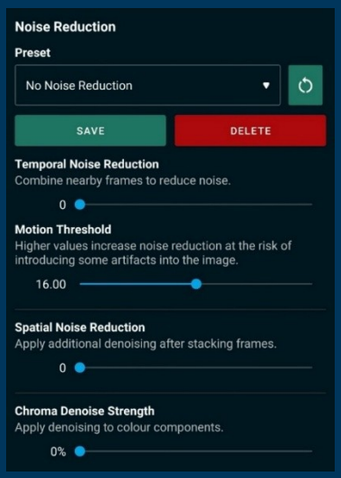






Color Temperature will cover the ***A*** (Amber/Warm) and ***B*** (Blue/Cool) tones!

Tint can only cover the ***G*** (Green) and ***M*** (Magenta) Axis!

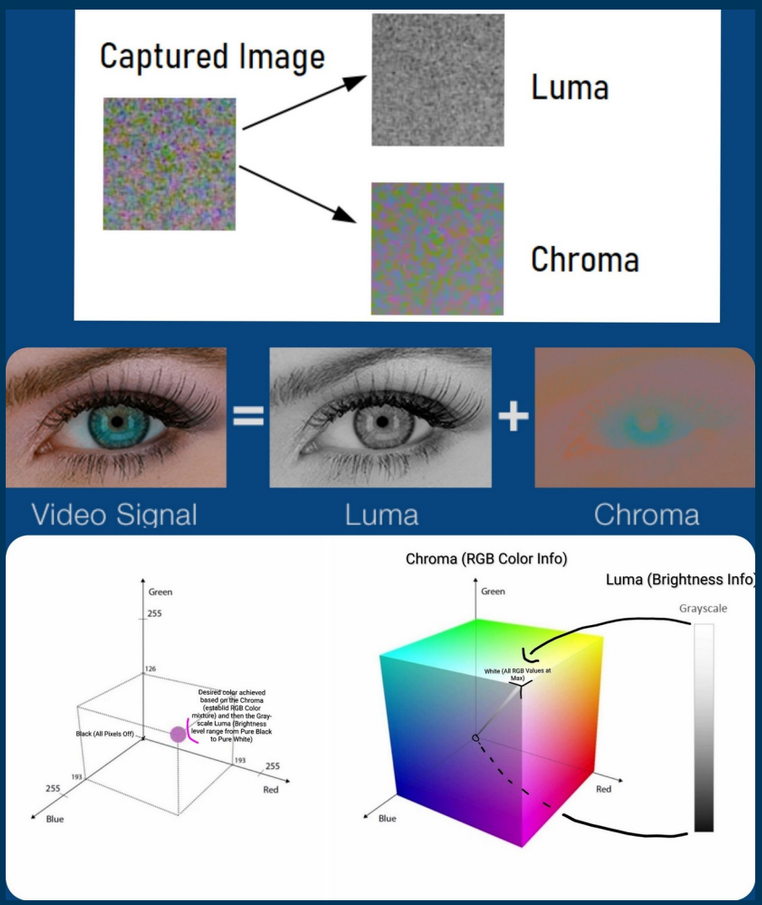


[4.75.6] Noise Reduction (Offline Render): Yet another strength in rendering offline, this category provides additional noise reduction and denoising tools that can be applied to your outputs during rendering.

The key strength is that we're applying this process directly from the RAW data—not from an already encoded video file that has already tossed away tons of information. This means the app's custom-developed algorithm gets the full, pristine Bayer data for its denoising capabilities. These highly effective tools can allow you to greatly reduce or at least mitigate that annoying noise in your videos when simply getting more light is not an option.

Just be warned: this does come at the cost of substantially longer rendering times, which is why some of these advanced settings (with the exception of Chroma Denoise) are not available in the Direct Log Video mode—at least, not yet... wink.

As a reminder, when we talk about noise, we're referring to those ugly and annoying grainy looking blobs and patches that appear on your image that are otherwise NOT what the scene shows—they are simply sensor signal interference.



Noise comes in two delicious flavors: Chrominance (Chroma) Noise and Luminance (Luma) Noise.

Both Luma and Chroma are components of the sensor image data that come together to create your final image. While Luma is simply how bright the value is (from pure black to pure white), Chroma is the mixture given by each of the Red, Green, and Blue color components.

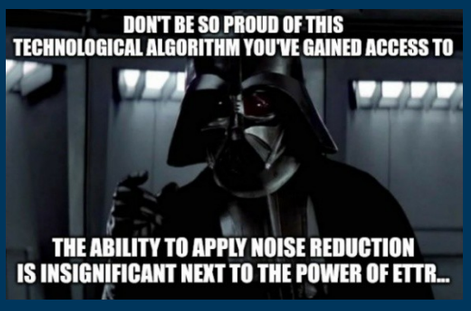
In a perfect world, you'd capture and preserve the image exactly as shot, but the world and your sensor are anything but perfect. Due to factors like inadequate lighting or underexposed settings, you'll begin observing both the Chroma and Luma components generate 'impurities' as fluctuations or visible jumps in the pixel data.

To better grasp this fluctuation, think of a specific color value on your image, say a green shade with a theoretical value of "50." Mild noise might see this value jump frame-to-frame in a small range, for example from 47 to 53. Heavy noise is when it jumps wildly, say from 35 to 65!

This massive variance is why the pixel color changes in a far more noticeable, colorful and ugly, eye-catching manner with every frame. The worse of these two noise sources will generally be Chroma, as the color flickering it produces is far more detracting to your visual quality. Make no mistake about it, noise will always be present, but the key part is to ensure it's more natural and 'grainy' rather than big blotches of flashing purple, red, and green flickering all over the darker textures of your image.

The objective here is to mitigate noise in a way that doesn't look unsightly rather than completely eliminating it. Too little noise can actually backfire, as you may potentially start seeing the image look like a static texture with moving objects more closely resembling the Mona Lisa that just came alive—we call that the nasty 'oil painting effect.'

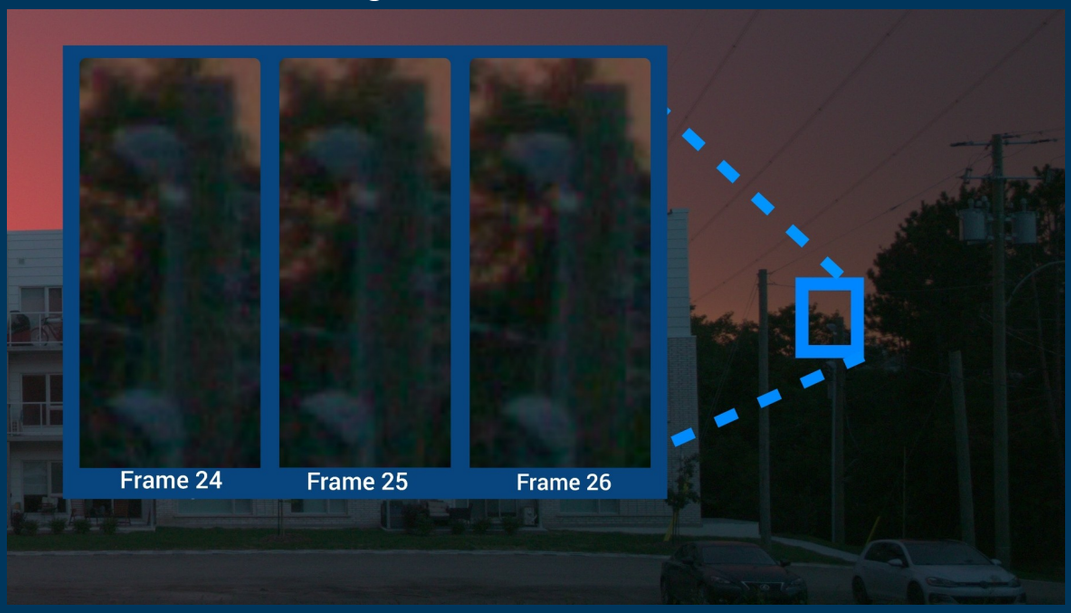
A critical aspect, however, is to ensure you get the lighting right first, as the best form of noise reduction is to ETTR (Expose To The Right). This means adding more light to your scene to saturate the sensor with clean data, thereby drowning out the noise. This is always superior to trying to suppress it after the fact, which will generally cost you micro detail and texture quality.



'But Ragu, what the f**k? I came here to learn to clean up my image, not learn sensor theory!' I hear you say. And yes! It seems strange to start with this – BUT, now that you've been surprise force-fed this knowledge and hopefully understand how the noise occurs, let's explore the options to fight it!!

So, we've established that noise is basically the Chroma (RGB) and Luma (Gray-scale from black to white) individual values per pixel getting jumpy and beginning to fluctuate off-target at random, in a range that worsens when the light that gets into the sensor deteriorates or gets reduced.

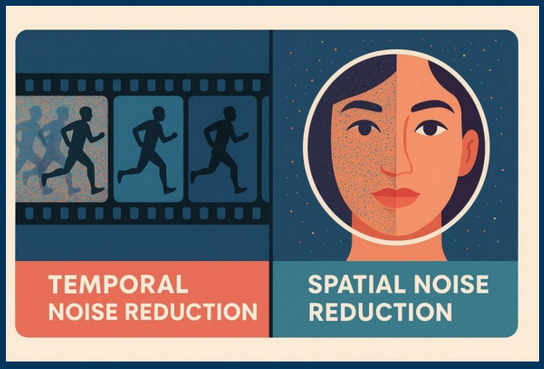
Eg. if you targeted an arbitrary Red Tone: 60, the frames may show a value every frame with an example slippage range every few frames from 55-65 rather than 60 at all times. And this slippage worsens when light decreases.

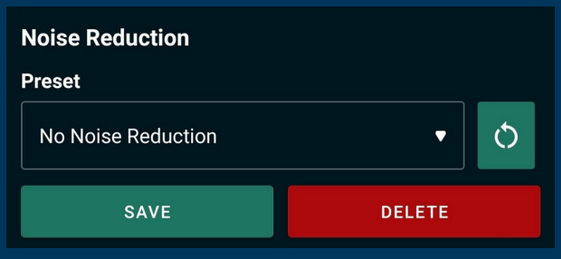


Look at the frames to the right taken from the same video, they're each one after another of the same static scene in a video!

If you zoom in and pixel peep, you will find the same colorful red noisy blobs are not in the same location on every frame because of this jumpiness we just described. This will be the key to mitigate noise!

Enter the advanced noise reduction methods; Temporal and Spatial Noise Reduction!



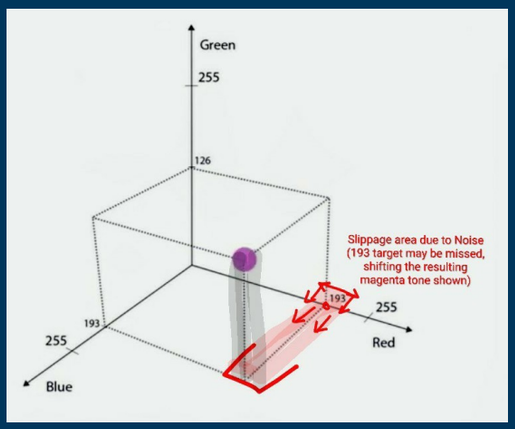


[4.75.7] Noise Reduction Preset (Offline Render): Given the nature of the available Noise Reduction options and the infinite combinations and variations you can play with, use this area to save any created presets (or access generic pre-made ones for you)

To create your own, simply adjust the following options about to be covered and then press SAVE to name them and keep them. You may also select them and press on DELETE to erase them.

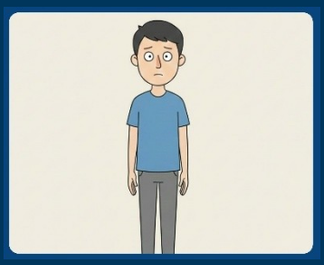
Lets now move on to the Denoising methods the app makes available to you!

[4.76] Temporal Noise Reduction



The fundamental principle of Temporal Denoising is simple: it uses multiple neighbor image frames in a video sequence to calculate and average out high-frequency noise in both the image's luminance (Luma) and color (Chroma) channels.

When a video algorithm examines a single frame, a noisy pixel value or reading that is momentarily "off", it has no reference point to confirm if a jump is a real scene change or just random noise, the "slippage" I mentioned, as seen on the right illustration and prior page shifting example. Temporal Denoising solves this by looking at an entire lineup of frames—the frames immediately before and after the current one.



This concept is easiest to understand with an analogy: imagine you are tasked with spotting a person whose height in a lineup is "irregular," representing noisy data. If you only see one person (the current frame) and they happen to be slightly taller than average, you can't be sure if they are a genuine anomaly or just naturally tall. Similarly, when the algorithm looks at a single frame's pixel value, it has no easy way to tell if a variation is a true color change or just noise.

Temporal Denoising provides context: if that original person (the current frame's value) is suddenly standing next to three shorter people (the surrounding frames whose values are all consistent), the original person instantly stands out as an irregular outlier. The algorithm can then confidently identify that outlier as noise, calculate the most probable, clean value by averaging the neighbors, and correct the noisy pixel value to match. This consistent frame-to-frame comparison is what grants Temporal Denoising its incredible power to clean up highly variant noise in both color and brightness.

It is important to understand that the effectiveness of this denoising relies on the fundamental concept of the Signal-to-Noise Ratio (SNR). Noise, such as read noise from the sensor, may remain relatively constant across all exposure levels. However, because the signal (actual light) is much stronger in the highlights, those brighter areas have a better SNR and are considered to be less “buried” in noise than data captured in the shadows.

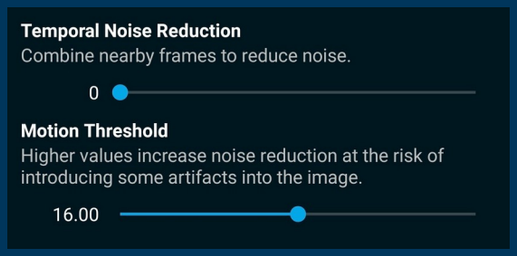
This is why Temporal Denoising, while effective everywhere, has the greatest impact in reducing the noise visible in the darker, lower-signal areas of the image. The mathematical process of noise reduction is applied to the raw, unprocessed brightness values received directly from the sensor, before the final image is created.

While incredibly effective, this method has two primary downsides. First, it suffers from movement artifacts: if objects in the scene move heavily, the same pixel location across different frames no longer represents the same element and this therefore reduces the effectiveness.

This lack of reliable comparative data causes the algorithm to misfire, resulting in visible noise trails or ghosting around moving areas. While you can mitigate this movement issue by algorithmically using neighboring pixels to estimate the upcoming correct values and tones (motion compensation, something MotionCam’s robust algorithm is surprisingly good at – remember the origin of the app?) movement practically always reduces the effectiveness of temporal stacking; it is only a question of how much.

Second, being too aggressive with temporal denoising can remove noise too well. The slight pixel "slippage" that is naturally present in video often contributes to a desired filmic feel. Removing it entirely makes the image look overly static and artificial, creating an unpleasant, overly smoothed appearance often called the "oil painting" or "frozen photo" effect; it is a great way to make your subjects look like the Mona Lisa just came alive, though..!

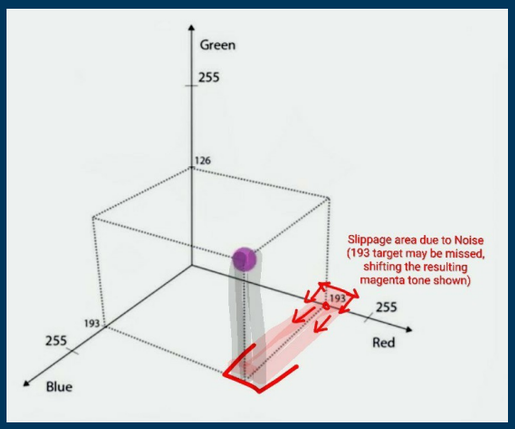
You can choose to do this Temporal Noise reduction in-app or with advanced tools like Neat Video, or editors like DaVinci Resolve, all with different implementations that may perform better or worse.



***[4.77] Temporal Noise Reduction (Offline Render)*:** Slide this setting to the right to increase the number of Frames used for the aforementioned Temporal Frame cross referencing the app will use to identify outlier noisy pixels! The more frames used, the more effectively the algorithm can become however at the cost of potential artifacts and slower rendering speeds.

Motion Threshold (Offline Render): This establishes the degree of motion compensation. A higher value provides less motion compensation and increases the frame merging intensity, but may introduce artifacting mentioned previously. A value of 0.00 disables Temporal Noise Reduction (use a value of ±0.13 to engage it)!

[4.78] Spatial Noise Reduction

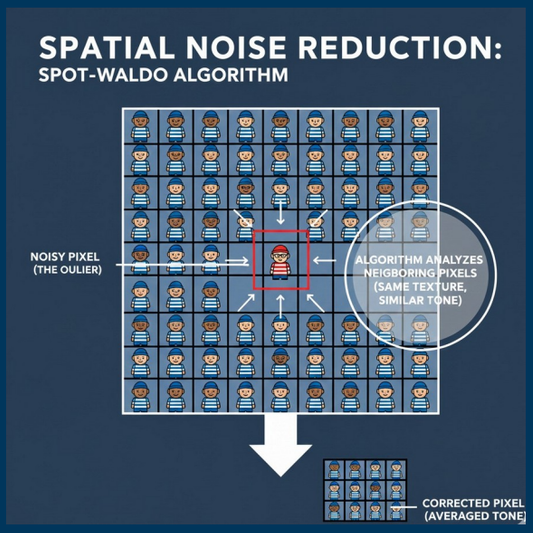


The fundamental principle of Spatial Denoising is simple: whereas Temporal Noise Reduction uses neighboring image frames to cross-reference and filter noise via averaging, Spatial Noise Reduction (NR) works independently within a single frame. This is accomplished by using the surrounding neighboring pixels to clean up a noisy outlier.

Although this method is generally considered to be mathematically inferior to temporal stacking, it is an effective tool to mitigate extreme noise when external frame data is insufficient or when heavy scene movement makes temporal analysis unreliable.

To understand this, let's assume you are shooting a plant with a consistently green texture in low light. While noise is highly likely to appear in the dark, the majority of the pixels on the plant should still be green, right?

Spatial NR allows the algorithm to mathematically spot variances by, for example, observing a pixel that is suddenly fluctuating towards red, despite all its neighbors being green. The algorithm then calculates and estimates a corrected value for the noisy pixel based on the surrounding, similar-toned pixels. This is the "Find Waldo, sensor pixel edition" approach (see right illustration).

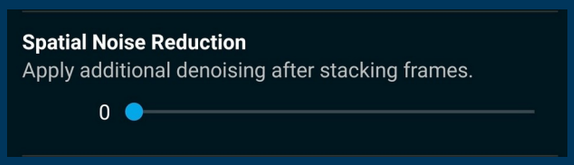


This method is useful for aggressively dealing with noise even in the absence of external frame data. The issue, however, is that this is merely a mathematical way to spot noise with the presumption that neighboring pixels aren't changing much. If the image contains genuine micro-details that vary in color, and the algorithm gets a false positive, this method essentially starts killing off fine textures and details.

In other words, while Temporal NR potentially preserves or even enhances detail by adding data from other frames, Spatial NR instead blurs and smooths the noise using only the data from the current frame.

You are effectively trading fine resolution for less noise. Since this process tends to be done to both the Luma and Chroma channels indiscriminately and may vary slightly every frame, it can be a less effective and more quality-costly method of noise reduction.

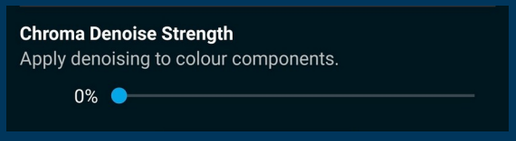
Ultimately, it is up to the user to judge if Spatial NR suits the scene, though it can render otherwise unusable noisy frames palatable.



***[4.79] Spatial Noise Reduction (Offline Render)*:** Slide this setting to the right to increase the aggressiveness of the Spatial Denoising.

A value of 0 is disabled, and values above 1 will gradually increase the Spatial NR strength. This option heavily impacts rendering times.

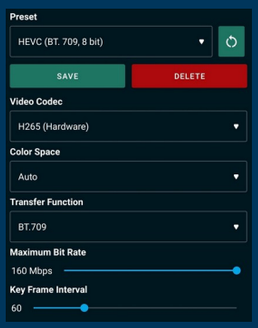
Of important note, the MotionCam Spatial Denoising algorithm requires you to currently use at least 1 stacked frame on the prior Temporal Noise Reduction option with a Motion Threshold of 0.13 in order to engage the Spatial Noise Reduction - this is done to mitigate it's shortfalls.



***[4.80] Chroma Denoise Strength (Offline Render)*:** Slide this setting to the right to increase the strength of the Chroma Denoising algorithm applied.

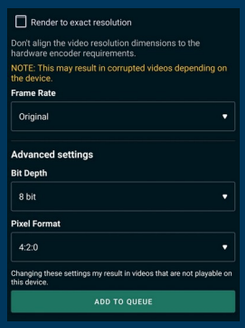
This option works identically to the previously discussed Direct Log Video Chroma Denoising options and you can refer to page 74 for Chroma Noise Reduction in order to understand this method.

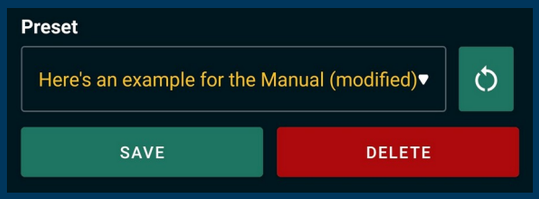
The idea is simple however, this method is a light Spatial Noise Reduction althorithm that runs quickly and attempts to mitigate light chroma noise while leaving Luma noise mostly unharmed. Although it's an excellent option that can avoid some of the problems of full-on Spatial Denoising, it may also be insufficient for heavy noise conditions.



[4.81] CODEC EXPORT OPTIONS (Offline Render): Use the following section and settings to define the encoding settings that will be applied for your video about to be encoded

Similar to the Direct Log Video mode encoding menu, you will also observe additional rendering settings made available particularly as far as codecs go, however.



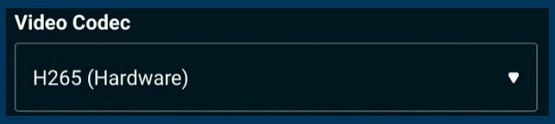


[4.81.1] Video Codec Preset (Offline Render): Use this setting to quickly apply saved presets that you've created with the following list of options (or that come built-in) for quick exporting.

Important minder that the list of presets will also show under the VIDEO MANAGER/MANAGE VIDEOS MENU (Page 93)

You may simply press the SAVE button once you've finalized your tune, which will prompt you to name the setup. Alternatively, to delete any saved presets, simply select any and hit DELETE.

Hitting the circle arrow green button simply undoes any changes that you may have further applied to an already created preset – if you've modified a preset the name will turn yellow and say (modified) to advise you of it.

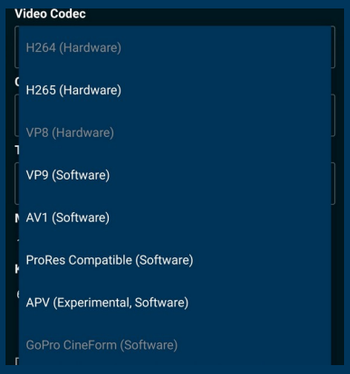


[4.81.2] Video Codec (Offline Render): Please see Page 59 for the prior comprehensive explanation about the app's Video Codecs and how they're handled.

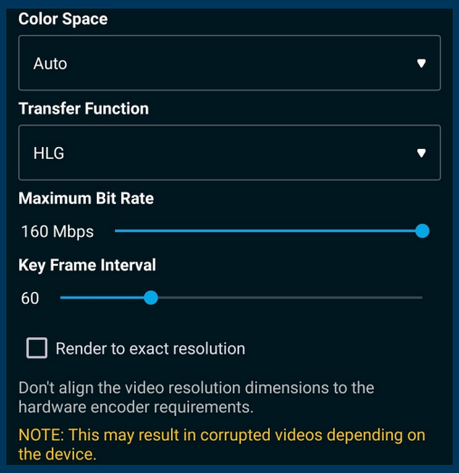
Unlike Direct Log Video however, you get to exercise granular control of the color depth, framerate and Chroma Subsampling of it, however (in the next options).

Additionally, extra software codecs show up while rendering offline, including APV (Advanced Professional Video) which are too intensive for real-time encoding via software acceleration.

Grayed out codecs simply mean that your current selection is incompatible with the selected color depth (eg. you selected 10-bit but the codec can only take 8-bit).



[4.81.3] CONTAINER DATA HANDLING: Please refer to the following pages for all prior options to see comprehensive explanations and illustrations respectively as they operate the exact same way as their Direct Log Video counterparts…



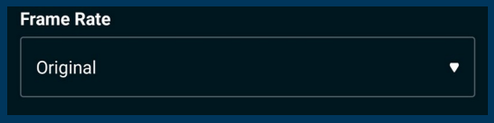
Color Space – Page

Transfer Function – Page

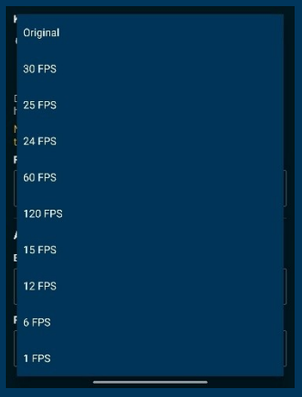
Maximum Bit Rate/Profile – Pages

Key Frame Interval – Page

Render to Exact Resolution – Page



[4.81.4] Frame Rate (Offline Render): Yet another benefit of rendering offline. Unlike Direct Log Video which requires you to set a framerate for the encoding to operate as designed, you can actually use this option to alter the framerate of the sequence altogether without consequences.



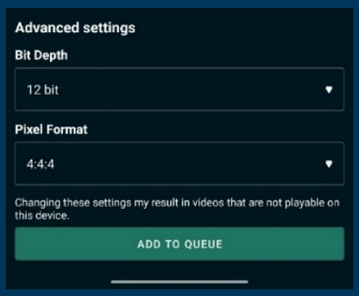
This means that you can effectively speed up slow down the video by simply altering the export framerate! As an example, exporting a video initially captured at 60fps with a Frame Rate option of 30fps slows it down by x0.5!

Do be mindful it's default is kept at 'Original' and any alterations may impact audio and cause slippage of it, necessitating adjustments in post. This is nevertheless a great tool even to export timelapses with video framerates if you so desire as well!

Pro Tip (Encoder Hack): Did you know, that if you slow down the video (eg. 60fps into 30fps) you can effectively push even more data out of the encoding this way?

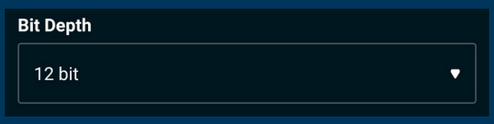
This works quite well for Key Frame based codecs like HEVC which generally operate on a Megabits PER SECOND approach.

It's actually quite simple; by slowing it down and extending the playback length, you're essentially letting the encoder stretch its legs and have more seconds to pack the data into! The downside is that you now need to fix the speed in post, and you may further cause audio slippage unlike the 'Original'. That said, if you're farming bitrate for some reason, this is a neat trick to retain higher density files (to a limit, of course)!



[4.81.5] Advanced Settings [Codec] (Offline Render): Unlike Direct Log Video which provides a list of a Codec + Color Depth + Chroma Subsampling that's bundled (and tops at 10-bit currently) for the encoding output options, the Offline Render also allows you to set them in a granular manner!

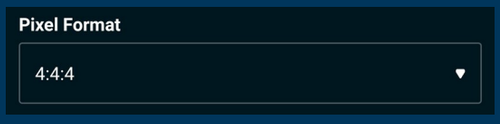
Although making changes to these two following options can further push the quality of the encoder to its limits, be aware that doing so may also render the videos unplayable unless you load them on an advanced editor!!! Be aware that some Codecs may force these to a specific value only and may gray out/disable these boxes.



[4.81.6] Bit Depth/Color Depth (Offline Render): Use this setting to define the desired color depth to be used for the encoding container.

Although you can achieve up to 12-bit Color Depths currently, this requires a Codec that can accommodate it (such as software accelerated AV1) and also this may be wasteful unless your sensor can actually capture 12-bit data to begin with! Additionally, if the codec is incompatible with a certain color depth, it will be grayed out.

Don't fall for the temptation to max this one out unless you have a good reason to —you can always check the RAW containers explanation if in doubt!

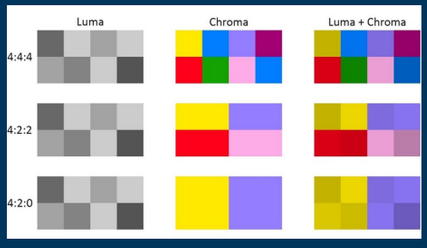


[4.81.7] Pixel Format/Chroma Subsampling (Offline Render): Use this option to adjust the Pixel Format, which defines the Chroma Subsampling ratio applied to a video's pixels during encoding.

The available formats are dependent on the chosen codec; unlike Direct Log, this process can achieve ratios up to 4:4:4. Any grayed-out options simply indicate the codec is incompatible with those formats.

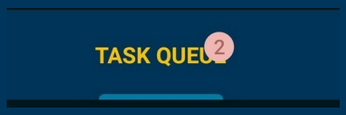
If you survived the Temporal/Spatial Denoising sections, you probably have a better understanding of how Chroma and Luma noise can get "jumpy." This same effect is quite costly if left unchecked when it comes to compression. To combat this and compress the video effectively, encoders take advantage of a brilliant loophole in human vision: your eyes are superb at discerning Luma noise (grayscale brightness jumps) but far less sensitive to Chroma noise (color changes).

By systematically "bunching" pixels together in a way your eye will barely notice, you can effectively take a bunch of colors and average them out into bigger, less dense clusters, thereby saving significant data. This deliberate reduction in color information is the mechanism of subsampling. For practical applications like green screen editing and CGI, retaining as much original color information as possible (i.e., less subsampling) is generally preferred.



When choosing a ratio, 4:2:0 is popular for viewing and final delivery, 4:2:2 is popular for intermediate editing and color correction, and 4:4:4 is known to be a luxury compression level often used for the highest quality post-production work. This leads to a crucial point: denoising a signal that is less accurate or already highly variant becomes far worse. Imagine you are trying to guess a number, but that number is already a guess made by someone else—your final result is essentially a guess off a guess. Similarly, when the encoder is forced to compress a highly variable, noisy pixel, the resulting compressed data is a weaker average built upon a bad original reading. Therein lies the value of a higher value like 4:2:2!

It is Important to remember that RAW data does not use this compression; the full Bayer data inherently retains the complete "jumpiness" or color variance of every single pixel! So, while someone might flex that they can record in 4:2:2 and you can only do 4:2:0, just remember what's even better than 4:4:4—**it's a detail that's a magnet for forum arguments and that makes the inner spec junkie happy…** Oh yeah, it's come full circle now hasn't it?? 😉 Only the G's who read my crash course will get this one.. I suppose this is a great to end the Render.



[4.82] TASK QUEUE / RENDER QUEUE

Use this tab on the MCRAW Video Manager Menu to review the progress of any current/past/future files you've tasked the app to render. The red floating bubble number indicates the number of current active queued tasks!

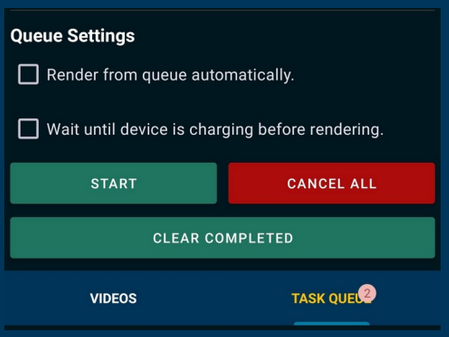
This comprises both CinemaDNG sequences and standard Codec outputs; you'll see both individual exports but also batches accordingly.

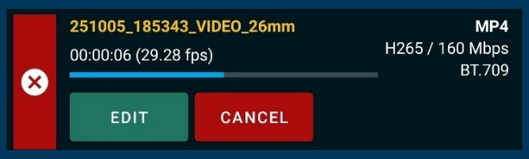
Do keep in mind only one output at a time can be rendered, the others get queued sequentially.

Depending on the device orientation, the top/left half (see left image) shows the individual task list.

The bottom/right side shows additional settings to define the session's preferences and batch handling settings (see right image).

This queue will be your files' final stop before experiencing their full quality!





[4.83] Task/Output Monitor: Regardless of whichever codec, format or otherwise, any output (single/batches too) you send for rendering, all tasks will individually be shown not unlike MCRAW reels in a box showing their general details as well as description and quick actions you may take.

Use this box area for every individual task to quickly identify them and understand their overall rendering health and what they contain accordingly.

Of special note, each box indeed does represent a task and can represent multiple variations of the same MCRAW rendered. This is to say, if you render the same MCRAW Video in 4 versions (eg. one with HEVC, one with ProRes, another as cDNG, then another trimmed HEVC with different settings) then you'll have 4 independent tasks accordingly, even if they're all originating from the same source reel!

The left side of the monitor shows a rounded X that you can press to dismiss the Task; this aborts the exporting if already underway plus removed the item as well! You can also press CANCEL to stop an individual task this way from rendering.

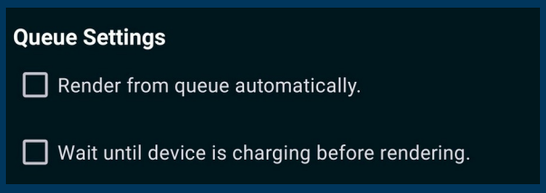
You can also observe the name at the top, and the estimated Rendering time plus a rendering bar showing the expected progress.

Lastly, you also can see the Codec/Format and container in use, plus the Bitrate/Profile and Transfer Function in use for your export as needed!

Once completed, a thumbnail will appear for codec videos as well as the status will be displayed (eg. Succeeded).

[4.83.1] OPEN / EDIT / RUN AGAIN (Task Queue): Use these options to individually either open the exported output just rendered (if successful) using an external video viewer, or to re-edit the same Task using the keeping the existing edits you've already used on it (reloads the edited settings!)

Use the Run Again button to restart the same task in the case it failed or something interrupted it/the app preventing it from completing



[4.84] QUEUE SETTINGS: Use the following options to set the parameter for the queue's tasks to be rendered

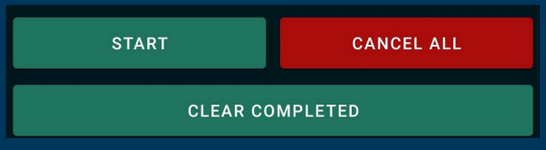
[4.84.1] Render from Queue Automatically (Task Queue): Check this box to allow any tasked outputs to immediately commence rendering on a first come-first serve basis as soon as they're queued.

If left unchecked, you will have to press the START button at the bottom to initiate rendering. Files pending the start button will show ENQUEUED statuses.

[4.84.2] Wait Until Device Is Charging Before Rendering (Task Queue): Check this box to allow rendering only when the device is plugged in and actively charging.

This is useful due to both the heating produced when doing the otherwise super intense rendering task, as well as to avoid running out of power during extended rendering sessions.

This option can be used with the prior box as well and will ignore the START button. Any pending outputs no longer restricted by the prior box (or if you pressed the START button) will be shown under the "Waiting to run" status.



[4.84.3] START (Task Queue): Press this to begin rendering tasks or set files in an ENQUEUED/WAITING TO RUN state.

[4.84.4] CANCEL ALL (Task Queue): Press to immediately abort and cancel all current tasks either being rendered or pending rendering. Successfully aborted tasks will be shown under a CANCELLED status.

[4.84.5] CLEAR COMPLETED (Task Queue): Press this to clear any completed/failed/cancelled tasks on the queue.

//[4.85] CLOSING THOUGHTS ON UI EVOLUTION + UP NEXT//

So, you've made it past that. Whether you scaled this learning curve the easy way or the hard way, you've hopefully absorbed some aspects of using MotionCam Pro. You've come a long way!

It is absolutely crucial to remember that this application is not static; it evolves, often aggressively, so everything we've covered is subject to change. For instance, you may have noticed a deliberate omission of features like the Burst and Photo modes. This is because the plan is to harmonize and unify them into the streamlined Direct Log mode for much better convenience and flow (though rest assured, the foundational RAW video mode settings, like RAW stream selection and framerate, will carry over).

A bunch of new, crazy things are always coming down the pipeline, so keeping up is an ongoing effort. But take comfort: the core essentials will always remain the same. It's like getting into a new, high-performance car; it feels weird at first, but everything makes perfect sense once you finally locate that damned seat recliner handle!

Alright, with that brief detour out of the way, let our Mountain Climb resume. We are ascending into the final, critical stretch.

Did you know that on Mt. Everest, above a certain altitude, climbers enter what they call the "Death Zone"? Essentially, this is the highest area where air is so thin that humans can't survive more than 24 hours without oxygen tanks.

Likewise, we're now entering the technical equivalent: the FAQs and Troubleshooting zone. You don't want to linger here. It is essential to get you going out of any issue that may arise, else you risk irreversible frustration, anger, or worst of all, the dreaded one-star review where you rant "it doesn't work!" without ever explaining what happened. (Yes, that incoherence is the same symptom as Death Zone-induced hypoxia.)

Let's get the RAW oxygen flowing and solve your problems quickly, shall we?

- [5] FAQs & Troubleshooting –

"My phone should work, but MotionCam Pro says there's no RAW support."

Ouch. You hit a wall. Please note that based on user feedback, the device you've mentioned is often known to be incapable of shooting RAW images or has a locked RAW stream, which prevents MotionCam from accessing the pure sensor data it needs. This is an Original Equipment Manufacturer (OEM) decision, not an app bug.

The fix is a political one: ensure you voice your concern by contacting your device OEM and submitting feedback through their official channels and forums asking to enable this feature and unlock proper Camera2API raw stream capabilities.

"Are Exynos devices from Samsung good for MotionCam?"

If you're considering an Exynos device, you need to read this. Exynos devices from the S21 series and prior are infamous for a firmware bug that stops them from capturing above 15FPS reliably. For that reason, they are strongly discouraged for MotionCam usage.

Samsung has since addressed this from the S22 series and onwards, but you can still expect 30FPS limits and overall lesser/more unstable performance. The takeaway? Snapdragon devices are always recommended over Exynos.

"What about MediaTek? Are those processors good or at least better than Exynos?"

The short answer is yes, MediaTek is generally better than Exynos when it comes to supporting third-party camera apps. However, they come with their own set of limitations, so your experience will vary significantly depending on the device tier.

On flagship MediaTek models, we usually find a much better and more reliable camera pipeline than what you'd find on comparable Exynos devices. The common issue here is that they often impose rigid 30 FPS limiters on the RAW stream. While this is far better than having a completely crippled system, it's unfortunately not near the unrestricted performance that Snapdragon flagship models consistently offer.

The situation worsens on budget and mid-range MediaTek chipsets. Here, your mileage may vary greatly. You might find yourself capped at an abysmal 15 FPS (similar to the worst Exynos performance), or, even worse, the manufacturer may not expose the RAW stream access at all.

In summary, MediaTek is a safer bet than an Exynos flagship, but you must be wary of their budget and mid-range offerings. Always check community reports before relying on them for RAW video performance.

“Then what about Tensor? Isn’t it just an Exynos in disguise? Should I avoid them?”

You are absolutely right to question this! Everyone in the tech community knows that Google's Tensor chipsets are rooted in Exynos designs. However, the key difference—and why MotionCam Pro works well on Pixels—lies in one simple fact: the firmware and camera drivers are not the abominations that Samsung provides for Exynos.

For all of the Google Pixel's flaws (and there are plenty, including often crude performance compared to top-tier Snapdragons), the third-party and camera-specific drivers are generally excellent as far as low-level camera operation is concerned.

Think of it like this: the processor is the sports car, and the firmware/drivers are the tires.

You could, in theory, have an insane sports car perform horribly with terrible, worn-down, low-cost tires—you simply cannot extract its performance in full, even if the car itself is great. Conversely, great tires can make an average car feel much better! Google focuses on ensuring those 'tires'—the camera drivers—are top-tier, guaranteeing reliable access to the RAW stream that MotionCam Pro relies on. Therefore, you do not need to avoid Pixel phones simply because of their Tensor chip.

“Where are my 8K or 50MP and 200MP modes?”

You’ve got a great sensor, but yet the option is missing. Although many devices offer high-resolution Quad Bayer (48/50MP) sensors or even Nona Bayer (200MP), access to these modes is often blocked for third-party apps and reserved strictly for the stock camera.

Furthermore, since MotionCam uses the RAW stream instead of the standard video or photo stream, to record 8K you must have 50MP mode available for RAW shooting which the app can then reduce to 8K if you choose to crop that.

While a few devices are the exception, they tend not to be the norm. You can of course root some devices to pry these modes open, but even then there’s no guarantee this may be available for your specific model, so we encourage strong research within the community to confirm this!

“I can't find a 10-bit option for HEVC!”

This one stings. If you're using processors like the Snapdragon 888, 8 Gen 1, or their respective Exynos counterparts, you won't have access to the 10-bit HEVC option in MotionCam.

Although these chips do in fact offer 10-bit encoders for the video stream, MotionCam uses an alternative pipeline to bypass the Image Signal Processor (ISP), and that encoder pipeline is sadly disabled on those particular SoCs. Samsung dropped the ball on those for some reason and although they’re in fact Snapdragon chips, these were fabricated by Samsung for those model years.

Still need 10-bit? Try rendering into ProRes instead, that'll sort you out! As ProRes is software accelerated, this will not matter then; keeping in mind the performance toll!

Not satisfied yet and want an actual fix? Well, there’s some hope perhaps! This issue has sometimes be fixed by using Custom ROMs instead of the Stock ROM that comes with the devices.

Basically, if you have the means to unlock your bootloader and are ok with a Custom ROM, you likely can address this one!

“Why am I getting dropped frames in Direct Log mode?”

The Direct Log name itself reveals a big detail often missed by new users. If you are using Direct Log mode and experiencing dropped frames, be mindful that MotionCam taking a totally different approach than all other video apps to create classic video files! It is performing an on-the-fly RAW video capture and immediate encoding onto a Codec file.

This is actually more intensive than shooting standard RAW footage because the phone is doing two super-heavy tasks simultaneously.

It’s immensely more intense than what standard video apps do - which in their case is simply to obtain the footage pre-processed by the Image Signal Processing chip!

If the issue persists, try shooting in plain RAW and then export offline after your shoot is complete.

“I’ve done everything but keep getting dropped frames at 60fps in Direct Log mode!”

Only the absolute best of the best, most optimized and powerful flagships can currently capture in Direct Log at 60fps HEVC. As per the prior question answered above, this is NOT classic video and you’re assaulting your device with this workload, 60fps being even crazier to capture this way.

We strongly encourage checking on the Discord for the latest devices that can pull this off, such as the Xiaomi 15 Ultra, Nubia Z60U and above models, as well as the Vivo X200U to name a few.

“My footage is noisy. Did I do something wrong?”

Welcome to true RAW video! The reason your footage is noisy is that MotionCam captures directly from the RAW stream, meaning the image is completely unfiltered and not denoised whatsoever, unlike other apps.

While a dark scene may look very noisy at first, this noise is actually just pure sensor data, which gives you the best potential for detail.

The fix is three-fold: Expose To the Right (ETTR) to capture more light and reduce native noise, add more light if you can, or simply use denoising (in-app or in post) and crush the shadows to hide the grain during grading.

“How do I find the ideal ISO (Analog Gain)?”

Forget what you know about cameras. Phone ISO is like a volume knob, with the lowest setting being the 'base' or best ISO to shoot in. You can increase it within a certain range—your 'analog gain,' or how much you are boosting the ‘sensitivity’ via physically increasing the sensor current, which amplifies the signal. Unlike many camera which have ‘Base’ ISO can sometimes become ‘invariant’ or otherwise show no changes to the actual again applied when you go past it

Increasing Analog ISO essentially provides better SNR (Signal To Noise Ratio) which helps shadow details, but you sacrifice maximum dynamic range. The higher the gain, the less dynamic range you can capture vs base ISO.

These ranges can be easily found in your phone's Camera2API reports, and the golden rule is to never exceed the analog gain limit when shooting RAW (you may digitally boost as needed for lossy codecs however as they can benefit from this by the fact of encoding Histogram). To find each lens’ maximum analog ISO, use this app

https://play.google.com/store/apps/details?id=com.camera2.test

Identify the lens you are searching for (same numbers as in MotionCam) and then look for their respective android.sensor.maxAnalogSensitivity value.

The number defined there indicates the lens’ maximum ISO it can boost towards by physically increasing the signal gain. Anything above is basically fake ISO and applying digital boosting, meaning it’s useless for RAW, however it may help with lossy encoding to preserve shadow details instead of discarding them. This will further amplify existing noise however.

“Which mode should I use to record? RAW video or Direct Log? Which is better?”

The choice is simple, and it's a battle of quality versus size. RAW will always give you the absolute highest quality in all situations, but it comes at the cost of storage space.

Direct Log is there to reduce the storage issue with some compression. This trade-off is generally fine in strong lighting, but the encoding process begins to struggle when you introduce darkness.

If you intend to shoot in the dark, stick with RAW or ProRes to prevent detail loss as it’s the best sort of quality insurance you can buy.

Furthermore, Direct Log is actually the more taxing of the two modes simply by nature of having to basically shoot RAW video, but instead of saving the video, it buffers and holds it in the RAM then proceeds to try encoding it real-time. This is unlike any other video app’s approach and orders of magnitude harder – you’re basically trying to harness the RAW purity without its size!

“Wait, my RAW or Direct Log videos are black (RAW10 Black Video).”

You're likely on a Xiaomi 14 Ultra or S25U. These devices are known to have issues using the RAW10 stream. Thankfully, the fix is easy: simply switch to the RAW_SENSOR mode to resolve the issue and try again.

“I have issues with Stabilization/My OIS is not working!”

You're right, it feels weak. MotionCam can only use Optical Image Stabilization (OIS), but it cannot access the phone's Electronic/Digital Image Stabilization (EIS) because it bypasses the ISP.

Many don't realize that EIS accounts for about 80% of the stabilization work done by phones. While OIS helps, it’s often severely overestimated. The true pro fix is to digitally stabilize in post: record the Gyrodata in MotionCam and use an external application like Gyroflow for the best possible results.

You can even use the Google Photos app (yes, you read right) to stabilize videos – a surprisingly robust option!

“I can’t zoom in or switch cameras while recording.”

Unfortunately, because MotionCam records via the raw stream, things like digital zoom and seamless camera switching are currently not possible (although the developer has expressed a way could be achieved down the line… in theory).

These functions are generally handled by the ISP, which the app seeks to bypass for maximum control. While you can achieve the zoom effect in post-production, camera switching while rolling is something we're hoping to see enabled in a future update.

“How do I choose the correct RAW Container/is RAW10 or RAW_SENSOR better?”

Think of it like soda bottles: RAW10 is a 10-bit container (a 0.5L bottle), and RAW_SENSOR is a 16-bit container (a 2L bottle). If your sensor produces 10-bit data, you can use the 16-bit container, but you are carrying around an extra large bottle you don't need. If your phone is 12-bit, however, and you only get RAW_SENSOR, then it's appropriate.

The goal is to size accordingly. You can find your sensor color depth on a Camera2API report. The below app is the community favorite one and in my opinion the most powerful of them all…

https://play.google.com/store/apps/details?id=com.camera2.test

Simply find the lens you are searching for (all lenses are different so may vary) then search for the android.sensor.info.whiteLevel category.

8-bit would be 255, 10-bit is 1023, 12-bit is 4095 then 14-bit is 16383 and so on.

Keep in mind these values are not including the complete off/black tone, so although for example 10-bit is 1024 shades, assuming you eliminate the OFF/Total Black state, it’s actually now 1023 shades or ‘levels’ of white/ON!

“How do can I Focus Rack?”

This is one of the coolest features! To use it, simply:

* Press on the Focus value on the UI to pull up the focus slider.

* The focus rack feature will appear on the bottom left.

* Set your initial focus position and long press on the rack’s left value to define that as the starting point!

* Set your final focus position and long press on the rack’s right value to define it as the end point as well.

* Press the play button to start the automated rack focus.

Pro Tip, you can also use the above method, but press on the circle of the slider to manually move the focus level in a much quicker and shorter way at an aggressive speed.

“How can I record in or shoot video in my own Capture Crop (eg. 9:16 Open Gate)”

Don't stick to the presets! Can’t find a specific shooting ratio crop or resolution? You can actually change the capture crop or use a custom ratio, press the '####x#### Video' option on the bottom left UI bar and either select one of the many presets or hit the ADD button below the selector to create your own custom values.

The list shown is merely pre-made ones, but you can also make your own!

“The image/field of view is cropped in when I use lower resolutions than 4K!”

If you find your Field of View is too zoomed in when you select a lower resolution, use Binning mode. This allows you to achieve a lower resolution (e.g., 4K into 1080p) while magically keeping the same original Field of View (FoV), ensuring you don't lose the wider look just to save on file size.

Stock apps tend to do this automatically however MotionCam goes the literal route and if you therefore invoke an open gate image, 4K is simply cropping 3840x2160 off the 4000x3000 area. 1080p/FHD would likewise do a 1920x1080 crop from 4000x3000.

Binning enables you to reduce the resolution while maintaining the same field of view (eg. Bin 4k into 1080p without cropping). As a nice bonus, binning can significantly enhance performance due to having to render at lower resolutions!

“Why do I see some camera lenses repeated/double with weird numbers and dashes?”

Welcome to the Android jungle of camera lens IDs! This is what we refer to as a “logical multicamera ID” configuration and it’s a bit strange at first, but there’s a reason! Jump to page 51 for a complete explanation on this!

“How can I exporting and edit/use MCRAW Footage?”

MCRAW is a MotionCam-exclusive, lossless compressed RAW video format. Only a few editors can handle it directly at this time (we’ve been pushing for its wider adoption, so speak up and request it to the developers and communities of your favorite editors if you want to see it more commonly).

Currently, you can directly load MCRAW files onto MLV App, VKDT, and even on Adobe Premiere using the app’s PC Plug-in!

The most reliable workflow is to use the new MotionCam Fuse project which at this time can essentially implant virtual cDNGs into DaVinci Resolve, saving you the conversion headaches.

Alternatively, you can select the 'Export DNG' button in the Manage Videos menu. This will create a zip file full of DNGs, otherwise known as CinemaDNG, which you can drag and drop directly into your editor of choice, like DaVinci Resolve.

You can also export from MCRAW into other codecs like ProRes and HEVC in-app via the Edit Videos button (or in bulk via the 'Queue All' button).

“My colors are shifting and the gamma is wrong in DaVinci Resolve!”

This is a classic RAW/Log workflow issue, not a bug! MotionCam shoots in wide color spaces and Log formats to give you maximum grading latitude. To fix the color shift:

* Navigate to the Color Page in Resolve and add a Color Space Transform (CST) node.
* Set the Input Color Space and Input Gamma to match what you recorded in the app (e.g. DaVinci Wide Gamut and DaVinci Intermediate).
* Set the Output Color Space and Output Gamma to your delivery standard (e.g., Rec. 709 / Gamma 2.4). This translates the file correctly for grading.

It’s worth mentioning that RAW data is linear and DaVinci Resolve may apply a Rec. 709 output function to it; this is fine as all the data is still being held, but merely visualized as Rec. 709, however all the editing latitude remains intact!

“My audio is drifting out of sync in my editor.”

If your phone struggled with the high data rate of provides an odd numbered framerate output, the output file might be a Variable Frame Rate (VFR) file. Professional editors hate VFR.

They will misinterpret the footage and cause the audio and video to drift apart over time. The complex fix is to transcode the file to a Constant Frame Rate (CFR) using a tool like HandBrake or Shutter Encoder before you import it into your editing timeline. You can also use MotionCam Fuse to achieve this!

Alternatively, separate the Video from it’s audio component, then manually adjust the audio component’s playback framerate which may be rounded down/up by the editor, causing a mismatch and subsequent slippage!

“I want to Stabilize with Gyroflow; where do I get a lens profile?”

Gyroflow is the king of stabilization, but it needs a lens profile to work its magic. It is generally highly recommended that you create your own lens profile specific to your device and MotionCam setup for the most accurate results.

Intimidated? Don’t be! It’s actually fairly easy! Check out this comprehensive tutorial which includes details on how to use Gyroflow, create lens profiles, and much more!

https://youtu.be/QAds3x8UU1w?si=ME95Kj6qik3ae4Pt

“Got any free LUTs?”

Of course! Check out Fresh LUTs. You’ll find some great resources there to get started on your grading journey.

https://freshluts.com/

You can also use import pre-made LUTs for the app from the MotionCam website itself, and directly retrieve them in-app even!

“My external microphone is not recognized in the app.”

This is an annoying issue. Especially with high-end external wireless mics you may notice the app doesn't pick up the external signal, defaulting to the internal mic and not providing additional options under the audio devices list.

Unfortunately external wireless mics are not currently supported by MotionCam and they must be physically plugged into the USB port of the phone (or headphone jack even!).

“Why Can't I Record Higher Than 30 FPS? Other Apps Can Do It”

This limitation comes down to the core difference between MotionCam Pro and virtually every other non-stock camera app: MotionCam exclusively uses the RAW stream for all of its functions, including video and photography modes, as well as Direct Log Video.

What this means is that MotionCam is never invoking the standard video encoding functions of your phone. Instead, it is asking the Camera2 API for a high number of full-capacity RAW images (frames) per second.

Contrast this with a classic video app. A standard app asks the Camera2 API for, say, 60 FPS of encoded video (like an HEVC file). This stream is quickly processed and handed off by your phone's dedicated Image Signal Processor (ISP).

MotionCam, however, asks exclusively for the unprocessed RAW data. If your phone's manufacturer (the OEM) did not expose the capability for the Camera2 API to deliver 60 RAW images per second, then you are unfortunately locked to the maximum rate that the RAW stream can deliver—even if your phone can shoot 60 FPS or even 120 FPS video via the ISP. This is why if your phone can't shoot RAW images at all, you can't use MotionCam. The same rule applies to frame rate.

Why do my Offline renders (eg. LUTs) look different than what the app preview shows?

As of this release, the current Offline Render preview is not entirely accurate, so only use it to get an idea of the settings’ impact.

This is being imminently dealt with as of the upcoming version 4.5 but remains an issue for now. Use it as a reference point but do not attempt professional or final level calibration and grading this way.

Why does my final capture look different than the viewfinder image?

The primary reason is that the camera viewfinder you see when you open the app is essentially the image provided by the Android Camera2 API. We call this the ‘Android Preview’ mode. This image represents the video that your device’s own Video Image Signal Processor (ISP) would produce—it’s the same processed look you see in your stock camera app.

However, MotionCam Pro fundamentally works by using the unprocessed RAW stream to create all video, completely bypassing that stock ISP processing. This is exactly where the app's quality advantage becomes apparent versus ordinary solutions, but it also means there will be a visible difference between "what you see" in the standard viewfinder and "what you get" in the final capture.

The good news is that you can get a live look at the true RAW data! You can use the Direct Preview mode (look for the Eye icon on the top left bar) to actually see the unprocessed stream in real-time. Keep in mind that this function is extremely intense to render, so it runs at a lower framerate than the standard preview to preserve power.

The even better news is that the developer has been working some serious optimization magic to get the Direct Preview running smoothly and in real-time—even while you're shooting! Expect significant improvements to the preview experience in version 4.5.

“I can’t find, use or see a specific lens on my device!”

First, before you jump to conclusions, ensure you’ve tried the quick fix: use the ‘Search for Hidden Lenses’ option within the app. Here’s why that sometimes works: The Android system uses an interface called the Camera2 API to report exactly which lenses are available. Sometimes, the API reports an incomplete list. For example, the system might report, “You have lenses 1, 2, and 3,” but if the app pushes back and pretends there’s a lens 4, it can say, “Great! Now open lens 4!” and poll for more – the system often grants access to that undeclared lens anyway!

That 'Search for Hidden Lenses' option is essentially our way of asking the system one last time pretending it did declare it, just hoping for the best. It’s kind of like telling your friend out of the blue “I know what you did..!” then they proceed to panic and spill the beans about their extramarital affair…

If you still can’t find a lens you know you have (like a telephoto or ultrawide), that’s where things get complicated and extremely frustrating. Unfortunately, some manufacturers—particularly on older devices such as the OnePlus 12 and older, Samsung S21 Series and older, and certain older Xiaomi phones—did not expose all available lenses to third-party apps via the Camera2 API. Why the lockout? Your guess is as good as ours, and yes, it absolutely stinks.

If you're stuck, the only remaining options are rather extreme. In some cases (like older OnePlus devices), installing a Custom ROM can circumvent this restriction and successfully unlock the missing lenses. However, in other cases, particularly on some older Samsung Galaxy series phones, the lenses are permanently locked out of the API. Sadly, no amount of rooting or custom software can change this.

If you are unhappy with this restriction, your best recourse is to complain directly to your phone manufacturer. Spread awareness and demand that they never attempt such restrictive nonsense again. Access to your device's full hardware should always be available!

“The focal ranges shown are not the same as what’s reported on official sources, why?”

Do note that the app receives focal range information from the Camera2API itself. Although this won’t cause any cropping whatsoever nor impact the camera module, it can throw off users into thinking they’re using the wrong lens, or that there’s an issue of sorts.

The reported focal range can be wrong versus something like GSMarena’s specs reporting, but it’s not to be taken too seriously. It can sometimes provide interesting insights though! As an example, on the OnePlus 8 Pro, GSMarena reports it has a X3 (72mm) telephoto at 8MP. If you actually pull this hidden lens with MotionCam, you obtain a 13MP sensor at 59mm! What does this mean? They cropped it in stock and simply pretend it’s got higher zoom!

Conversely, some OEMs may round off on the focal range sometimes to hit a specific ‘classic’ number rather than an odd one. Will it matter? Probably not, but Android jungle, eh..? Yes, I am a Canadian. How do you like them maples..?

“The videos I recorded/rendered outputs in the app are corrupted/broken!”

So this one can get tricky. The app’s 10-bit HEVC exporting in particular uses an alternative encoding pipeline that’s otherwise different than usual. The issue is that this encoder mode is EXTREMELY temperamental and pushing it ‘out of spec’ can produce issues.

As an example, some devices will break the 10-bit encoder if you use any resolution above 3840x2160 (4K). Want an example? Let’s say you shoot Open Gate RAW video (4000x3000, give or take) and you attempt to encode it. As you’ve technically exceeded 3840x2160, it’s gonna potentially create a neo-modern Picasso line abomination.

Likewise, want to shoot 4000x2160 or something of the like? Yep – out of spec, so will also break! If this should occur to your device (doesn’t happen on all) then you may use an alternative codec like ProRes, VP9 or even AVC if you should have it in 10-bit mode available.

Of particular note; in some cases like with the S20 FE or such Samsungs of the like, they completely broke the encoder pipeline and Direct Log mode will immediately produce broken and flickering outputs – this issue is unrelated and a separate matter of its own! The next FAQ point will talk about this.

“My Direct Log Video Mode Captures Are Broken! It Just Won't Work!”

While this is a relatively rare issue, it unfortunately tends to happen most commonly on the Samsung Galaxy and Note 20 series devices. On these specific models, the Direct Log Video mode often fails to encode properly, resulting in unusable video.

This issue won't manifest as frame drops; instead, the problems appear directly in the output file. You will observe severe flickering, flashing, and persistent green lines across the entire frame, even though the image initially appears as expected.

Why does this happen? Simply put, something appears to be fundamentally broken at the Samsung firmware level for this generation of devices. (Believe us, one of our developers owns an S20 FE and exhausted every possible attempt to make it work, all without success.)

Unfortunately, there is no known fix for this encoding anomaly. The only reliable solution is to avoid Direct Log entirely on these specific devices. You must shoot in RAW video and then promptly use the Offline Render feature. Crucially, the offline rendering process is not impacted by this firmware bug, so by using them instead you can ensure you get a clean, usable final file.

“Why Do I See Green Lines or Mirrored Edges on My Videos?”

This common issue happens at the very borders of the frame. It's caused by a quirk in how the Android operating system handles video encoding.

While MotionCam Pro aims to render your video at the exact resolution you choose, the Android encoder on many devices actually requires the resolution to be padded to a specific multiple (often a multiple of 64) for it to function correctly. If this required padding isn't present, the encoder can often break or introduce an undesirable green line along the outermost borders of the frame.

To prevent this sudden, jarring green line, the app dynamically adds this required padding region itself. Instead of leaving it as a blank or broken area, it fills the extra space with a "mirror" effect of the frame's edge. This effectively conceals the critical encoding issue.

For example, if you choose a resolution like 3840x2160 (4K), the app might be forced to make it 3840x2176 to satisfy the encoder's requirements. Those extra 16 pixels will display the mirrored edge. If this mirrored border is too noticeable, you can simply crop it off in post-production with minimal loss. If you wish to manually override this (and risk potential green lines), you can use the manual resolution settings discussed earlier in the guide (page 83).

“Does MotionCam avoid all processing on my phone, including AI”?

Absolutely! This is the core reason you are using MotionCam Pro in the first place. By default, the app is designed to deliver footage that is as pure and untouched as possible, as seen by the sensor.

However, just because we bypass the stock processing doesn't mean you can't customize your look. All processing within MotionCam is applied only at your explicit instruction and to your specific taste. If you happen to fancy some sharpening, oversaturation, or heavy-handed noise reduction (yes, even masochistic levels!), you have complete control to apply them exactly how you like.

The key takeaway is this: you control the process. You can apply modifications to your specific level, or you can turn everything off and leave the image as pure as possible. Choice is the menu!

Are photos processed with this app?

While MotionCam Pro allows you to obtain completely pure RAW stills and video, photos—particularly compressed files like JPEGs—will generally benefit from some level of processing. This is necessary to correctly transform the pure, linear RAW data into a standardized color space like sRGB or 8-bit SDR. If you skip this step, compression can eat away at the useful information, often leaving you with an underwhelming final image.

For this reason, some levels of processing are ideal, but you have complete control. You can use the Photo Mode's 'Photo Settings' box to fine-tune the output, adjusting factors like saturation, contrast, and sharpening to your exact taste. Of course, you can also aim for an output that is as pure as possible, even in compressed formats.

Additionally, you can leverage the dedicated Night Mode to extend exposures and apply more aggressive frame stacking to combat noise in low-light environments. Although we pride ourselves on providing the most natural image possible, adjusting processing is highly recommended for photos to ensure a beautiful final compressed image. We are also excited to announce that an all-new, improved, and unified photo mode pipeline is coming in version 4.5, so stay tuned for more!

What's the best transfer function/what log should I use??

There is no single "best" Log or transfer function because they all work well, provided you capture them correctly to maximize the usage of the tonality within their container. The goal is efficiency: packaging the light received by the sensor into the digital file with minimal waste and without bunching all data in a small container region.

For maximum post production latitude, Log is your best choice because it uses math to intelligently redistribute precision, protecting shadow details better than a Linear function, which is prone to errors during encoding if using classic gamma functions like Rec.709 with 8-bit. You must avoid clipping and maximize your usage of the container, a process that the Encoding Histogram helps you measure and verify.

Here is the key takeaway: If you correctly apply a Color Space Transform (CST) in your editing software to convert the Log footage to a standard output like Rec.709, all Log curves should render with the exact same image and color interpretation. Therefore, choose the Log that allows you the most efficient capture based on your existing workflow and matches your existing experience or cameras used if any.

"My image looks purple/pink/magenta!"

If your image is tinted purple or magenta, it means the raw color data from the sensor has not been properly interpreted or debayered by the app.

This severe color cast occurs because the app may be receiving uncorrected or misarranged Bayer patterns from the system, —the essential grid of red, green, and blue pixels—and misaligning the color channels, resulting in that aggressive magenta shift.

This most notably applies with full resolution raw streams in some cases, however some odd devices may use unusual Bayer arrangements that may cause this.

The fix is simple: you must Enable Remosaicing (or Demosaicing) in the app settings. This option forces the app to manually correct the alignment, bypassing the device's incorrect interpretation. However, keep in mind this correction only becomes active and necessary when the captured resolution is detected as being above 16MP.

If this still doesn't resolve the issue, report the device to the devs as it may be a one off exceptional grid arrangement needing a targeted fix!

“Will the app work on Huawei devices?”

This is where things get tricky, and the short answer is: No. It is extremely unlikely to work, and if it does, it will be crippled and useless for high-quality capture.

The core principle of MotionCam Pro is that it must have direct, low-level access to the pure RAW stream via the standard Android Camera2 API.

The issue with modern Huawei devices is their shift to HarmonyOS. They are not using the standard Camera2 API as usual, nor are they opening up the critical high-speed RAW stream on their later models.

This creates a complete software roadblock. Even if the hardware sensor is fantastic, the operating system and drivers will not expose the necessary raw capabilities that MotionCam relies on for capturing video. You will either find the app unable to function, or it will be capped at unusable frame rates (like 15 FPS), making it useless for professional video work.

If you are using a Huawei device, you should assume that MotionCam Pro will not provide the quality or functionality you expect. It's best to stick to devices that are confirmed to support the open Camera2 API and RAW video access.

Can the app work on iPhones or iOS?

Err… I neither confirm nor deny that it definitely will not not fail to succeed to run properly incorrectly nor be possibly improbable that it’s potentially a massive, code-breaking, firmware-wrangling, caffeine-fueled project currently being worked on in a windowless room.

Look, you’re asking about moving a highly optimized, low-level Android application (one that relies entirely on directly wrestling the guts of the Android Camera2API and specific driver quirks) over to a platform built on entirely different foundations. This is not a simple port; this is a highly complex, multi-dimensional physics problem.

For now, the only thing we can firmly confirm is that it’s not running today. ...Wait, hold on a second...

gets whispers in ear

...Oh yeah, an cheeck the community Discord. RAW Cam is here! You know where to look!

Is an SSD better for performance?

Yes, an SSD is generally a must for serious work with this app, as storage is the biggest bottleneck when shooting RAW video. Although not applicable with Direct Log mode, MotionCam Pro captures a monstrous amount of data, often pushing 300MB/s or more, and common storage solutions simply cannot cope. For instance, Micro SD cards are completely inadequate and should not be relied upon, as you will see dropped frames instantly.

Even your phone's internal UFS (internal SSD) storage, while fast, is often limited by device firmware and thermal management, which restricts your reliable recording time and sustained speed the moment serious heating begins (unless actively cooled or optimized).

If you want the maximum sustained performance, longer recording times, and a reliable workflow, an external SSD is one of the easiest solutions. By shooting directly to an external SSD connected via the phone's USB 3.0 or USB 3.1 port, you bypass the internal storage's thermal and bandwidth constraints.

This allows you to achieve the sustained speeds necessary to prevent dropped frames more often than not. Just ensure your phone's USB port supports at least USB 3.0 (and use a high-quality cable!) or the external SSD will be throttled, making it no faster than your internal UFS storage!

I keep getting an 'Error With Record Path'

An "Error With Record Path" is the app's way of telling you that it cannot write the stream of data to the designated location. This failure almost always boils down to a problem with either permissions or the storage module having problems connecting.

The most common cause is usually permission-based: the app must have explicit, uninterrupted write access to the recording location you selected. If you recently updated your phone or changed system settings, or if you just installed the app and did not designate the storage location yet for the first time Android may have revoked the necessary write permissions, or simply you may not have given them to begin with. To fix this, you must go into your phone's system settings, find the App Permissions for MotionCam Pro, and ensure storage access is fully granted and not restricted. You should also use the app's options to select where to write videos and render externally to!

I see a Throttling warning, is that bad?

Seeing a Throttling warning means your device's available peak performance power is being reduced. This CPU Throttling Indicator provides a real-time assessment of your device's thermal state and appears regardless of whether you are actively recording. Just because you see this alert does not necessarily mean you must stop recording; it simply tells you that your maximum performance is shrinking.

The severity of the issue begins to worsen based on the indicator's color. The Yellow Alert (THERMAL_STATUS_LIGHT) shows only mild reductions, primarily alerting you that heat buildup is beginning. The impact plays a more significant role with the Orange Alert (THERMAL_STATUS_MODERATE), where apps like MotionCam begin suffering heavier processing restrictions, and you may notice less stability. The Red Alert (THERMAL_STATUS_SEVERE) is the critical level, indicating a massive performance reduction where you should attempt to cool down the device, as the Android system may intervene further to prevent damage.

How Can I Cool Down My Device Or Extend Its Performance If It Overheats?

Cooling the device or reducing heating becomes more pressing once you see the Red Throttling indicator, as it means your session's sustainability is under threat. The most effective method is active cooling, which involves using a dedicated, external cooling solution. This usually means attaching a Peltier cooler or an advanced phone fan to the back of the device. These devices actively pull heat away from the battery primarily as well as from the processor and storage, preventing the thermal throttling mechanism from further limiting your power. Furthermore, you should offload the storage workload by using an external SSD, which removes the heat generated by the immense, high-bitrate write process from the internal UFS storage and surrounding circuits.

Beyond hardware, you can take steps to reduce the heat generated by the phone itself. You could remove any thick protective cases that act as an insulator and trap heat against the phone's chassis. Reduce the screen brightness to the minimum usable level, as the display is a major source of heat and power draw. Finally, eliminate unnecessary background workload by closing all unused applications and services. If possible, put the device into Airplane Mode to shut down power-hungry radios (Cellular, Wi-Fi, Bluetooth), dedicating the CPU entirely to the camera pipeline and maximizing your cooling efforts.

For advanced users with rooted devices, you can technically increase the thermal limiters in the system configuration to buy more time before throttling, but be warned: this choice puts significantly higher stress on your battery and internal components, potentially decreasing the overall lifespan of your battery cell!

Should I use OIS for Gyroflow?

No, you should almost ALWAYS disable the Optical Image Stabilization (OIS) actuator when recording video intended for Gyroflow stabilization, or even when creating calibration lens profiles for it!

Gyroflow is an advanced stabilization tool that uses the raw rotational data (gyroscope and accelerometer) captured by the device's sensor to precisely counteract camera movement in post-production. The problem with OIS is that the mechanical stabilization system in the lens shifts the optical element to keep the image stable, and the Gyroflow-compatible .gcv file records the movement of the device body, but not the independent movement of the stabilized lens.This creates a conflict: the gyro data no longer accurately represents the motion of the captured video frame, resulting in warping, jitter, and poor stabilization results.

The movements introduced by the OIS mechanism itself are typically not accounted for by the software, which can interfere with the stabilization. Therefore, for the cleanest, most predictable, and most reliable stabilization, always ensure OIS is turned off in your camera settings before capturing your footage.

“I can’t crop my photos in-app!”

That is correct, the photo cropping function is not currently available in the app. While the application allows users to alter certain aspects of the photo processing before export, the function for defining a crop is deliberately excluded in the current version.

The app is designed to capture the full raw stream from the sensor (often referred to as 'open gate'), ensuring you retain the maximum amount of data and framing flexibility for post-production. All final cropping, as well as complex editing and grading, must be done later in a dedicated desktop editor.

This limitation may change, as a cropping feature might be added in a future update, but it is not an option yet.

“Long exposures are not applying or working past 0.5s”

If your exposure time will not extend past 0.5 seconds, this is because the app requires you to manually engage a specialized mode for slower shutter speeds. The app's interface is designed to reflect real-time adjustments, and once you attempt to exceed 0.5 seconds in length, continuous real-time refreshing of the preview would slow the application down significantly. To prevent this performance hit, the app stops further real-time changes.

To enable the selected longer exposure, a SET SHUTTER SPEED button will appear on the left side of the screen as soon as you select a value slower than 0.5s. You must press this button to engage the extended exposure mode. After pressing the button, the selected slow shutter speed will be locked in, allowing you to capture long exposures up to the maximum supported by your device's camera API. The key part is your API may limit you to as little as 1/4s durations however so beware! App can only use what it’s allowed to unfortunately! And yeah… Exynos love this limiter too…

Note that the image on the screen may take a moment to refresh and might appear frozen. This is strictly because the camera is capturing at the requested slow speed, so the resulting image is not instantly reflected on the display.

“My log output is way too dark and doesn’t look like it should!”

This is a common observation when shooting in Log video modes, and it is usually not a fault with the application but rather a consequence of the Log transfer function. Log gamma curves are specifically designed to compress the maximum dynamic range of the scene into the video file, and they do this by pushing the midtones and shadows down. This results in a flat, desaturated, and often very dark-looking image on the screen or in your initial raw output, even though all the data is correctly recorded.

To correct this encoding flaw and ensure you are utilizing the full tonal container (best visualized by the Encoder Histogram) you often need to intentionally expose the image much brighter than you would for standard video. You may need to boost the exposure gain significantly, often to a setting like +3 EV Gain, to shift the shadow and brightness distribution of the scene to the right. This practice, known as 'Exposing to the Right' (ETTR) with Log, helps you avoid bunching up all the data in the limited tonality available in the dark (left) side of the container. By properly distributing the tonality across the container, you ensure a cleaner, higher-quality image after you apply the final color correction LUT in post-production.

Why can’t I view MCRAW files on a video player or edit them directly?

You cannot view MCRAW files directly on a standard video player because they are not compressed video files; they contain the untouched full Bayer data for every image frame of the sequence straight from the camera sensor. Standard video players are designed to read highly compressed, finalized formats like H.264 or H.265, which contain all the necessary decoding instructions for instant playback. MCRAW, however, retains maximum image quality and flexibility but requires specialized processing.

To work with MCRAW files without encoding them, you have several options: you can convert the files into the CinemaDNG (cDNG) format for use in editors like DaVinci Resolve, use the MotionCam Fuse program to create instant virtual proxies and real-time conversion, or leverage the few available post-production programs like MLV App or VKDT that have native MCRAW support. For quick viewing with maximum quality, you can also use the purpose-built MCRAW Player program for PC.

Do note that the app itself will be able to view these files with sound in real-time as of the upcoming version 4.5!!

“Why can’t I find my MCRAW files on a file explorer”

The reason you cannot find your MCRAW files using ab Android file explorer, whether on your Android device or when connected to a PC via USB, is likely due to the Private Storage setting within the app.

When you select the Capture to MotionCam Private Storage setting, the app is instructed to write your MCRAW captures into the app's protected, 'private' area on the system level. This location is generally inaccessible to conventional file explorers, user accounts, and is hidden from view when connected to a computer. The benefit of using this private storage is that it can provide faster storage I/O and eliminate system write speed bottlenecks on devices which have firmware restrictions or max speed restraints. The major downside is precisely that the files will not show up on file explorers.

To make your MCRAW files visible and accessible for transfer or post-processing, you must first use the in-app function labeled MOVE TO RENDER FOLDER from the MCRAW Workbench. This option transfers the MCRAW files out of the app's private storage and into the designated render folder you’ve chosen, which is publicly accessible and will resolve the issue!

Why can't I view ProRes files?

The primary reason you cannot view ProRes files directly on your Android phone or a standard media player is a fundamental limitation, Android devices do not have hardware decoding support for the Apple ProRes codec.

ProRes is a proprietary, intermediate video format, and its high data rates and specific structure are not some meant to be shared with Android devices, full stop.

When a device lacks the dedicated hardware decoder chip for a codec, the entire burden of decoding the file falls onto the CPU. This is known as software accelerated decoding.

While a dedicated media application like VLC media player can perform software decoding for ProRes on Android, it is extremely taxing and much slower. The massive computational load required to decode high-bitrate ProRes files in real-time overwhelms many lower end CPUs, causing playback to be choppy, stutter, freeze, or fail entirely.

The files are meant to be transferred to a desktop computer running professional video editing software for editing, color grading, and final conversion into a low-bitrate delivery format like H.264 or H.265 before being viewed smoothly on a mobile device.

"I deleted my mcraws accidentally, can I recover them?"

If you have accidentally deleted your MCRAW files, you need to understand that the likelihood of recovery depends heavily on how and where the files were deleted.

The critical distinction is the method of deletion: If you delete the MCRAW files from within the app itself, regardless of whether they were in the app's private storage or a public folder, the deletion is immediate and permanent. The app's built-in file management function executes a final removal command, and the files are not routed to a trash bin, making recovery highly unlikely.

However, if the MCRAW files were accessible (meaning they were saved outside of private storage) and you delete them using a standard file explorer or a connected computer, they may fall into a 'junk' or 'trash' bin. This is not a guaranteed feature and depends entirely on the specific operating system and file manager you are using (e.g., Windows Recycle Bin, Samsung's or Google's dedicated Android trash features, or a third-party file manager with a recycle bin enabled).

Because MCRAW files are large and may hold critical data, it is essential to be extremely careful during deletion and to rely on immediate, regular backups to a separate computer or external drive for file safety. The app will generally ask to confirm deletion commands when pressed, so always double check!

Is there any way to set up an intervalometer function?

Yes! there are two primary ways to achieve an intervalometer function (a method of capturing still images or video frames at regular, set intervals) using the app.

The first method is by utilizing the dedicated 'Timelapse Mode Settings'. This mode's menu allows you to precisely control the capture interval between individual frames using the Time Between Capture (Seconds) slider, which can be set between 0.25 seconds to 30 seconds. You can also define the total duration of the sequence with the Total Capture Time (Minutes) slider, which is adjustable in 5 minute intervals up to 6 hours, or overridden using the Unlimited Capture Time checkbox.

The second method is to use the RAW Video mode and select a very slow Frames Per Second (FPS) value, such as 1 FPS or 5 FPS. By intentionally dropping the frame rate significantly, the recording cadence is slowed down massively and with equal spacings per frame duration per second, which effectively achieves a similar spaced-out capture effect to an intervalometer!

There’s a black/green bar on my video and image is cropped even though I used open gate!

This issue, where you see an unexpected black, fuzzy-lined, or green bar on your video and find your image is cropped despite selecting an Open Gate stream, is not a bug in the app but rather an indication of an underlying problem with your device's OEM (Original Equipment Manufacturer) RAW stream implementation.

When you select an Open Gate Stream (designed to use the full sensor surface), the app expects full resolution data even before you crop, but the OEM may fail to deliver a functional stream at higher settings, such as 60FPS. If the device is internally limited to a cropped stream (like 16:9) at those framerates but doesn't properly communicate this by only giving you a single open gate raw stream, the data is pushed into the wrong container, creating "dead" regions where no image information was captured. You most likely won't have a cropped RAW stream available to select to fix this if this happened, and unless one is available, this broken stream behavior is an unavoidable annoyance.

This glitched area may appear differently depending on how the stream is broken. It may be centered, off-set, or mostly at the top, and could show nothing, shifted warped areas from the cropped regions, or even duplicate off-centered data, meaning the final appearance may greatly vary.

The immediate solution is to experiment with your device's available RAW Streams or reduce the frame rate until the camera delivers a stream that correctly maps the sensor data to the full container size. Although this issue is annoying, if the image data itself is good, the glitched section can be simply cropped off in post-production within your editing software. Alternatively, the MotionCam Fuse program can often correct this issue automatically by properly interpreting the RAW data and correcting the broken stream mapping by chopping it off and aligning the frame correctly.

A native app function to trim this dead region off if it should occur is also currently in development and coming in a future update.

Lastly, it’s worth mentioning that if you rooted your device, it’s often possible to pry open all of the ‘correct ‘ raw streams via two simple build.prop lines! They’re as follows if you want to try them..!

persist.vendor.camera.exposeFullSizeForQCFA=TRUE

persist.vendor.camera.maxRAWSizes=19

What’s the point of MotionCam Tools? Do I need to use it at all?

The MotionCam Tools is a legacy PC application that served as the original method for processing MCRAW files. However, it has been largely deprecated in favor of the newer and superior MotionCam Fuse utility and the app's powerful integrated renderer.

Many users, particularly rookies, mistakenly believe that MotionCam Tools are mandatory for processing MCRAW files because they do not realize the app itself has a fully integrated renderer with much higher quality. This is a rookie mistake and not necessary whatsoever!

MotionCam Tools remains a backup method, but should generally be avoided because they are much slower than newer solutions, use older, less effective algorithms compared to MotionCam Fuse and the in-app processing for demosaicing and noise reduction, and their original video encoding options have been removed due to their older and lesser quality, leaving them mainly to render MCRAW files into a cDNG (CinemaDNG) sequence on a PC.

The only compelling reason to use MotionCam Tools now is as a backup solution for MCRAW Recovery, as they can sometimes salvage files that may be broken or corrupted due to device-specific recording issues or rare anomalies during recording. Otherwise, you should use MotionCam Fuse (which gives much quicker results and can directly mount MCRAWs to DaVinci Resolve) or rely on the superior in-app and offline rendering functions for processing your footage, and the MotionCam MCRAW Player for viewing your MCRAWs at full quality.

[6] MotionCam Workflows: Find Your Persona (1/2)

Discover your MotionCam personality! Understanding your shooting style and needs will help you conquer the settings and modes to achieve footage you'll actually love. Take our hoRAWscope test then jump ahead once you've chosen your path!

Persona	Skill Level	Core Goal	Workflow Description
The RAWligious Priest	Elite	Absolute RAW Purity	You love rare meat, and if it's not Wagyu, why bother? The only way to have coffee is black. Plan B is Plan A. The letters VKDT are not jibberish to you, and you don't go to the spa; you get RawTherapee. You are an absolutist who records exclusively in MCRAW. You accept massive file sizes, high data rates, and fully embrace desktop processing (using MotionCam Fuse or cDNG pipelines) with exotic tools such as Neat Video as the price for maximum image quality. You prioritize preserving every single bit of sensor data for meticulous color grading, because you know the truth is in the untouched pixels. You put the Resolve in the DaVinci.
The Starving Artist	Advanced	Professional Production	You know the difference between a lens flare and a highlight bloom. You treat your phone like a proper cinema camera, mounting it to a cage and connecting external sound and monitoring gear. You enhance your existing Arsenal with all tools at your disposal and your phone is another one of them. You shoot RAW Video or Direct Log (in log modes) strictly based on needs and prioritize clean, flexible data that your desktop editor (like DaVinci Resolve) can chew on later. You value speed, but you absolutely demand professional-grade results.
The Mobile Warrior	Intermediate	Efficient, Mobile-Only Workflow	Your phone is your camera, your edit suite, and your delivery platform. You are the master of efficiency. You shoot in Direct Log to maintain gradeability, but you use the app's powerful integrated renderer to quickly process and export your footage. You render directly to high-quality codecs (AV1, APV, ProRes if available) on the device, ensuring a fast turnaround without ever touching a desktop computer.
The Smooth Operator	Intermediate	Fast, Gradeable Footage	You've got an eye for composition, but you don't have time for multi-terabyte drives. You live by the Log and die by the Gamma. You use Direct Log footage because you value the extended dynamic range, but you often skip the RAW headache. You handle a simple color-grading pass in a quick mobile editor or the app's renderer, sacrificing the extreme flexibility of RAW for smaller files and a much faster overall process. You use PC editors but won’t overdo it.
The Interbrolometer	Intermediate	Time-Based Creative Capture	You are secretly a time lord. You don't just record events; you accelerate or slow them down for dramatic effect and accentuate the passing of time and the celestial skies. You utilize the app's Timelapse Settings Menu for precise intervals, carefully setting the time between captures rather than taking in the whole barrage. Alternatively, you achieve similar effect by shooting in RAW Video mode at an extremely slow frame rate to create stunning time-compressed sequences. Video to you means rapid photo reels!
The Bursta RhAWs	Basic	High-Speed Action Capture	Stills are life, bursts are life insurance. You focus on capturing fleeting action with the guarantee you won't miss the shot. You prioritize Burst Mode or shooting RAW Video with the explicit intent to scrub through and retrieve the single perfect still frame in post. Your mantra is "Burst first, ask storage questions later," ensuring maximum coverage of the action.
The "Shut Up, Geeks."	Entry-Level	Good Video, Zero Hassle	You are allergic to settings menus. You just want a button that delivers a great-looking, finished video without any homework. You're guided to use Direct Log with minimal settings tweaks and apply a simple LUT conversion profile during the render. Your goal is a perfect, delivery-ready file straight out of the camera with the highest quality possible for the least amount of effort. You’ve discovered the shortcomings of the stock app’s processing however and you can no longer look back.
The Vlogmeister	Entry-Level	Travel and Everyday Footage	You're on the move, and every moment is content; your ‘transfer function’ is a social media app. You prioritize manageability and shareability. You want fast rendering to small, standard file sizes and simple, pleasing image that is ready to upload directly to social platforms or YouTube without delay. You're looking for the sweet spot between decent quality and instant gratification.
The Light Hunter	Basic	Single-Frame Excellence	You are a still shooter who appreciates a powerful sensor. You use the app not just for motion, but for the absolute highest quality single image possible. You focus on capturing RAW (DNG) photos and JPEGs carefully capturing single, perfect frames from a RAW video, treating the app as a professional-grade capture tool for stills. You detest AI processing and crave natural results.

MotionCam Workflows: Find Your Persona (2/2)

Persona	Skill Level	Core Goal	Workflow Description
The Mad Programmer	Master Debugger	App and OS Control	You love the camera, but you love the code more. You are an extremely talented individual who appreciates the app not just for shooting, but for the pinnacle of control and technical achievement it represents and let's you harness. You routinely heavily modify Android, perform excellent debugging, love sensor/imaging theory, and contribute greatly to the community by developing root mods and custom kernel tweaks. You are insatiable in your curiosity and value technical mastery above all else—just don't ask you for a color-graded sequence, as creativity takes a backseat to code.
Dr. Androidstein	Extreme Hardware Modder	Phone-to-Camera Conversion	"Why buy a camera when you can craft the camera?" You are the hardware Dr. Frankenstein of Android, physically modifying your phone to achieve extreme photographic results. Your workshop is full of external lenses, custom cooling solutions, and specialized sensor attachments and custom made filters. You might straight up open up the sensor element and attach actual glass optics to imbue them with mechanical contraptions. Some users see your work as mad and unnecessary, but your results, while extreme, are often insanely unique and push the boundaries of mobile imaging into realms never thought possible.
The Specs Junkie	Expert Benchmarker	Theoretical Maximums	Your creative output is measured in frames-per-second, not finished films. You obsess over every patch note, firmware release, and API update. You constantly think about insane, overkill features that are either technically unnecessary or would take a team of engineers six months to implement. You are the anti-thesis of 'talk less, create more,' often engaging in non-stop benchmarking and device comparisons but rarely producing anything creative or purposeful. You thrive on the potential of the newest device and the app's capabilities on paper, and while you're invaluable for testing the absolute technical limits, you often get "addicted" to the next big feature before using the one you just got. Your life goal is to hit 8K 240 FPS on a flagship phone just to prove it can be done. The dark side of the RAW, but dammit, isn't it fun at times...?

[6.1] -THE RAWLIGIOUS PRIEST-

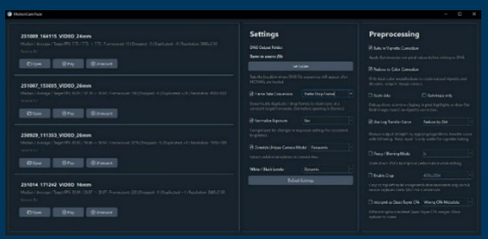
If you identify as The RAWligious Priest, I'm not going to spend time spoon-feeding you. You understand that image purity requires embracing file size and intensive desktop processing and you're probably packing storage to spare anyways. MotionCam's proprietary MCRAW format will be your primary weapon of choice, and while a near-future update will be introducimg a Direct Log Video mode for cDNG, you will likely continue to embrace MCRAW files for their numerous technical advantages. This workflow is may not be for the faint of heart, but it is the singular path that yields unparalleled visual fidelity and grading latitude.

You already grasp the inherent limitations of the CinemaDNG (cDNG) format in a mobile context. Although cDNG is open-source, it's rarely found in high-end production cameras for a reason. The core problem lies in file fragmentation: you are creating a new DNG frame container for every frame recorded. This means generating a ton of individual DNG frames for every second of footage, which is simply clunky and inefficient. This places an enormous burden on your phone's Input/Output (I/O) subsystem, constrained by the short time required to compute a container creating operation, a basic division of 1 second ÷ framerate amount, which severely limits recording stability and transfer speed. Furthermore, the DNG container can become susceptible to dynamic range losses once necessary corrections, such as vignette correction, are baked into the file during the export process, as some phones often do, impacting how you can store the pure Bayer data.

The MCRAW format was developed specifically to circumvent these I/O and data integrity issues. MCRAW files are impervious to the cDNG limitations as they fundamentally act as a "Dump" for all the pure Bayer data from the sensor, storing it as one direct stream into a single file. This radically reduces I/O load, making the files easier to move around and significantly improving recording stability. Additionally, MCRAW achieves data lossless compression ratios that simply cannot be matched by cDNG; it is capable of up to 50% data lossless compression over standard uncompressed RAW, allowing it to compete with high-end formats like ProRes RAW HQ in certain light situations. The compression ratio is inversely dependent on light, meaning that as you expose more or add more light, you can expect smaller file sizes accordingly.

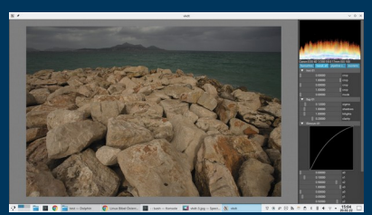
If you noticed that we said MCRAW is compressed RAW video and raised an eyebrow… Then congratulations! You are indeed a RAWligious priest. The key part is that it's DATA lossless, a key distinction. Unfortunately as you know, due to the RED visually lossless/data lossy compression patent, MCRAW currently cannot be compressed in anything less than data lossy (must be fully reversible) manner, else you open a box of legal horrors. It is my opinion that whichever legal clowns granted RED this patent should go sodomize themselves with retractable batons for the injustice they've pulled on the industry as a whole. If you must compress RAW for visually lossy purposes, many in the community have generally found "slimRAW" to be an excellent solution, but it's not free unfortunately, and only works with cDNG.

Where was I? Oh yeah… If you use DaVinci Resolve, MotionCam Fuse is the technical marvel you should be leveraging to mostly avoid the above headaches. Fuse is not a classical file converter; it is an Operating System-level utility implemented using the Filesystem in Userspace (FUSE) mechanism. When Resolve, which is expecting a file-per-frame cDNG directory structure, attempts to access the single MCRAW file, Fuse intercepts the file system request. It then acts as a Virtual File System (VFS) driver, presenting the internal, compressed MCRAW data stream to Resolve as if it were a directory full of uncompressed DNG files.



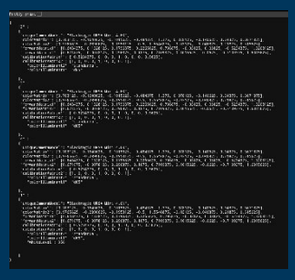
Fuse performs the necessary on-the-fly translation and decompression of the contiguous raw Bayer data stream for each requested frame, delivering it directly to Resolve's memory. This process provides substantial technical and workflow benefits over traditional in-app cDNG export: the elimination of the hours-long, compute-intensive step of rendering a physical cDNG sequence, resulting in zero pre-processing time, and significant storage efficiency by requiring only one copy of the footage on your desktop. Crucially, Fuse performs several high-quality, automated post-processing corrections, including essential tasks such as frame rate corrections (to ensure accurate playback speed) and exposure normalization (to compensate for in-camera exposure shifts), all while utilizing superior, up-to-date demosaicing algorithms compared to legacy options.

While Fuse provides the fastest path into Resolve, direct compatibility for MCRAW exists in other high-end RAW processing ecosystems. MLV-App, a popular tool derived from the Magic Lantern community, offers integrated support for decoding and rendering MCRAW files, positioning it as a highly competent auxiliary processing alternative. Similarly, the open-source RAW processor VKDT features native support through an input module, allowing for direct processing of the pure MCRAW data stream. For users dedicated to the Adobe suite, a specific MCRAW plug-in for Adobe Premiere Pro is available for download on the official MotionCam website, offering a path for direct ingestion into that NLE without requiring the cDNG intermediary.



Additionally, the pursuit of purity requires leveraging every tool. For ultimate technical control, the Calibration JSON feature allows you to alter the raw data's color matrix, enabling you to create or tweak color profiles, or even emulate the RAW data of specific camera models.

If you intend to use the in-app renderer to export cDNG, this emulation can unlock specialized color controls otherwise hidden in Resolve, such as Blackmagic Design RAW options or Panasonic V-Log tab settings. Additionally, for in-app renderings you should leverage the 'Scale Data' and 'Highlight Reconstruction' settings in the event you do not plan to correct the vignette yourself, as failure to do so will potentially impact your maximum dynamic range during the container conversion. Lastly, **ensure you toggle the 'Uncompressed DNGs' option if you use Adobe Premiere** without the plugin. Although most high-end users are on DaVinci Resolve where this isn't an issue, Premiere is notoriously incompatible with the compressed DNGs MotionCam produces, requiring the full, uncompressed file structure.



As far as the legacy MotionCam Tools program, it is largely deprecated, offering inferior algorithms and speed compared to Fuse and the in-app renderer; its only compelling use case is as an ancillary option for emergency MCRAW Recovery on corrupted files. That said, some users do have a fondness for these Tools, so hence you may still see them around both on the website as well as commonly in leveraged by some users. Nevertheless, they're no longer the suggested workflow!

As far as Noise Reduction goes, the community has widely come to the consensus that if you've got the skillset to match it, you may want to try out Neat Video as it's incredibly powerful with with RAW Bayer data, however MotionCam itself also offers internal cDNG frame stacking solutions that provide rather robust denoising if you're looking for an integrated solution!

If you read this far and wish you could simply plug in MCRAWs into your favorite editor like Resolve; you're not alone! Although Fuse is amazing, many users do prefer a direct native support, so make sure to join us in requesting program developers and companies embrace native MCRAW support! Strength in Numbers!! Now, thank you for coming to my TED talk…

[6.2] (USEFUL RESOURCES FOR A RAW PRIEST)

Petition to Blackmagic Design to add MCRAW Decoding Support for DaVinci Resolve
https://forum.blackmagicdesign.com/viewtopic.php?f=12&t=198544

MotionCam Fuse README
https://github.com/LeonardSander/motioncam-fs

MotionCam Fuse Download (Windows)
https://github.com/LeonardSander/motioncam-fs/releases/tag/0.10

MLV-App (Native MCRAW support enabled)
https://mlv.app/

VKDT/Vulkan Darktable (Native MCRAW support enabled)
https://github.com/hanatos/vkdt

MCRAW decoder library (for Native MotionCam RAW format reading implementation purposes)
https://github.com/mirsadm/motioncam-decoder

MotionCam Tools (Windows/Mac + Adobe Plug-in)
https://www.motioncamapp.com/

MotionCam Tools Legacy Versions
https://www.motioncamapp.com/legacy-tools

slimRAW (CinemaDNG lossy compressor, paid)
https://www.slimraw.com/

Neat Video (Video Denoised Plug-in, paid)
https://www.neatvideo.com/

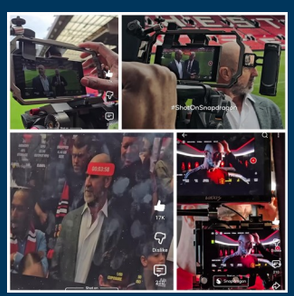
[6.3] -THE STARVING ARTIST-

If you identify as The Starving Artist, you aren't just here to capture "content"; you are here to make films. And believe us, it's been done before; Google, Samsung, Qualcomm, and many other users have used the app to successfully craft professional grade content meant for actual professional tier delivery! Some have even managed to earn a living this way!

You understand that while RAW is the nuclear option, your deadline was yesterday and your storage drive is already getting full. You demand professional-grade results, but you value a workflow that doesn't require a supercomputer to render a 30-second clip and can survive the rigors of a working environment. Your philosophy probably is around "Maximum Quality, Minimum Friction, Client Won't Care". We need to define your roadmap, balancing the ultimate power of RAW Video with the surgical precision of Direct Log since you're probably already familiar with core essentials, but must understand the app's restraints to efficiently put your skill to use. Your primary workhorse will likely be Direct Log Video. This mode is vital because it still completely bypasses the phone's internal image signal processor (ISP), which typically bakes in ugly sharpening and noise reduction. Instead, Direct Log mode encodes the untouched sensor data on-the-fly into a high-quality, edit-ready codec like ProRes or 10-bit HEVC. This gives you a large portion of the grading flexibility of RAW, but with a fraction of the storage and post-production headache.

However, you need to properly understand why Direct Log is actually a massive stress test for your device, often more so than RAW. Many users mistakenly believe that encoding to a smaller file is "easier" than RAW, but the opposite is true: when shooting standard RAW, the device simply captures the sensor data, compresses it (into MCRAW), and dumps that stream directly to storage—it's one heavy, high-bandwidth task. In Direct Log, the device must simultaneously execute two colossal tasks: first, it captures the raw sensor stream data in a buffer, and second, it performs an on-the-fly/real-time high-quality video encoding of that stream (eg. into ProRes or HEVC) before writing the final compressed file. This concurrent RAW processing plus heavy encoding places an intense, sustained demand on your CPU +GPU and cooling system as you're not taking advantage of the otherwise poisoned chalice that the ISP otherwise provides in return for its efficiency. Use the memory buffer and encoding threads settings to help!!

This means you must tread carefully with high frame rates; if you attempt 60fps HEVC in Direct Log, or any setting/codec that's too intense for your device's capabilities and see that dropped frame counter tick up, or if your device starts to struggle with heat, don't try to force it. We strongly suggest securing the shot by switching to standard RAW Video capture and then using the Offline Renderer later to quickly generate your Log files on your phone itself at your leisure, bypassing the thermal strain on your phone. When setting up your shot in Direct Log, you cannot treat exposure like you would on a standard camera app. You likely know Log curves are designed to preserve dynamic range by pushing shadows and Midtones up to preserve data while bringing down highlights, which often results in a flat, dark, and desaturated image on your screen. To utilize the full tonal range of your 10-bit container that's often present in Direct Log codecs, however, you should employ the encoding options such as the Gain Slider to Expose to the Right (ETTR), digitally speaking! You shouldn't be afraid to push the digital Gain to +2 or even +3 EV to shift your encoder histogram data away from mostly shadows and into the cleaner midtones to spread out the data coverage.



This brings us to a crucial distinction: you will now have two histograms to monitor. The primary RAW Histogram shows the absolute, uncompressed data from the sensor—this is the true source of your information and must always be protected from clipping in the far right. The secondary Encoding Histogram (which only appears in Direct Log) shows the data after the Transfer Function has been applied and before it is encoded. You must constantly monitor the Encoding Histogram to ensure your data is spread across the graph and not artificially bunching up in the dark left corner due to the Log curve, and certainly not clipping the top right of the graph. Also, while you may rely on Log for day-to-day efficiency, we suggest you view RAW Video Mode (MCRAW) as your ultimate, failsafe insurance policy. It is for scenes where you cannot afford instability or the hard limits of the encoder. In scenes with extreme dynamic range or complex mixed lighting, an encoder might irreversibly clip highlights or crush shadows, limiting your grading options. RAW eliminates this risk by preserving the full, untouched sensor data. Because RAW recording is just a data "dump" without the heavy encoding overhead, it is often more thermally and computationally stable than Direct Log in demanding situations. If the shot is once-in-a-lifetime and you cannot risk a performance hiccup, encoder instability, or lost detail, that's when you deploy the nuclear option.

Additionally, Direct Log does carry a professional weakness that you must be wary of, particularly when shooting in very low light scenes. Assuming professional usage, you will probably use Expose To the Left methods, meaning you will bring your own studio lighting or equipment to bridge dynamic range. However, if you do not do so, be extremely wary of low light as Direct Log will run the RAW data into encoding with all the added noise. You can potentially choke the encoder by running a clean, pure noise feed, as it's extremely difficult to predictably encode otherwise chaotic noise patterns. What this means is way worse encoding efficiency and effectively lower useful bitrate (with a higher performance tax). An update is coming to address this via real-time noise reduction capabilities, but until then, use external lighting or choose the RAW option if lighting is marginal.

Regarding storage, while internal recording might suffice for basic, intermediate Log files, an external USB-C SSD should be a key component of your professional rig. It is absolutely required if you choose to shoot RAW or use heavier ProRes codecs to handle the high data rates for longer periods (unless you've got a 512GB/1TB device); aim for sustained write speeds of at least 350MB/s to ensure stability and ideally vet the model in our community as some do not possess the required sustained endurance despite their advertised write speeds. For Direct Log, while less critical, the SSD becomes an essential, non-throttling tool for instantly offloading and managing your finite storage quickly on set. Some users (myself included) even go as far as adding external cooling to the device in the form of Peltier/Phone Coolers, which provide massive relief to the thermal issues that appear under these intense uses. They're indispensable when you cannot afford thermals hiccups and extremely cheap too, so I'd strongly recommend one.

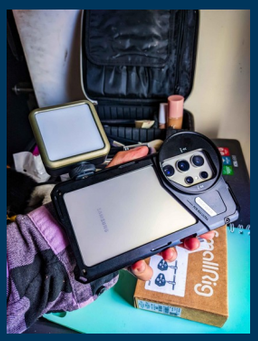
Furthermore, you need to be aware of color fidelity issues unique to phone sensors. You may encounter devices (like some Pixel phones) that tend to desaturate RAW data significantly, while others (like Nubia devices) saturate heavily. To counter this at the source, you should utilize Custom Matrix files (.json). These files allow you to alter the matrix coefficients of the RAW stream data directly, essentially creating a custom color profile that fixes these saturation variances or even emulates the specific color science of cinema cameras before you grade. Your post-production pipeline should be disciplined, preferably running through DaVinci Resolve as many of the app users use. You could skip the random LUT packs and go straight for the technically correct approach: the Color Space Transform (CST) node. You should use the CST node to transform your Log footage (eg. Input of DaVinci Wide Gamut + DaVinci Intermediate) into your timeline space and then into your delivery space (Rec.709). Perform all your primary and secondary grading in between these transforms.

Lastly, be mindful of Audio Drift—a common symptom of Variable Frame Rate (VFR) recording caused by performance hiccups. If you encounter it, the solution is not to blame the editor, but to transcode the clip to a Constant Frame Rate (CFR) using a simple tool or to use a utility like MotionCam Fuse to stabilize the video structure. You can also opt to separate the clip Audio and Video to ensure the correct playback framerate is applied as oftentimes the audio component gets rounded down triggering a mismatch versus the video! Your ultimate goal is to spend your time being creative and grading, not fighting your tools!

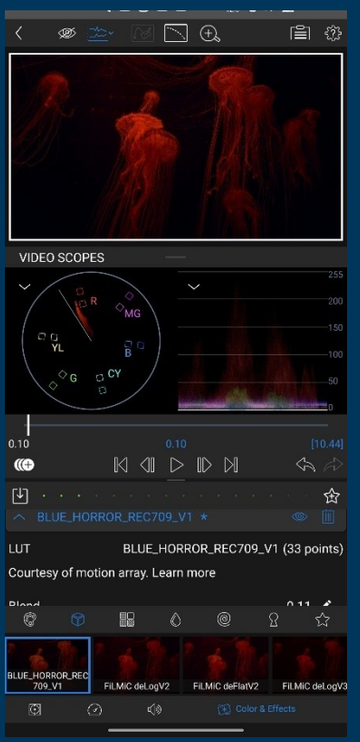
[6.4] -THE MOBILE WARRIOR-

Your approach as The Mobile Warrior is defined by a singular, non-negotiable goal: maximum quality and speed, phone only. Your philosophy is: "The best rig is the one on me, and it better do it all." You are a one-person army focused on getting the shot, encoding it efficiently, and getting it online fast; anytime, anywhere. For you, the ultimate quality of RAW is often the artisan option that is reserved for the best shots only, as its processing simply takes too much time for everyday use. I myself identify as this user!

Your workflow is centered on a dual approach: Direct Log is your primary weapon, and RAW processing via the In-App Offline Renderer is your heavy-hitter option, although your preference is Direct Log Video; this is your one-and-done solution. You prioritize this mode because you are acutely aware of the limited options for HDR grading on Android, where powerful mobile editors capable of handling complex HDR pipelines (like LumaFusion) are the rare exception, not the rule. To be clear, you CAN plug in 10-bit inputs into mobile video editors and retain their editing latitude, however you can only export them in 8-bit end results in all of them except for Lumafusion, which can continue to grade in HDR and export in 10-bit (an ultra rare capability on Android).



Given the continued prevalence of 8-bit SDR deliveries for universal compatibility across most social media platforms and standard devices, mastering the first shot, first export from MotionCam becomes a must. You accept that you initiate data loss (as it's an encoded file), but you gain immediate access to a usable, high-quality file that avoids later additional processing. Android operating systems cannot natively edit ProRes files, which makes the codec an immediate logistical bottleneck. While new codecs like Advanced Professional Video (APV) are emerging to provide a suitable intermediate format for Android users, for now, your best option is the efficient HEVC Codec, namely in 10-bit. You do however avoid the massive file sizes of ProRes and its overhead, recognizing that its primary benefit lies in handling multi-layer, complex desktop timelines—a task your mobile-centric workflow avoids.



Since on-device convenience is the name of the game, it's of the essence that you rely heavily on In-App Camera Processing to minimize work in post. The best way to achieve this is by using the app's Direct Log Bake-In LUTs, render tweaking tools and Direct Preview Overlays for Log captures. When appropriate, you should rely on baking in LUTs during the initial encoding to minimize re-encoding and preserve the highest quality before the final compression. By applying a Rec.709 or other delivery-appropriate LUT during MotionCam encoding, you ensure your file leaves the app ready for final cuts only, preventing additional lossy edits.

Similarly, you must use Preview Overlays (like Direct Preview's Sensor Clipping and False Color modes) to ensure the scene is captured once, captured right, preventing the need for massive exposure adjustments later. This reduces your post-production to quick cuts and minor adjustments.

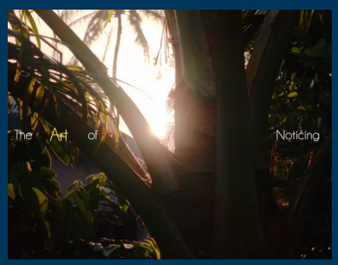
When you absolutely must capture a challenging scene with extreme dynamic range, unpredictable conditions, or need the peace of mind for a once-in-a-lifetime shot, you deploy RAW Video (MCRAW). Critically, you only process RAW in-device and avoid the hassles of more granular desktop processing. You should utilize the In-App Offline Renderer (your heavy-hitter) to tap into the full potential of RAW, performing advanced, non-real-time processing (like superior Denoising and Noise Reduction) directly on your phone or tablet. This two-step process (RAW capture followed by In-App Processing) is your method of harnessing the highest possible quality, while remaining fully mobile, reserving the full power of RAW for when you truly need it without breaking your mobile-centric commitment.

You must understand the performance trade-offs: Direct Log's heavy, real-time encoding can easily limit you to standard frame rates (like 24 or 30 fps) on many devices. In contrast, you can sometimes use the lower computational load of the RAW mode to pull higher, more stable frame rates (like 60fps), as it's simply dumping data to storage without the simultaneous encoding tax.

For stabilization, a pocket mobile gimbal is highly recommended to achieve smooth, usable footage without relying on processor-intensive software correction. If you must rely on software stabilization, you will likely accept the limitations of basic in-editor stabilization, or even use the surprisingly effective correction available in the Google Photos app if absolutely desperate, as the extra step of using Gyroflow (while superior) is too much effort for your convenience-focused workflow. For storage, bulky external SSDs are used for ultra long RAW sessions, but external drives are primarily a mobile dump/cache to quickly offload massive RAW files until you can render them down to smaller, manageable Log files using the Offline Renderer on your device

[6.5] -THE SMOOTH OPERATOR-

Your approach as The Smooth Operator is defined by a passion for creative visuals without the need for professional, client-driven deadlines. Your goal is simply great video for the sake of great video. You are the classic consumer who knows their way around a camera and seeks to bridge the gap between simple phone video and cinematic quality. While you are not strictly mobile-only, your editing remains simple, using a hybrid workflow that touches both mobile editors and beginner-friendly desktop editing software.



You are here because you seek the technical control that modern phones are prominently taking away from the user. You are disgusted by the oversharpening and heavy-handed denoising still present in other mobile video solutions despite their marketing claims, and you outright detest the invasive AI image pipelines that interfere with your creative intent.

Your philosophy is "A cinema camera only becomes such with an operator to match. My phone has one". For you, the phone does not restrict your creative vision; it is merely the capable vessel for capturing it, anytime, anywhere, and always in your pocket! You are likely the average serious MotionCam user!

Your primary weapon will likely be Direct Log Video, and you primarily shoot in 10-bit Log. Unlike the convenience-focused Mobile Warrior, you do not shy away from ProRes as your codec, accepting the massive file sizes for the superior editing performance and reduced rendering overhead it offers on your desktop Non-Linear Editor/Video Editor (NLE). You avoid the complex, massive, and bandwidth-intensive RAW format entirely, viewing it as the domain of professional colorists. Instead, you trust the highly efficient and high-quality Direct Log output as the perfect starting point. Since you plan to do a light grade later, you do not bake in your final look.

Your primary focus in MotionCam should be exposure purity, relying exclusively on Preview Overlays like Sensor Clipping and False Color instruments alongside the Encoder Histogram to ensure the maximum dynamic range is captured within the Log curve, treating the Log file as a blank canvas.

Your workflow is simple: for quick social media cuts, you drop the file straight into a mobile editor and apply a simple conversion LUT. For polished projects, you transfer the files to your desktop and use an easy-to-manage NLE (like DaVinci Resolve Free). Accessing a PC grants you benefits otherwise unavailable on mobile, such as powerful, specialized PC editors' stabilization fratures (like what DaVinci Resolve brings in) as well as more denoising options for cleaning up subtle grain with already encoded files. Your entire grading process consists of applying a single Color Space Transform (CST) node or a standard conversion LUT to bring the Log footage into Rec.709, followed by minor contrast and saturation tweaks.

You must be mindful of performance limitations: Direct Log is inherently more intensive than RAW capture because it requires on-the-fly encoding—the complex process of real-time encoding and compression of the vast sensor data into the final codec (HEVC/ProRes) must happen simultaneously with the capture itself. This severe real-time processing burden means that ProRes should be reserved for 24fps, and attempting 30fps begins to get harder for most devices. 60fps ProRes via Direct Log is currently unheard of due to the immense overhead of software and hardware acceleration. Furthermore, you must avoid the ProRes Proxy profile at all costs due to its deceptive labeling and inherently poor quality.

Your Stabilization strategy is layered and intentional: You prioritize stable technique during capture, recognizing that physical stability is always superior to software correction. If a smooth, cinematic shot is necessary, you rely on external tools (tripods, monopods, stabilizers). For slight movements or minor handheld jitters in mobile-edited clips, you utilize the readily available correction in the Google Photos app or basic in-editor stabilization for fast social media cuts. However, for the final polish of a polished project, you leverage the powerful and precise algorithms of PC editor stabilization found in your desktop editors, which provide professional-grade correction that better maintains image detail (you may even use Gyroflow!)

Therefore, you only tap into RAW to pull higher, more stable frame rates (like 60fps) if your device cannot reliably sustain them using Direct Log ProRes. Since RAW dumps data without the simultaneous encoding tax, you should use it purely as a temporary high-FPS capture method. You then leverage batch exporting in-app to quickly encode those high-FPS MCRAWs into editable ProRes files without performance bottlenecks, and discard the MCRAWs promptly as they are only a means to an end.

[6.6] -THE INTERBROLOMETER-

Your approach as an Interbrolometer is defined by a fascination with compressing time and capturing the unseen flow of the world. You are a specialist in Timelapses, Hyperlapses, and Astrophotography, viewing the smartphone not just as a video camera, but as a high-precision, portable sensor array capable of cinema-lapses.

Your philosophy is: "Time is relative, RAW is constant." You are not chasing real-time moments; you are orchestrating sequences where patience and planning yield spectacular, accelerated realities. This workflow is uniquely inclusive, as it breathes new life into older or 'crippled' devices—even phones with broken streams or strict 15fps caps become powerful tools in your hands, because your art form rarely demands high capture cadences.

Your only weapon should be RAW Video (MCRAW within RAW Video or Timelapse Modes), utilized specifically for its robust data integrity and maximum quality with data lossless compressed sizes during long and complex sequences. While encoded files like Direct Log can theoretically work, if you are committing to a capture that stretches over hours, you may as well go all-in on quality and stability. You avoid the instabilities and inherent compromises of Direct Log codecs, whose real-time encoding process creates a point of failure and introduces unwanted artifacts over extended durations. MCRAW is chosen because it is a simple data dump, offering maximum reliability for sustained, continuous capture.

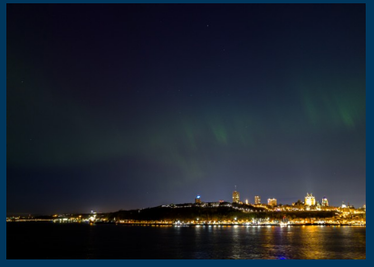
Your control over time can be exercised via two distinct methods, depending on the effect desired: You can utilize the dedicated Timelapse interval settings to intentionally introduce gaps between frames, which is the classical form of speed-acceleration for documenting slow-moving events like cloud movement or the setting sun. This mode also allows to designate shooting durations to allow the app to shut down once completed after hour desired timeframes, preserving resources.

Alternatively, for complex hyperlapses or when recording a subject that demands gapless, continuous motion (even at a slow speed, like 1 or 5fps), you rely on the app's standard RAW Video mode. Crucially, by selecting a low target frame rate—even as low as 1 frame per second—you are instructing the sensor to refresh and dump data far less often. This action drastically reduces the overall computational toll and heat generated by the device, profoundly increasing stability and allowing you to record for extended durations on hardware that would fail due to idling at 30fps.

This mode unlocks powerful capabilities for Time Montages and Astrophotography, particularly when paired with long exposures for night lapses. However, you must be aware of the "hardware lottery" regarding Camera2API limitations; while some devices may cap exposure length to a meager 1/4 second, others allow for 30 seconds or even 120 seconds, turning your phone into a deep-sky tool. A distinct advantage of MotionCam here is its Real-Time Viewfinder. Unlike other apps that cap the preview refresh rate to keep the image smooth (hiding the true exposure), MotionCam provides a "what you see is what you get" experience. Once you request a shutter speed longer than 0.5s and press the SET SHUTTER SPEED button that appears, the viewfinder engages the actual sensor integration time and presents it to the viewfinder. This effectively gives you "Night Vision," allowing you to scout compositions and hunt for angles in near-total darkness instantly, without the tedious "shoot, wait, review" cycle required by traditional cameras. Beware that this may appear as if the app froze at first, but it's merely trying to pull the longer exposure at which point, once completed, will appear on the viewfinder.

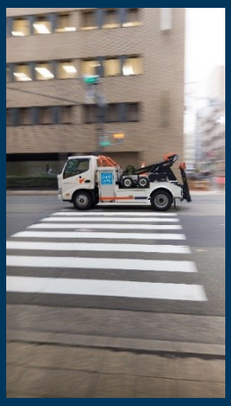
You understand that the power of this method lies in efficiency. Because you are capturing at such low frame rates, the resulting data rate is significantly lower than real-time video, allowing you to record for extended periods without overheating your device or instantly filling your storage. Once captured, you rely entirely on the Internal RAW Editor—your primary darkroom and stitching engine—to bring your vision to life. This powerful in-app tool acts as an end-to-end solution, handling the headache of stitching the sequence for you.

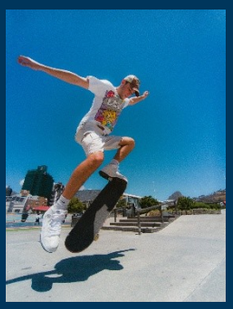
You can batch process exposure and white balance across the entire MCRAW sequence non-destructively (essential for day-to-night holy grails), apply denoising directly to the raw data, or even use frame stacking to reduce noise further. You can then export a fully compiled, pre-prepped video—even encoding it into a Log file for desktop grading—or simply extract high-quality single stills from the sequence. You are the master of efficiency, turning the limitations of mobile hardware into an artistic advantage, proving that you don't need the fastest processor to capture the passage of time in breathtaking detail.



[6.7] -THE BURSTA RHAWS-

Your approach as Bursta Rhaws is defined by the relentless pursuit of the perfect static image, hidden within a fraction of a second. You are not a videographer; you are a Burst Photographer who understands that the “Video” mode in MotionCam is actually just a high-speed engine for capturing DNG stills. You see the MCRAW container not as a video file to be watched, but as a digital binder carrying dozens or hundreds of raw photos. Your philosophy is: “Life goes fast, I shoot faster.”





Your primary tool will be Burst Mode, which can be used exclusively to capture unpredictable, instantaneous and momentary events like a lightning strike. The immense advantage of Burst Mode is that it is constantly buffering all the frame data for the immediate past (about a second or so) before you press the capture button. This means you can point your phone at the subject, wait for the event (e.g., the lightning flash), and only press the shoot button AFTER the event has occurred, retrieving the buffered frames to ensure the exact moment is caught. Basically, this is time travelling photography.

The Burst output consists of original RAW frames, meaning the full, flexible editing power of a typical RAW photo remains. You can easily pluck the perfect still from the reel directly off the app itself. It’s also important you select the ‘Save burst as video’ option so that all Burst sequences are saved as MCRAWs for later review!

Your secondary tool Is RAW Video Mode (MCRAW), but you can treat it strictly as a continuous, non-stop burst mode. You should use this mode instead when you need a longer duration of capture than the standard Burst Mode buffer allows. Even a modest frame rate like 15fps or 24fps is sufficient; although you can choose 60fps or the incredibly rare 120fps if available, do so with caution because not only is framerate speed not as important to the quality of the individual frame, but you will consume far more storage this way. This pursuit of high frame rates introduces a critical trade-off as well: higher framerates inherently necessitate faster shutter speeds, which severely hurts low-light performance by limiting the amount of light hitting the sensor per frame (eg. 60fps means you must use 1/60s as the slowest).

Your shooting is a careful dance between freezing the moment for maximum sharpness or allowing just the right amount of motion blur to enhance the image dynamically. Direct Log is completely useless for bursting and should be avoided completely for Photography purposes as it provides an encoded, compressed output (HEVC/ProRes) that destroys the raw data required for high-quality photography, and is noticeably inferior to JPEGs. By capturing MCRAW, you perform a simple data dump that preserves every pixel’s integrity, allowing you to extract specific DNGs later.

Your workflow must be highly strategic and tactical. You do not film to create a movie; you film as insurance. Since RAW capture generates massive file sizes, you should not record continuously. Instead, you should opt to capture short, intense sequences purely to secure the moment. For post-production, ignore the video export settings entirely (unless you actually think your burst sequence would make for a great video!).

Your goal is to enter the Internal RAW Player once you got the shot, scrub through your MCRAW containers frame-by-frame (much easier than it sounds), and extract specific DNGs to be processed in Lightroom or your preferred photo editor. The MCRAW file is effectively a binder of potential masterpieces, and once you have extracted your winning shots, you can discard the container promptly to reclaim storage space.

[6.8] -THE “SHUT UP, GEEKS…”-

Your approach as The "Shut Up, Geeks" is driven by a singular, powerful desire: high-end cinematic quality in your pocket, ready to go at any moment with zero post-production homework. This is a very difficult proposition but not impossible! You have no interest in color-grading nodes, file size debates, or the concept of a ‘log cur—’

–YEAH, YEAH, JUST TELL ME WHAT BUTTON TO PRESS!

Ok, fine..! You saw the natural, unprocessed look that MotionCam can deliver, and you want that perfect, delivery-ready file the moment you hit the stop button. This workflow is for the user who is done with the stock camera's fake, overly processed aesthetic (the oversharpening, aggressive noise reduction, and the general AI fakery and harsh tone mapping) but refuses to become a desktop editing hermit like us Maniacs.

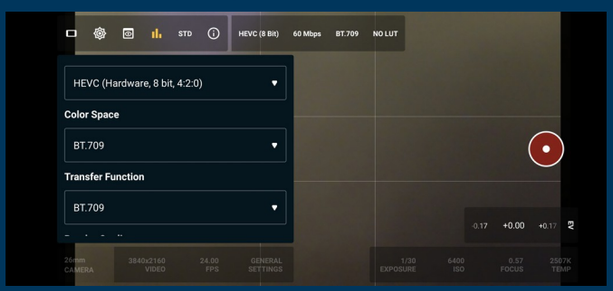
Your primary and only weapon will be Direct Log Video mode. You will treat this as the "Easy Mode" because it performs a critical act of digital alchemy: it bypasses the ugly processing of your phone's stock camera sensor while simultaneously handling the intense task of video RAW data encoding for you, delivering a standard, classic playable file immediately that is ready to go at once! Make sure you have a robust device, however, as you can do things fast, you can do them good, or you can do them cheap; you’re only getting to pick two of those, and it’s not going to be the cheap option in this case!

To ensure that file is not a dark, flat Log mess, go into the settings once and Bake-In a LUT that you wish to add for a finalized look. This is your secret to instant cinema. By applying a standard Rec.709 conversion LUT (or perhaps a custom look you downloaded) before shooting, you instruct the app to color-grade the video for you in real-time as it's being recorded. The file leaves the app looking exactly how you want it to look on YouTube, Instagram, or anywhere else, eliminating the single greatest barrier to entry for quality video.

Your setup configuration should be designed for permanent deployment. You will select HEVC as your codec, not for its technical perfection, but for its logistical convenience. It’s not just viewable right away, but it offers excellent color fidelity and rich data for a compressed file, and critically, it results in a small file size that your phone and every social media platform can handle without complaint. It can also be thrown readily into any editor for quick stitching or minor tweaking and cutting. You will set the following settings:

SDR VIDEO: 8-BIT HEVC + BT.709 TRANSFER FUNCTIONS + BT.709 COLOR SPACE

HDR VIDEO: 10-BIT HEVC + HLG TRANSFER FUNCTION + REC.2020 COLOR SPACE



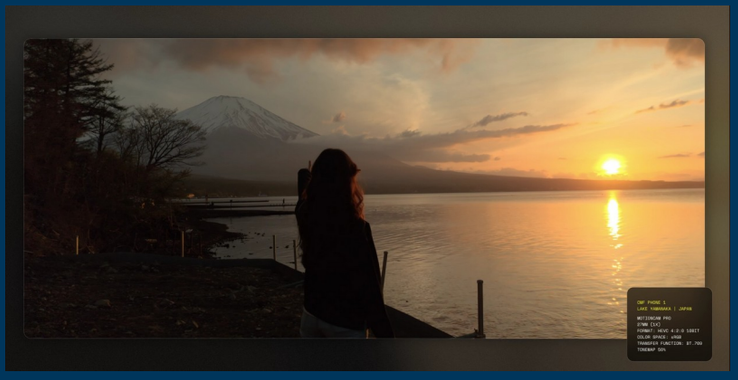
Also, note that Direct Log mode is very intensive, unlike all other video apps. Unless you have a vetted device known to work (like a top-tier flagship and even then not all can pull this off), 60fps is not to be used, as this app creates video by converting RAW into encoded videos real-time, instead of having the phone camera chip (ISP) handle it efficiently; hence how it achieves this level of quality. This, however, also means 60fps Direct Log is harder than any synthetic benchmark you’ve ever ran on your device!

If you use a LUT, simply ensure it matches your target look; typically, applying a LUT will automatically override the transfer function and color space to match. Once captured and rendered, it will become an 8-bit HEVC ready-to-view clip. And also, repeat after me: "10-bit HDR should only be graded with advanced editors". Stick to SDR; the majority of online content works with this, so do NOT try to use HDR pipelines unless you become more proficient. Failure to adhere to this will result in your footage looking flat and lifeless plus becoming very difficult to grade on standard screens.

You are to strictly avoid the massive file sizes of ProRes, which only introduce technical clutter you are trying to escape as well as need dedicated PC editors to handle.

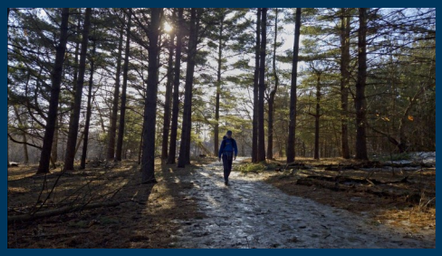
Once your look is baked in, your final adjustment is purely visual: you can use options such as the Gain Slider in the render settings to boost the perceived brightness of the image, amongst many other things, and even the transparency of the LUT itself. Don't be shy about pushing the digital Gain to +1.0 or higher. While technical users call this "Exposing to the Right" (ETTR), you simply call it "making the image look brighter when needed," ensuring you maximize visual quality without ever needing external editing software.

Finally, stabilization is handled through pure convenience. Because MotionCam bypasses the phone's stabilization processing in favor of a natural feed, you will need a steady hand. You can also invest in a cheap, simple gimbal or monopod for serious shots. For everyday jitters, your final step is to leverage the excellent, free, and simple stabilization feature built right into the Google Photos app. Your workflow is the height of simplicity: you shoot in Direct Log with your look pre-applied, perform a quick, single-tap mobile stabilization if absolutely necessary, and upload. You can stand to harness some of the cinematic quality that stands head-and-shoulders above stock video, and you do it without ever opening an editor or reading a manual. Just don’t forget there’s more to ‘cinematic’ than just the look itself!



[6.9] -THE VLOGMEISTER-

Unlike the "Shut Up, Geeks" folk who treat their phone like a one-click digital point-and-shoot, you are The Vlogmeister, a traveling storyteller. You understand that your video isn't finished until the story is told—a process that requires planning, mobile editing, and dealing with a few extra steps. You possess an understanding that is slightly more advanced than the instant-cinema crowd, but you are not chained to a desktop like the RAW purists.



Your approach is defined by a need to document the journey without letting the gear get in the way of the experience. Your philosophy is: "The best camera is the one you can carry for a while." Your primary weapon of choice is Direct Log Video, and you wield it with a focus on universal compatibility and mobile-first color grading. You accept that professional mobile editors handle Log footage well, so instead of baking in a final look (like the "Anti-Geeks" do), you deliberately select a widely compatible Log curve, such as Samsung Log or Apple Log as your base output. This decision is crucial because these formats are industry standards; most mobile NLEs (like CapCut, VN, or LumaFusion) are pre-loaded with or have easily accessible conversion LUTs for these specific Log types, making your color grading simple and predictable. You are essentially using the app to output a log file that is intended for intermediate difficulty editing, but nothing too excessive.

Your configuration prioritizes the sweet spot between decent quality and file size efficiency. You utilize HEVC (H.265) encoding because it keeps your file sizes small enough to store hours of travel footage on your phone, yet retains enough quality to look professional on YouTube. While you might dabble in 10-bit color for high-contrast scenes, you are smart enough to primarily use 8-bit HEVC when shooting standard daylight vlogs that don’t require grading to ensure maximum compatibility and speed with your mobile editing apps. This can potentially speed up your editing process by cutting down on the amount of grading required.

Since Direct Log is a continuous conversion of RAW data (an incredibly taxing process), you know that heat is your greatest enemy, especially when traveling in hot climates. To guarantee stability and long takes, you should definitely consider external cooling solutions, like a specialized Peltier cooler (a phone cooling fan) to keep your phone's processor from throttling and ruining a crucial sequence, even if only used to briefly cool the device in between shots.

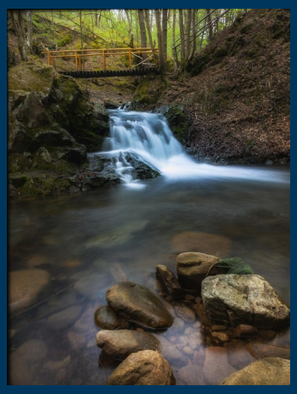
Because audio is half the video in vlogging, you should be acutely aware of MotionCam's limitations here; know that Bluetooth microphones are not currently supported, so you should always pack a compact USB-C microphone or a wired lavalier to ensure your voice is crisp. Also monitor the Audio Meter besides the histogram (can be toggled from the General Settings) to ensure you aren't clipping noises when things get loud!

Your workflow should be a streamlined cycle of Shoot, Stitch, and Grade on the Go. Therefore, shoot in Direct Log, utilizing your preferred universal Log curve, perhaps stabilizing handheld shots with a simple grip or utilizing the Google Photos stabilization tool for jerky walking segments. You then import these flat Log clips into your mobile NLE, where your first step is always to apply the relevant conversion LUT (e.g., Apple Log to Rec.709) before arranging your timeline.

Stick to standard frame rates like 24fps or 30fps and focus purely on the story structure. You are the bridge between the casual user and the pro, proving you can create comprehensive, high-quality travel films without needing a laptop or rig in your backpack.

[6.10] -THE LIGHT HUNTER-

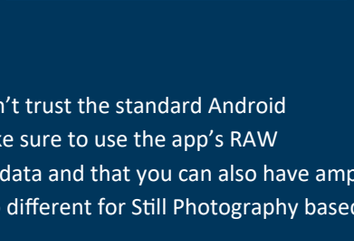
Your approach as a Light Hunter is defined by a rejection of the modern smartphone's "computational photography" overreach. You are not here for video; you are here to reclaim the still image from the clutches of aggressive AI, over-sharpening, and oil-painting noise reduction. Your philosophy is: "The perfect frame isn't generated; it's captured." You treat the app as a dedicated mirrorless camera that happens to be attached to a phone.



The core advantage of this app is that its manual controls don't lie or interfere. Unlike stock camera apps that secretly override your manual settings to brighten dark areas or attempt to 'fix mistakes' which can often be actually purpose-serving decisions you've made for your capture, MotionCam respects your artistic intent. If you want shadows to be shadows, highlights to be bright, and even if you want noise to be noise and show the purity and innoc of the sensor readout grain and it's sometimes limited dynamic range's contrast, this app lets it happen. This results in an unadulterated base file that will capture exactly what you saw and intended to, either in the final JPEG you can get from the app or with the editing latitude of an unprocessed raw, for better or worse.

Your primary weapons will be the Photo and Night Modes, but you should wield them with specific intent. Rely on the Photo Mode for standard conditions where you want a "pure" look without the stock camera's heavy processing. You should utilize the Night Mode not just for darkness and lowlight environments, but for any somewhat static scene where you want to leverage frame stacking to reduce noise and increase clarity naturally. You should be willing to dive into the Photo Settings (found in the Photo sub-menu) to dial in the exact contrast, saturation, and sharpening levels you want applied to your JPEGs for processing the app's RAW linear data which may otherwise not sit well on the 8-bit JPEG container if left fully untreated, ensuring they match your unique style.

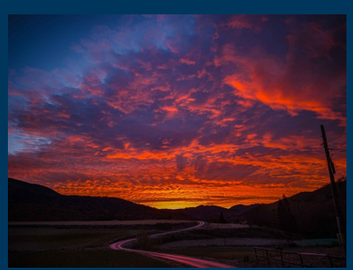
It's important to note that MotionCam's Photo Modes are also coming from the RAW stream of the camera, meaning unlike other photo apps, this completely continues to bypass the ISP, producing shots that are unlike other Photo or Camera apps which simply request images from the 'JPEG Stream' that is extended from the ISP via the Camera2API. MotionCam will essentially handle the processing otherwise dealt with by the ISP.

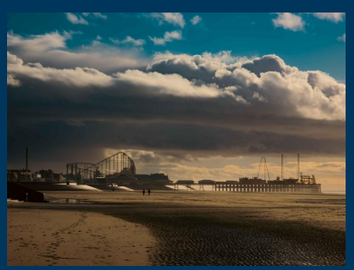


For action or unpredictable lighting, you should deploy Burst Mode as a safety net. Instead of hoping you timed the shutter perfectly, you capture a rapid burst of RAW frames, allowing you to scrub through the sequence later and extract the single, absolute perfect moment using the "Edit Photo" tools. This workflow effectively gives you a "time machine" for your stills. More importantly however, you can then use the reel burst to apply denoising with the right amount of frames later rather than hoping the app gets it right in the first shot; insurance in other words.

Crucially, you should leverage the Direct Preview (the eye icon) modes to see exactly what the sensor is seeing—don't trust the standard Android viewfinder because it lies to you with fake brightness and processing that the app doesn't use whatsoever. Also make sure to use the app's RAW Histogram to expose perfectly to the right (ETTR) without clipping highlights, ensuring your RAW files are thick with data and that you can also have ample sensor latitude if you decide to capture as a JPEG instead. The best noise reduction is no noise-reduction, and it's no different for Still Photography based shooting.

Should you decide to use the app to harness strictly RAW photography, your post-processing will happen in two stages: First, you can use the in-app Edit Photo menu to apply granular Denoising, Frame Stacking, or even pull the JPEG itself straight from here, lightly tweaked and ready to go directly from export processing to your bursts or raw captures. If the image needs more work, you can take the DNG (RAW) file into Lightroom, Snapseed, Photomate R3, or any advanced PC processor knowing you have a clean, unadulterated base file that won't fall apart when you push the shadows.





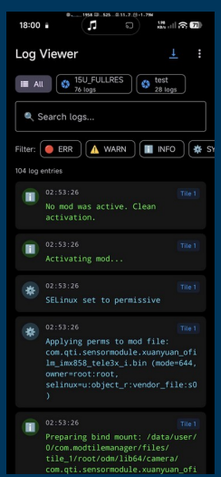
[6.11] -THE MAD PROGRAMMER-

The type of user that dares to take on The Mad Programmer's mantra is not about capturing great art; they're about tearing down the walls of vendor-imposed limitations. You represent the vital, highly technical segment of the MotionCam community that thrives in the underground of custom ROMs, kernel tweaks, and root access. You see the capabilities of MotionCam and are deeply intrigued by its unique ability to accept and interoperate with almost any mod thrown at it, unlike the majority of camera apps that require standard, rigid ISP-driven data to work.

Your philosophy is: "There's no restriction imposed by an OEM that can't be removed with an unlockable bootloader, an all-nighter, and a Red Bull in hand." You are amongst those who push the envelope, testing the absolute limits of your device's advertised versus actual capability, sharing precise technical data, and often creating the very mods that benefit the rest of the user base who dare to root.

The essence of your workflow is unlocking latent sensor technology and improving crippled efficiency. You are the user who has rooted their phone, knowing the risk to warranty is a fair trade for the reward. You see MotionCam as a vessel, and your technical goals are to improve device efficiency and performance, often crippled due to poor optimizations in stock firmware. You achieve this by leveraging root access for key functions. First, you utilize custom firmware tweaks to activate hidden features such as Dual Conversion Gain (DCG/Samsung's Smart-ISO Pro). This dramatically increases the sensor's dynamic range, providing cleaner shadows and superior highlight detail—a technical victory over the stock sensor driver.

You may even apply Magisk or kernel mods to increase the thermal throttling thresholds or change the device's thermal profile, letting it push harder and hotter before it taps out. This allows for significantly longer recording times, pushing the device far past its stock limitations, transforming a 30-second clip capacity into a stable, multi-minute take.



Your primary tool within MotionCam is the RAW stream itself, harnessing the MCRAW format as well as Direct Log modes for convenience of capture. You use the app not because it's convenient, but because it captures the pure, unaltered sensor data (often at 12-bit or 14-bit depth, which you may have also unlocked). You are the one performing the rigorous, comparative testing: capturing a shot, analyzing the waveform and histogram plots, and comparing the noise floor and dynamic range of a stock capture versus a modded, DCG-enabled MCRAW. You are less concerned with color grading and more concerned with the mathematical purity of the data captured, tearing open the possibilities.

You don't just use the app; you are a key part of its hardcore enthusiast ecosystem. You are the one compiling custom kernels, helping to debug driver issues on a specific chipset, and discussing performance metrics in online groups. While others discuss framing, you are discussing CPU scheduler behavior, memory bus bandwidth, and Android's Camera2 API implementation on various devices. You are the high-stakes test pilot for MotionCam, and without your technical curiosity and dedication to rooting out hidden functionality, the app would be in a completely different shape.

Realistically speaking, this type of user is a rare breed. If you fancy yourself one of these and just love to develop and tweak for the sake of it, you are absolutely welcome within the Discord group. This type of chatter technically can blur the line of "it is/isn't MotionCam," however, the curiosity to push and help in the advanced debugging of the app, particularly within exotic brands, is absolutely essential for pushing the limits.

While the app will often run totally fine with root, absolute maniacs like some of us love pushing it beyond, so know that if you crave more and are willing to go through the trials and tribulations that come with running mods and their creation, this community and its members are THE PLACE to be. Our ranks spread from Discord to Reddit and numerous Telegram groups. I wasn't kidding in the manual intro when I said the expertise available with many of our members is second to none!

Want to join the cause? Jump on the Discord and explore the appropriate channels! We shall show you the way into the dark arts! The rabbit hole goes deep, and you can help us with boring it!

The MotionCam Maniac gang welcomes you!

[6.12] -“DOCTOR ANDROIDSTEIN”-

If The Mad Programmer is the Brain Doctor of the community, Dr. Androidstein is the mad bio-engineer, stitching together parts that were never meant to coexist to create a monster of imaging power. You represent the rarest and most extreme segment of the user base: the hardware modders. Your philosophy is that a smartphone is simply a compact sensor module waiting to be liberated from its restrictive glass and aluminum chassis. You look at a sleek, sealed flagship device and see a challenge, often meeting it with a Dremel, a 3D printer, and a soldering iron.

Your approach involves physical transmutation. You are not satisfied with clip-on lenses; you are the user designing custom CNC-machined backplates to mount full-frame cinema glass or adapting industrial CS-mount lenses directly to the naked sensor. You tackle the heat issues inherent in high-data RAW recording not just with software tweaks, but by physically modifying the device’s thermal path, attaching aggressive Peltier cooling solutions, or even drilling into the casing to create direct heat dissipation channels.

To you, a "voided warranty" is just proof that you have taken ownership of your device. MotionCam is one of, if not the only software capable of keeping up with your physical inventions. Where stock apps crash, fail to focus, or outright refuse to boot when confronted with modified systems, an external depth-of-field adapter or a massive ND filter array, MotionCam provides the manual overrides necessary to make these optical frankensteins sing. You rely on the app’s ability to handle raw sensor readouts without trying to "correct" the optical characteristics of your custom glass, allowing you to capture unique, cinematic textures that no computational algorithm could ever replicate and correct things like Infrared readouts.

This path is not for the faint of heart, but you are not alone in your laboratory! You are part of a lineage of innovators who hang within our ranks—legends like Mod & Shoot, Cosimo, and other known hardware modding creators who have paved the way. These folks have turned standard phones into cinema rigs, and they are active members of our community, proving that with enough ingenuity and adhesive, anything is possible!

There’s also an established and extensive documentation regarding prior mods, teardowns, and optical experiments. The community is incredibly eager to help you build your own rig. If you have an itch to saw through a camera bump to mount an actual camera lens, or if you want to see just how far you can push the physics of mobile imaging, we are waiting for you! Join the Discord and find your fellow mad scientists in the hardware modding channels.

Whether you are looking for 3D print files for a cooling mount or advice on sensor focal flange distances, there’s a little something for all the tinkers and makers. Don't build in isolation—bring your creation to the lab, and let’s see what kind of monster we can bring to life together!

[6.13] -THE SPECS JUNKIE-

The Specs Junkie is the archetype defined by an unyielding pursuit of numerical supremacy and the elusive "perfect device." Their existence is a cycle of anticipation and disappointment, driven by the obsessive need to secure the latest and greatest hardware with the highest theoretical performance. Their philosophy is not about making films, but about proving a phone's worth on a spreadsheet. They live in the realm of API readouts, Dual Conversion Gain (DCG) sensor promises, and maximum bitrates, constantly asking: "How close can I get to capturing 8K 120 FPS on this device?" and attempting all the app's maxed out settings.

Their days are spent not on the field, but buried deep within specification sheets, tech news sites, and compatibility forums. This user relentlessly runs benchmark after benchmark, comparing endless photo samples and raw video files between devices, convinced that their next purchase holds the key to unlocking true filmmaking potential.

They are constantly moving from one phone to the next, convinced that one model offers the magical combination of full MotionCam compatibility, the largest sensor, and the fastest chipset, failing to realize the creative nuance available in the hardware they already own.

This hyper-focus on technical perfection, rather than creative output, is their fatal flaw however. They are often brilliant testers and valuable members of the community for pushing limits, yet they can sometimes become perpetually stuck in a state of pre-production analysis. They fall into the "rabbit hole sinkhole" of specs, a never-ending chase where every potential improvement becomes an over-emphasized necessity, preventing them from ever exiting the loop to begin the act of creation.

Despite the pitfalls, the Specs Junkie's passion for obtaining the highest quality is inherently valuable. Their drive to seek out cutting-edge performance pushes the entire community to understand the limits of new hardware. However, if you do fall into this category, you must ensure you realize that more is not always best, and that maximizing numerical specifications is a means to an end, not the end itself.

The key to progression lies in the realization that creativity is vital to the specs; research less and shoot more! The Specs Junkie has already done the difficult work of mastering grasping potential; the next step is to move up the tiers of users by transforming that knowledge into visual storytelling. Once they find a device that delivers a reliable stream of high-quality data, the real challenge begins: using that data to create!

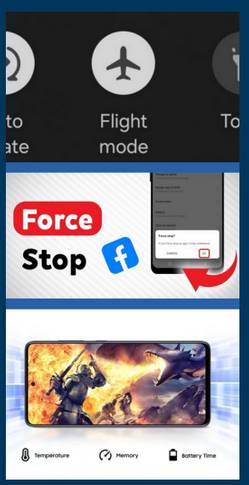
[7] -Optimizing for Max Performance-

If you've been paying attention, you've likely realized that MotionCam Pro pushes your mobile hardware harder than almost any other mobile application on the market. I dare say it actually makes for an excellent benchmark tool of its own..!

Unlike stock camera apps or even the majority of other video apps that rely on the dedicated, efficient ISP pipelines, MotionCam bypasses these safety rails to access the raw sensor data directly. This demands sustained, peak performance from your CPU, RAM, and Storage Controller.

As community testing has proven (even on older devices like the venerable OnePlus 8 Pro) the hardware is rarely the bottleneck; rather, it is the software bloat and aggressive thermal profiles that hold you back. To achieve stable, high-frame-rate recording, you must optimize your environment to assist the processor. Let's explore some of the available things that are within your control assuming non-root options…

[7.1] System-Level Optimization

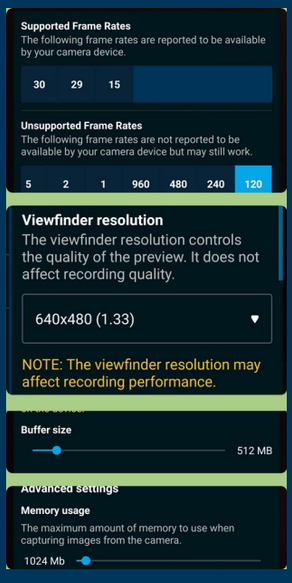


Before you even launch the app, you should (ideally) prepare the system to get out of the way. The single most effective, yet often overlooked, trick is to enable Airplane Mode. This is particularly crucial for Samsung devices, where background radio spikes searching for 5G or Wi-Fi signals can cause micro-stutters and CPU interrupts that ruin a recording. By enabling airplane mode, you eliminate this significant bottleneck and allow the device to dedicate all resources to performance.

You should also treat your filming session like you would a gaming session; if your device has a "Game Mode" or "High-Performance Mode" optimizing tool, ensure you add MotionCam to its whitelist. This tends to instruct the OS to allocate more resources to the app and to treat it seriously, often delay/extending thermal throttling thresholds.

Finally, kill unnecessary background processes. Social media apps are notorious for draining resources, so simply swiping them away in the Recent Apps menu can significantly augment available performance (bonus points for force stopping them..!)

[7.2] In-App Power Management



Viewfinder Resolution: The preview screen constantly renders a high-resolution image which drains CPU/GPU power. In the Camera & Viewfinder Settings, manually decrease the preview resolution to a lower setting like 480p instead of Auto. In order to reduce strain. You can then use the STD/MAX toggle on the main UI to quickly switch to a high-res preview only when you need to check critical focus, them return to a more economical option like the aforementioned 480p (just a guideline example).

The "FPS Slot" Trick: You can force the system to prioritize the camera sensor by selecting a high-speed stream (e.g., 90fps or 120fps) in the settings – even if you only intend to shoot at 60fps. Even if the framerate doesn't actually work, this forces the Android hardware layer to allocate significantly more bandwidth and priority to the sensor as it believes a higher demand is about to occur. ***Note***: You may need to manually override your exposure, as the Auto-Exposure driver may use the higher framerate as its shutter speed baseline (it will often assume the framerate selected worked).

RAM Buffer: Increasing the RAM Buffer allocation gives the app more breathing room to store raw data before handling it. This can improve performance for both RAW Video and Direct Log Video Modes! However, be careful! Allocating too much memory can cause the OS to think the app has gone "rogue" and kill it entirely!

[7.3] Direct Log - Quantity vs Quality

When shooting Direct Log video, you are asking the phone to perform real-time debayering and encoding, while also trying to hold incoming RAW/Bayer Data not yet processed in a RAM holding queue, which is an immense computational workload. To reiterate once again (I haven't done so enough already): ***Direct Log mode is substantially more intensive than standard RAW video capture.***

An extremely common mistake is to lower settings to "save performance," but this often backfires. For example, using 10-bit HEVC in something like HLG with Rec.2020 is often easier for the processor to run than 8-bit HEVC in BT.709 + Rec.709. You're probably confused, but think of it this way: shooting in 8-bit forces the CPU to perform extra compression calculations to squash the tonal range, whereas 10-bit allows for a more direct data path with less actual compression to occur! Too much compression? That take s power! Too little compression? Well that also takes power! You must avoid essentially running a setup that will assault your resources. Settings like Real-time Noise Reduction, Highlight Reconstruction, Sharpening and Detail, and software based codecs will all heavily take a toll on your device processing demands.

If your device is struggling; these require massive computational power per frame. Stick to a bitrate "sweet spot" of 75-120 Mbps for HEVC and run with 10-bit HEVC if you're familiar with how to handle it. Going higher overloads the storage controller and generates heat, while going lower forces the CPU to work overtime on complex compression algorithms.

Finally, **avoid the "poisoned chalice" of ProRes Proxy**; while it offers smooth performance, the quality penalty is severe and rarely usable for serious work. If all of the above codec options still fail, recording to RAW video remains the least intense option to capture simply by nature of having to only record the incoming data rather than processing. Essentially, **you can record RAW with the intent to apply high end settings after the fact if you can't in Direct Log!** (eg. 60fps, Highlight Reconstruction, Noise Reduction).

[7.4] Sensor Readout, Light, and Storage

Optimization also extends to the physical world. One of the biggest hidden performance killers is a mismatch in shutter speed. If you shoot at 60fps, your phone may actually be running at a jagged 59.91fps. In an effort to gather as much light as possible, some users will manually lock the shutter/exposure length to exactly 1/60s, which can create micro-drops every second; boosting the shutter slightly to 1/61s or 1/64s (just as examples, depending on device these may change) ensures the sensor clears in time for the next frame.

Furthermore, remember that MCRAW uses lossless compression, meaning that light equals performance. A noisy, underexposed image is mathematically complex and harder to compress, increasing CPU load. Adding light cleans up the signal, making the compression operate more efficiently and actually reducing the strain on your phone! If you needed a good reason to expose well, this is yet another one! This principle applies for both RAW Video and Direct Log Video modes!

If you have significant amounts of noise, Direct Log will see severe impacts since the encoder will be potentially suffocated trying to intelligently 'predict' otherwise unpredictable noise; an absolute nightmare, which actually results in the encoder having to basically resort to a compression based noise reduction! And yes, you guessed it, this will take more resources (plus produce otherwise inferior quality).

Finally, you must manage heat and storage. Heat is the primary enemy of performance; once your device battery hits roughly 40°C, the kernel will often begin to throttle the CPU. This will vary per device of course, but the OEM decides strictly at temperatures they'll clamp back. Devices like Google Pixels and Samsungs are extremely Diva-like in their behaviour. This means that once the heat begins to saturate your device, you can begin to expect potentially lots of dropped frames at this time.

Remove your phone case or use one that's not overly thick to let the back glass and frame metal dissipate heat, lower your screen brightness, and consider an active Peltier cooler for professional workflows where you plan to shoot for very extended amounts of time.

You can also store your footage on an external SSD to try and produce less internal heat, however internal UFS storage will tend to be the most 'battery efficient' so take that into consideration if you didn't bring a power bank or attachment hub. External USB-C drives can be more reliable for sustained sessions, however do not ignore your device temperature either way.

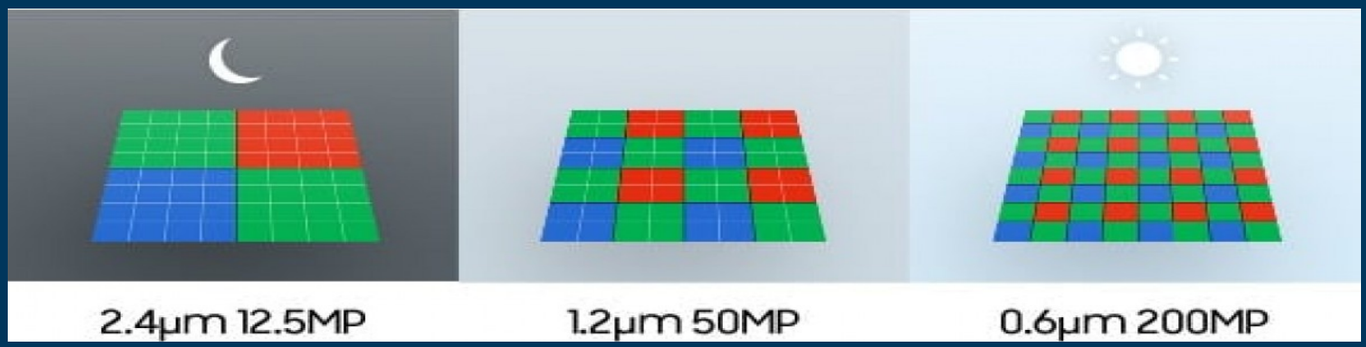
Lastly, **NEVER let your storage capacity drop below 20%**, as write speeds on internal flash storage degrade rapidly when the drive is nearly full. This issue is particularly notable on low storage models (eg. 128GB) which can easily run into this limit just by having everyday apps and content loaded on the phone. Rely exclusively on external storage recording if you're in this situation as internal storage is a recipe for disaster as far as RAW video will – this is one of the cases where Direct Log Video mode may yield you better endurance, the inverse of what you can otherwise expect!

On a **final note about External SSDs**; not all of them are made equal! Always ensure you do proper research on any models before you purchase them. The reason we state this is because some SSDs will advertise fast speeds, but the critical component is they're not always concerned about sustained endurance.

Case in point, a typical run-of-the-mill consumer SSD like popular SanDisk models may look good on paper, but when put through their paces and used for external recording which requires substantial endurance, they completely fall apart and do not reliably work for anything like 4K24fps RAW video.

Aim for properly vetted storage solutions, popular ones such as the Samsung T7 and T9 series, as well as Samsung EVO series models (you can use a housing solution for them). Aim for anything with a **sustained performance of 350MB/s to comfortably record in 4K30fps RAW video.** Failure to heed this warning will result in a miserable external recording experience; you've been warned!

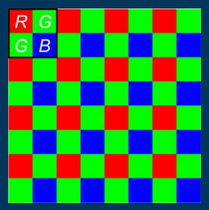
[8] -A Conversation & Crash Course on 48-200MP Binned Sensors-



If you've been shopping for a flagship phone lately, you have undoubtedly seen the "48MP," "50MP," or the massive "200MP" numbers plastered all over the marketing materials. It sounds like revolutionary; like a professional medium-format camera in your pocket. However, as soon as you open MotionCam, you'll notice the app usually defaults to something like 12MP or 12.5MP. This isn't because the app is "broken", "downscaling", "cropping" or even "line-skipping" your image; it's because those massive numbers are a bit of a marketing sleight of hand with a different intended purpose than actually producing ultra high usable resolution. Hear me out…

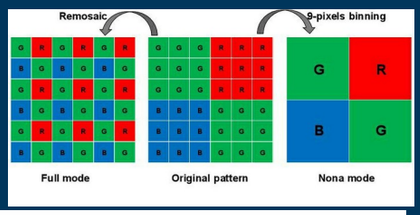
To get the most out of your device, you need to understand the difference between the physical pixels on the sensor and the "Super Pixels" the phone was actually designed to use.

[8.1] Strength in Numbers: Understanding Binning



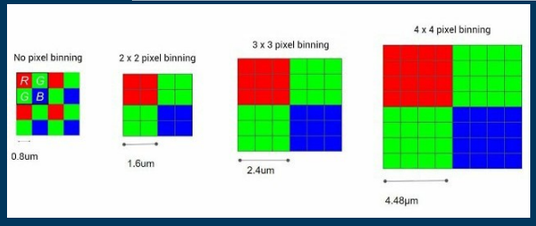
Let's go back a little bit. Here's how a Pixel array would generally look, the classical 'Bayer' array on which the sensor gets the equivalent of 3D glasses that apply different colors to different areas to enhance the color sensitivity of the sensor and give it the ability to create color rather than just black or white.

This 'filter' is what is generally expected, known as RGGB pattern (Red, Green, Green, Blue; the order in which pixels show). The pattern could be in different order but the principle remains, 1 red, 2 green, 1 blue)



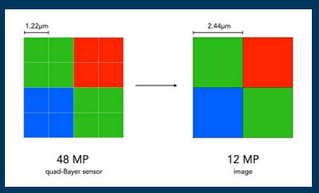
To understand why your 200MP sensor usually acts like a 12MP one, I want you to think about your sensor as a human hand and the individual pixels as fingers. If you use your entire hand together, you have immense, concentrated strength. You can grip a heavy suitcase or lift a massive weight because all your muscles and fingers are working in unison. You can't grab a lot of things, but what you can hold can gets full force! This is like Pixel Binning!

The sensor Is actually designed with the pixels having a color filter that relies on them having a full grouping with the same color filter in a 2x2 grid (on a 48MP sensor) or a 3x3 grid (on a 108MP sensor), or even 4x4 (200MP sensors) of tiny pixels together. Via this method, the sensor allows them to work as a single "Super Pixel." This "hand" is at its strongest when it's working together. We call 48/50/64MP sensors Quad Bayer, as in 4 pixel clusters. 108MP sensors are called Nona Bayer, as in 9 pixel clusters, and 200MP being called tetra2 or Quad Quad Bayer using 16 pixel clusters!! Beware! Companies use different marketing names for the same (Samsung Tetra Cell vs Sony Quad Bayer Coding, etc).



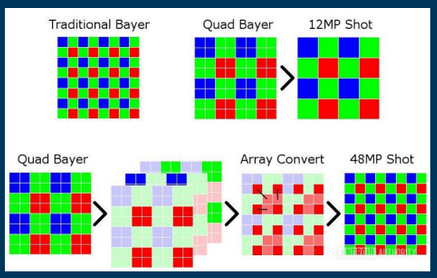
Now, imagine trying to carry those same heavy suitcases using only your individual fingers. You can technically hold more individual items (more "detail"), but your strength is gone. Your fingers will shake, struggle, and eventually give out. This is what happens when you "unbin" the sensor, or otherwise run it in its full resolution mode allowing you to shoot at the full 48/50/200MP resolution. It's only a viable strategy for well lit, ideal scenes otherwise in near perfect conditions. But, for anything challenging or demanding (low light in this case), you'll take a hit on the actual practical resolution of your sensor, since the pixels are meant to work together as a unit. Confused? Allow me to explain why…

[8.2] Remosaicing



‘So if those 48-200 million pixels are physically trapped under only 12 million color filters, how does the phone actually give you a “48-200MP” color photo’ I hear you ask? This is where a process called Remosaicing comes in. Think of it as a giant, math-heavy game of “Pretend.”

To produce that high-resolution file, the phone’s sensor has to take those grouped pixels and force them to act as if they each had their own individual color filters and were part of a ‘normal’ Bayer array. The hardware mathematically “rearranges” itself, attempting to simulate a standard Bayer pattern where every pixel has its own unique color data. It is essentially a software reconstruction of a reality that doesn’t exist on the silicon.



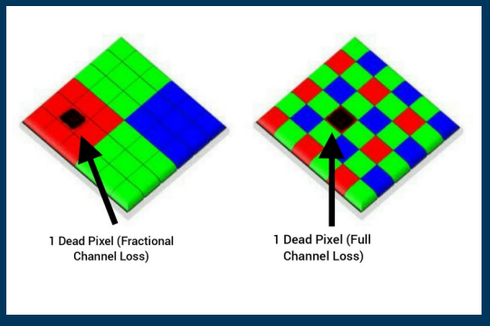
The problem is that you cannot cheat physics. Because the color information is actually being shared across a group of four or nine pixels, the software has to “guess” or interpolate what the specific color should be at each microscopic point. This “faking it” comes at a heavy cost to color fidelity. Because the sensor is guessing, the colors often lack the depth, accuracy, or otherwise the “fullness” of a binned image.

When you combine this with the fact you’re also having to now use individually smaller pixels, you’ve now opened the evil trifecta: Small Pixels on a Small sensor on a Small Lens, with a touch of reduced color fidelity to top it off.

You’ll also start to see the “math” failing in the form of artifacts – strange “zipper” patterns on sharp edges or “mushy” textures where the software couldn’t quite figure out the fine detail, or even blobs of Chroma artifacts that may show due to the inherent issue of binning 4-16 pixels into 1 or even separating/unbinning them. **See the right comparison as an example!**

You aren’t getting 200 million points of true color; you’re getting a very high-resolution mathematical estimate as close as it can get. While the pixels are getting 48-200MP worth of data, the actual useful data is in reality nowhere near that.

Now, this isn’t to say this technology is entirely useless, but rather that it’s not without its flaws. We’ll explore the benefits shortly, but ideally, you can obtain higher resolution in optimal conditions since the unbinned mode can then actually work decently in bright scenes as stated. Just don’t think you’re missing out on something game-changing if you can’t access it.



Another reason OEMs also tend to prefer these sensor arrangements on modern devices is because they also tend to tolerate manufacturing issues better. This is to say, if you make a 108MP sensor and you have 1 dead pixel on a color channel cluster of 9 like-colored pixels, odds are the Binning averaging process can all but conceal it.

This effectively affords OEMs better manufacturing error tolerances when creating sensors. 1 dead or defective pixel on a cluster becomes far less noticeable than if the whole color channel failed as it otherwise would on a conventional Bayer pattern!

[8.3] Signal-to-Noise Ratio: It's Not "More Light," It's "Cleaner Signal"

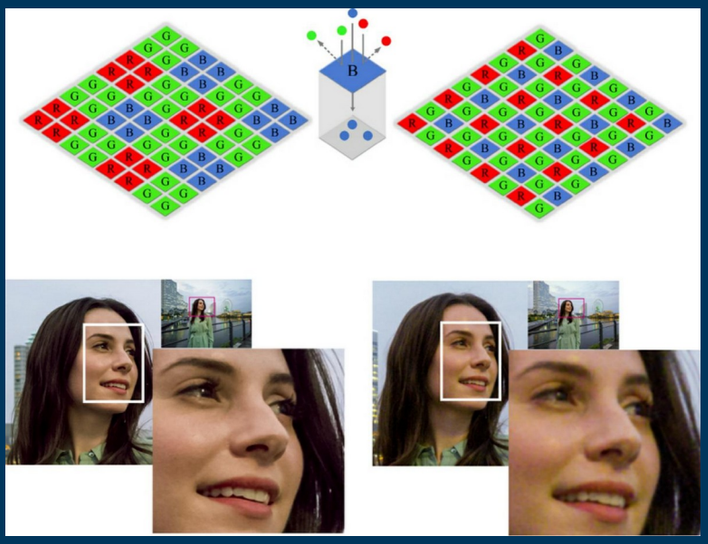
This deception leads to a very common misconception: that binning "captures more light." Let's be clear: the sensor is a fixed piece of silicon. Whether you are in 12MP binned mode or 200MP unbinned mode, the physical sensor area is identical; it is receiving the exact same amount of photons from your lens. What actually changes is the Signal-to-Noise Ratio (SNR)!

When you bin those pixels together, the hardware is able to average out the random Luma and Chrome Noise (grain and colorful blotches that otherwise appear randomly and everywhere when shooting in darker scenes) from the data.

By combining the data from multiple pixel points working together, the "signal" becomes much more robust and reliable, not unlike stacking multiple images together can improve noise patterns. Going back to the hand example, an individual finger itself is unimpressive, but when you use the whole 5 fingers together on a focused task, you can now tackle heavier items otherwise out of the reach of a single finger's strength.

This improved noise performance from binning pixels that would otherwise produce a lot of noise creates the impression of more light; because the image is so much cleaner and has significantly more "meat" for you to push in a color grade or boost the brightness and shadows. Inversely, when you UNBIN that sensor in anything less than ideal enviros full of bright sunlight or studio lighting, the noise can easily become so overwhelming that it actually contaminates the information.

You end up with a high-resolution file that actually looks worse than the 12MP version because the noise has effectively "eaten" all the fine details, leaving you with a grainy mess that requires excessive amounts of work to fix, if even possible at all. This is why I often recommend unbinning only when you have an abundance of light; otherwise, the "resolution" you're gaining is just an increase in the resolution of your sensor noise!



Do you really want to watch your Chroma noise enhanced in 8K/16K..?

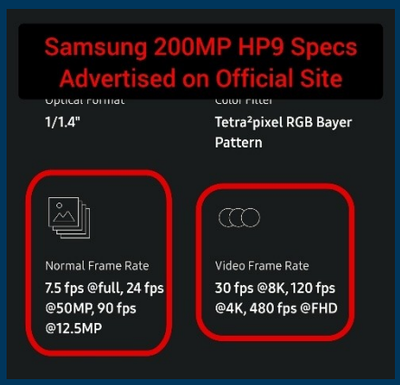
[8.4] The 8K Utility and the Frame Rate Wall

Now, you might ask: “If 12MP is so good, why do I have 48MP or 50MP options?” This is where we talk about 8K. To record 8K video, you need a minimum of about 33MP. Therefore, the 12MP binned mode physically cannot produce 8K. If you want that extra resolution, you have to unbin the sensor. This can be a useful tool when you have studio-level lighting and need that extra “crop-ability” for your project.

However, there is a massive catch on top of the previously mentioned concerns…

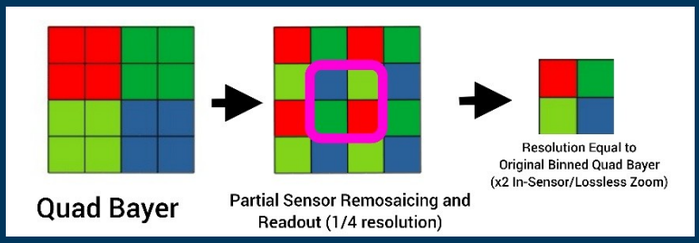
While the binned 12MP modes can often fly at 60fps or higher, the unbinned high-resolution modes are almost always capped at lower framerates like 24-30fps due to sensor readout speeds limits and remosaicing processing limitations, regardless of lighting.

When you jump up to the 200MP monsters, the situation becomes even more extreme. If you look at the actual spec sheets for these sensors, they are often only rated to run at a staggering 7 frames per second at their full, unbinned resolution. It is currently impossible for the sensor’s “brain” to clock fast enough to turn a 200MP Nona binned readout into an unbinned standard video stream (would basically be 16K video). This is why 8K video (even on the most expensive phones) often feels a bit more “choppy” and prone to stuttering than standard 4K, as well as why it produces inferior low light footage.



[8.5] In-Sensor Zoom (ISZ): The Double-Edged Perk

Despite the limitations, these high-resolution sensors do offer a genuine cool benefit: In-Sensor Zoom (ISZ). Unlike old-school digital zoom, which just crops into the image and stretches it out (creating a pixelated disaster), ISZ is much smarter. If you have a binned sensor and you want to zoom in, the sensor can simply “ignore” the outer edges of the frame and record only the middle 12MP region in the unbinned modes.

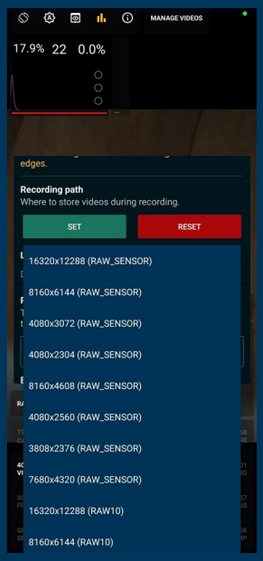


This provides a massive practical benefit. Because the sensor is only reading a small “crop” area in the center, the readout speed improves significantly. It doesn’t have to process the data from the entire 200MP surface; it only has to move the data from that small central region. This makes the data much easier for the phone to handle, reducing heat and improving stability compared to a full-sensor high-res readout. Generally you can expect to achieve a x2 zoom this way, for every crop into 1/4 of the sensor area

But remember our “hand” analogy. That 12MP crop is unbinned. You are now back to using those “weak fingers” to hold up the image. Because that zoomed-in region isn’t benefiting from the noise-averaging of binning, you will notice that the image looks noisier, the contrast feels a bit duller, and the colors can look a bit “thin” or flat. Furthermore, you are running into the Optical Limit. Smartphone lenses are marvels of engineering, but they have physical limits; most are simply not sharp enough to resolve detail at a 200MP microscopic level.

At a certain point, you aren’t zooming into more detail; you’re just zooming into the limitations of the glass itself and it’s noise patterns. There’s diminishing returns to this mode, in other words. Lastly, few devices expose ISZ modes to third party apps, with Google Pixels being one of the few options allowing you to control their RAW streams. This leads us to the final aspect…

[8.6] The Gatekeeping Problem: OEM Restrictions and Root Overrides



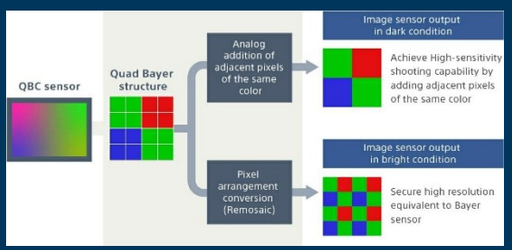
Finally, we have to address the often hostile or negligent behavior of phone manufacturers (OEMs). Even if your 200MP sensor is a masterpiece of engineering that comes with inherent flaws, many manufacturers purposely hide their high-resolution streams from third-party apps like MotionCam. They want you to stay in their “safe” 12MP binned lane where their own software can hide the sensor’s noise and flaws.

In many cases, even if we know the sensor can do +8K or +48MP RAW, the Camera2API parameters laid out by the manufacturer simply won’t let the app use those modes or provide with the access keys for them. This is why some users find that they need to root their devices to “force” the system to expose these hidden high-resolution modes.

Without root, we are often stuck with whatever “compliant” streams or sensor modes the manufacturer decides to allow. In my opinion, this is complete Bullshit, pardon my French… On the above image, you can see the RAW outputs available once you root some device (yes, even the 200MP stream can be used in-app).

Even though we now understand the limitations and practical use cases for the sensor binned and unbinned modes, we nevertheless believe it should be the end user whom determines whether they should activate them or not, rather than being protected against ourselves by the OEM. **Speak up and provide feedback accordingly to your OEM, only together can we seek to change this!**

[8.7] Closing Thoughts on Sensor Binning and ISZ; Do’s and Don’ts



So, how should you actually shoot, you ask? You should mostly respect the binning. Your sensor was designed to work as a team or “hand”, not as a collection of individual weak fingers trying to carry all the groceries in one trip just to flex how much we can hold.

Use the 12MP binned mode for 90% of your work, especially in low light or high frame rates. It provides the “fullest” and most robust cleanest signal for practical usage and intense color grading.

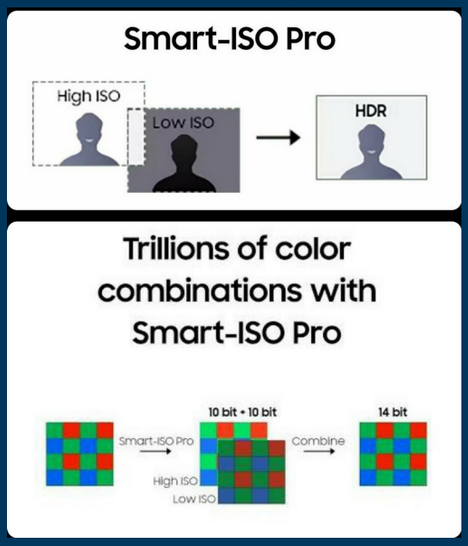
When available or if you’ve rooted to unlock them, use the 48/50MP “Unbinned” modes only when you have acceptable lighting and a specific need for 8K resolution, and be prepared for the 30fps limit. Also, be extremely cautious of the sizes. 8K RAW is absolutely enormous, you’ve never seen anything like it and it’s also too massive to run in Direct Log Video, so RAW is a must.

Use ISZ (Zoom) when you need the reach and have decent light, but be aware that your image will look “thinner” and noisier than your wide, binned shots. These modes avoid the brutal sizes of +8K/+48MP modes but are also not often available. Ensure you search for hidden lenses to search for them, they’ll generally appear as Lens IDs with double the focal length/zoom of one of the lenses (eg. 24mm lens may have 48mm variant in list).

In the world of RAW video, more megapixels usually just means more problems. Chase the quality of the pixels, not the quantity of them, and your footage will generally look superior!

[9] -Dual Conversion Gain (DCG) Sensor Capabilities-

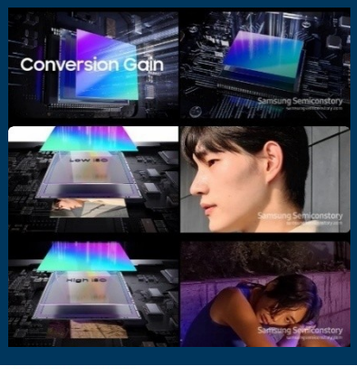
It is time for a bit of irony. We just spent an entire section bashing the marketing behind 48–200MP sensors, calling out the "Small Pixels" and the "weak fingers" of unbinned modes. But here is the plot twist: that exact same architecture, the Quad Bayer and Nona Bayer clusters we were just slamming is actually the secret requirement that enables one of the coolest technological breakthroughs in mobile cinematography.



While these sensors might be "faking" their resolution for the box art, **their unique physical layout is precisely what allows them to perform a hardware miracle called Dual Conversion Gain (DCG).** Although you may see it go by many names depending on the brand (DCG-HDR, Smart-ISO Pro, Dual Native ISO Fusion Max, etc.), the tech is practically identical!

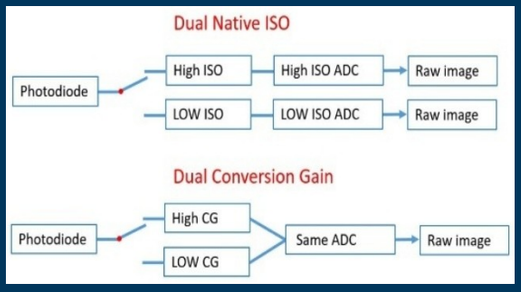
[9.1] The Evolution: From One to Two Amplifiers, then Both

To understand why this is a massive leap forward, we have to look at the history of how sensors "amplify" light, or the so called ISO “sensitivity”. For years, most phones came with a single gain amplifier **(a fancy way to call the circuitry that controls the ISO sensitivity/gain)**. You had one ADC (Analog-to-Digital Converter), and if you needed more brightness or light, you basically turned that single circuit up until it started flooding with sensor noise; essentially you cranked it up like a volume knob, up and down on demand to control the light sensitivity (or in reality, the Amplification of the signal)!



Then came Dual Gain. This gave sensors two specialized amplifiers (one for low ISO and one for high ISO) allowing you to "switch" to a cleaner circuit when the lights went down; one specialized in the lower sensitivity range, and another in the high sensitivity range (instead of having a jack of all trades) It was a huge improvement, but you still had to choose one or the other for each frame; you either protected the shadows or the highlights!

Dual **CONVERSION** Gain is the final piece of the puzzle. It takes those two amplifiers and runs them both simultaneously on a single exposure. This isn't a software trick or staggered HDR where the phone takes two different pictures (which causes ghosting). It is a hardware-level "stereo-like" readout where one single snapshot of light is processed through two different paths which end up converged (blended) at the exact same microsecond. The result means you no longer need to compromise! You can have the best of both worlds; have the cake, and eat it too!



[9.2] The Speaker Analogy: A Small Tweeter vs A Large Subwoofer

Think of it like having a dual speaker setup in your home.

- You have a Tweeter (Small Speaker), which represents your Low Conversion Gain (LCG); this is your low-ISO amplifier – it is tuned for the bright highlights, protecting the sky and the sun from clipping, but it doesn't quite perform well with deep shadows (lacks the sensitivity).

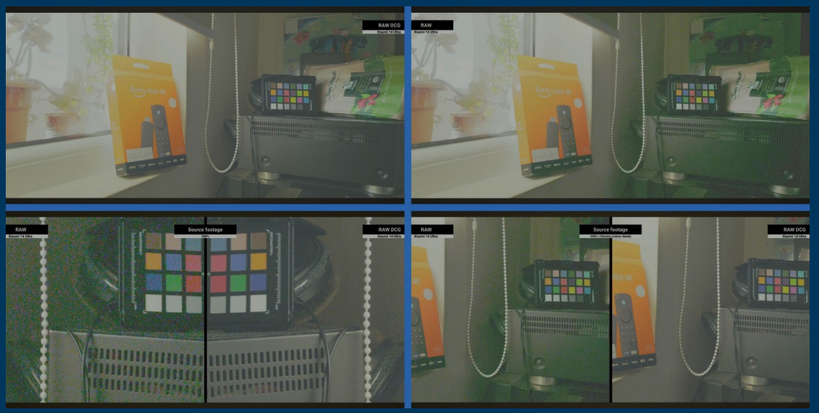
- You also have the Subwoofer (Big Speaker), which is your High Conversion Gain (HCG); this is your high-ISO amplifier – it is incredibly sensitive and brings out much more details in the dark, but it's not intended to be used with bright conditions as it can't quite hit the low range as well. Using HCG (a High ISO) in the sun would be like using a massive concert subwoofer to watch the morning news; it's the wrong tool for the task and results in a distorted, "blown out" mess.

Essentially, both Speakers, or in this case, Low and High ISOs have strengths and weaknesses when used independently. Low ISO can express better highlights and has a better noise profile, but it struggles with shadow details, while High ISO can express deeper shadows and dark details while keeping noise in check, however it cannot handle bright areas and suffers from reduced dynamic range.

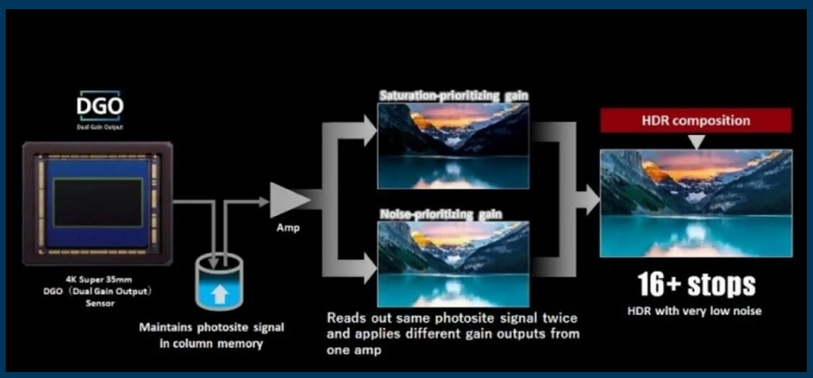
Previously it was really about knowing which to use at which time to compromise and use either the Low or High Conversion Gain respectively. With DCG however, this is no longer an issue. Dual Conversion Gain allows the sensor to Converge or otherwise merge the Low and the High ISO gain readouts (LCG + HCG) at the same time in every single exposure frame!

This is just like having a Stereo setup with your home speaker; the small and large speaker both receive the same audio signal (you aren't feeding them different audio inputs) but they amplify it at different levels at the same time in a coordinated manner that provides the best each has to offer on demand – same signal, two analog outputs covering wider spread: higher end quality!

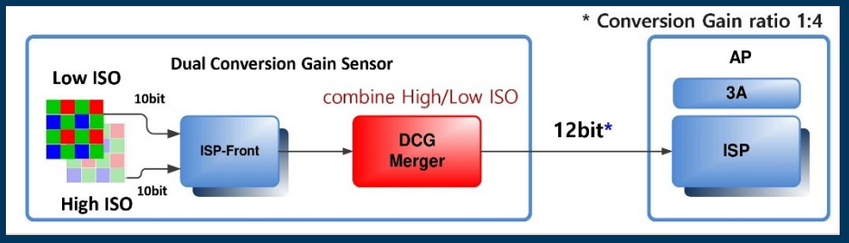
The best part is this method is invulnerable to motion or movement artifacts that otherwise plague past HDR methods (relying on separate, different time based Exposures or secondary readouts, no longer at the same moment in time)!



[9.3] Color Depth on Steroids:



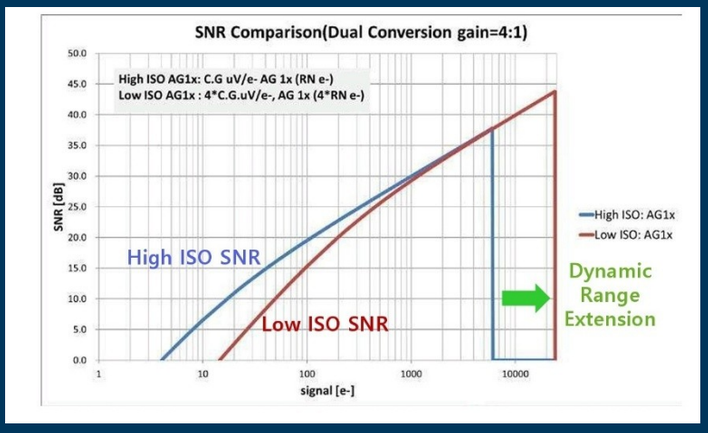
The technology is complex and you don't necessarily need to comprehend the architecture to use it, but it is made possible by the "clusters" in Quad and Nona Bayer sensors. The complex circuitry required for two separate Analog-to-Digital Converters (ADCs) needs the physical room and layout that these multi-pixel designs provide. It is essentially the mobile version of the legendary Arri Alexa DGA or Canon DGO cinema tech.



The sensor reads the HCG and LCG values and merges them at the high ISP level (not the part that handles processing, but rather the one that converts your sensor's photons into tangible digital data), essentially quantizing two 10-bit signals into a natively higher 12 or 14-bit output!!

(1 Low ISO at 10-bit + 1 High ISO at 10-bit = 12 or 14-bit single exposure)

Here's another cool aspect: unlike traditional camera ISO or Gain amplifiers with a fixed 'Base' ISO, phone ADCs (ISO amplifiers) behave like volume knobs, meaning that for example the The LCG might cover ISO 100–800, while the HCG kicks in from 800–3200. These are just examples but essentially even without DCG, having Dual Gain already provides with a neat advantage!



When running Dual Conversion Gain (Both LCG and HCG simultaneously) this does mean that the ISO readouts both must work in tandem and determine which ISO values they will each cover; **THEY DO NOT REMAIN STATIC AND WILL BOTH MOVE BASED ON ISO SELECTION**.

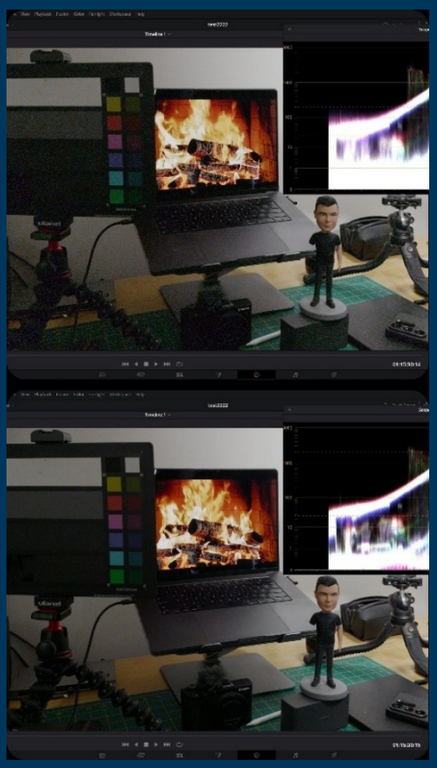
The "gap" between these two amplifiers is defined by the DCG Ratio. You will often see modes like DCG4 or DCG16. On some devices you control the Low ISO, on others you control the High ISO; the device then controls the opposite one (this will strongly depend on the device and how the root mod was developed or however the official driver was set-up, ask and test accordingly to confirm it).

DCG16: A massive 16:1 ratio spread (e.g., LCG at ISO 100 and HCG at ISO 1600 **OR** eg. LCG at ISO 200 and HCG at ISO 3200)

DCG4: A tighter 4:1 ratio spread (e.g., LCG at ISO 100 and HCG at ISO 400 **OR** eg. LCG at ISO 200 HCG at ISO 800)

By merging these two ISO ranges into a native single exposure, you gain a massive amount of tonal data long before the data gets turned into a file of any kind (whether photo, video or RAW frame data). This is what pushes your signal from a standard 10-bit, a 12-bit or even 14-bit shot, regardless of what container you place it into. That extra bit depth isn't just a vanity metric; it represents real, recovered information that allows you to grade your footage with professional flexibility!

Best of all, the additional readout provides the sensor with additional contextual data, allowing for unbelievable levels of noise control that can sometimes rival a Full Frame Mirrorless! Don't believe us? We've proved and posted the videos on YouTube!

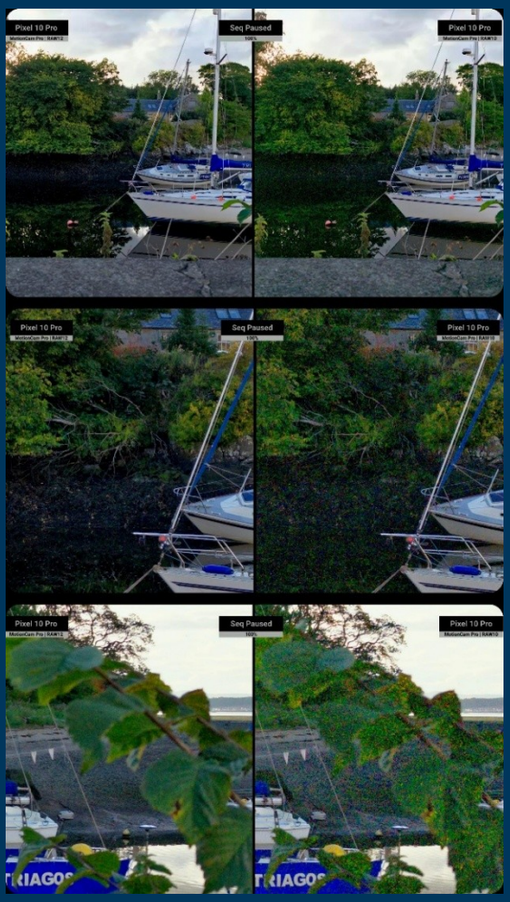


[9.4] The Breakthrough: Pixel 10 Pro

For the years, this was a "root only" secret. OEMs would use this tech for their own processed photos or sometimes not at all, but lock third-party apps out of the 12-bit hardware path. However, the Pixel 10 Pro recently made history as the first non-root device to natively expose these DCG streams to apps like MotionCam. It shows up as RAW12 (and also RAW_SENSOR), proving that we don't always need to hack our devices to get the 12-bit quality we were promised on all those past tech slides and keynotes.

If you want to see a comprehensive deep dive into how this looks in practice, I highly recommend checking out Elliot's summary at Epic-Tutorials, where he breaks down our findings and shows how DCG transforms noisy mobile shadows into clean, cinematic data. When you're in MotionCam, hunt for that RAW12 or RAW14 stream. If you see it, you've likely gained access to the unlocked hidden power of your sensor.

https://youtube.com/watch?v=YVj6JYXF14M



[10] -Root Mods and Enhancements-

So you've decided to potentially explore the viability of root, are at least mildly curious. Perhaps you've already taken the plunge and unlocked your bootloader! Well, let's define what root is and isn't, since this is often misunderstood to be a necessary app component given the amount of users who currently choose to root.

In order to root, you first need to unlock your bootloader. Think of this like a security label on a Tylenol bottle. When you open the bottle lid, it first comes with a safety seal to ensure the contents inside have not been tampered with. This is your bootloader basically, and unlocking it means taking that seal off; you can now practically put anything in the bottle (even happy pills), for better or worse. Unlocked bootloaders represent one thing to our community: Control. It allows users to take control of the OS in use, as well as access the root folder of a device.

The root folder is where things get interesting. Have you ever noticed when you get a new phone, there's already some storage in use that you cannot access? That storage area is not blank; rather, it is what holds all the existing files regarding your operating system, firmware, drivers, and much more. Most importantly for us, this is where you can find camera settings at a deep level, Camera2API configurations, CPU/GPU and Kernel affinities, thermal controls, and basically the stuff we can actually benefit from altering. Rooting essentially implies you inject a file that modifies the launch sequence of a device and grants you the ability to access those otherwise locked or hidden folders.

Make no mistake about it, root doesn't in itself improve the device; rather, it allows you to now change things if you understand what you're doing. Rooting itself doesn't mean you can simply click a button to unlock some crazy feature like DCG. This would be like me saying I can now change my car's Engine simply because I learned how to open the hood! Access is the first step, but the actual transformation requires the work of dedicated specialists.

[10.1] The Allure of the "Hidden" Hardware

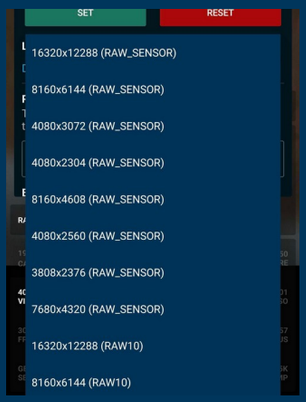
While MotionCam is an absolute powerhouse that works perfectly fine on many stock devices, rooting opens the door to functions that manufacturers often keep behind an artificial "paywall" or lock within their proprietary apps. This is where the proposition becomes alluring. For a power user, root is the tool used to bypass the Gatekeeping we've discussed throughout this manual.

One of the most immediate benefits is Thermal Management. Manufacturers set very conservative thermal limits to ensure the phone never feels "hot" in a consumer's hand, often throttling your processor just as you're getting into a long RAW recording session. Rooting allows for thermal mods that raise these ceilings or change how the phone handles heat, ensuring sustained performance. Furthermore, we can dive into Storage Performance improvements (optimizing UFS read/write speeds) which is critical when you are trying to dump massive amounts of RAW data to your disk without dropping frames.

Beyond just "speed," we can look at efficiency through improved GPU drivers and undervolting. By fine-tuning the voltage your processor uses, you can actually reduce heat and power consumption while maintaining the same performance levels. This is "free" efficiency that OEMs rarely tune to perfection for every individual unit. These are but a few of the tweaks available.

[10.2] Unlocking the "Restricted" Streams

The most exciting aspect for camera users, however, is the ability to activate hidden sensor hardware modes. Many sensors are physically capable of features that the stock software simply ignores or hides from the Camera2API. Through root mods, we can often force the activation of things like…



Hidden Framerates: Unlocking 60fps or higher in RAW where the manufacturer caps them at low values like 30fps.

Restricted RAW Streams: Accessing 12-bit or 14-bit data paths that were "glued shut" by the vendor, or sensor crop modes to improve readout speeds

Advanced Sensor Modes: Forcing the use of DCG, qHDR (Quad HDR), ISZ (In-Sensor Zoom), or even the elusive Full Resolution/Unbinned RAW modes on high-megapixel sensors.

[10.3] The Architects of the Mod Scene

It is important to set realistic expectations here: these mods are incredibly complex. While the final Magisk modules we use might seem user-friendly (just a "click and install" affair of ready made apps by the mod devs) the engineering and reverse-engineering required to create them are highly skilled and demanding. **Users capable of this level of deep-system manipulation are extremely rare and in high demand within our community.**

We owe much of these breakthroughs to legendary creators like **George Kiaire, the "Camera Mod Master" behind the Prometheus mods and advanced DCG unlocks.** His work has physically changed what these sensors are allowed to do. Similarly, performance enthusiasts like **JohnTheFarmer** have spent countless hours perfecting CPU/GPU affinities and system scripts to ensure the OS stays out of MotionCam's way.

Crucial Warning: Mods are almost never interchangeable. A mod designed for a Xiaomi 14 will not work on a Samsung S24, and often won't even work on a Xiaomi 13. Every device has unique firmware, drivers, and addresses. Furthermore, there is never a guarantee that a mod will successfully unlock a feature. Some hardware is "fused" at the factory in a way that even root cannot bypass. You are entering a world of experimentation and unknown, uncharted lands.

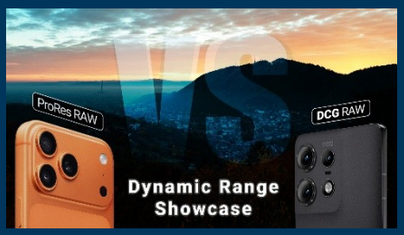
Lastly, take special note of the fact that some devices do not have active modding scenes around them, so while you may have a very highly robust mod selection on something like a Xiaomi 14 Ultra, you may have next to no mods on a random Oppo model. It’s strongly encouraged to shop around and ask in our communities to understand what you’re getting into!

[10.4] The App is the Powerhouse (Stock or Rooted)

I want to be absolutely clear: **MotionCam does NOT need root to be incredible**. Even in its "stock" and often classical 12MP 30fps modes, the app is an unprecedented miracle for mobile creation. The fact that we can get RAW video at all on a smartphone or even do real-time encoding via sheer brute force with no restrictions on the parameters is a feat of engineering,

Root is rarely the solution to an app "issue". In fact, mods actually prove how optimized the app is; when a modder unlocks 8K 30fps RAW or 14-bit DCG for example, the app handles it flawlessly even if the sensor produces partially optimized data, proving the bottleneck was always the OEM's artificial limits, not the app's code. MotionCam is now a tried and true powerhouse; root simply lets you take the "restrictor plate" off the engine.

[10.5] Flagship Killers aren’t Bought, They’re Tweaked



Rooting completely changes the game for device recommendations. Because of these mods, an otherwise "unimpressive" mid-range phone can suddenly become an elite performer that matches or beats flagships triple its price. You might find a $300 phone that, once rooted and modded by the community, gains 14-bit RAW features that even a $1,200 flagship lacks.

This makes recommending a device extremely difficult. A "good phone" for a beginner might be a "bad phone" for a pro who wants to root, and vice versa. Your choice of device is now entirely dependent on your personal skill level and your willingness to tinker.

[10.6] The War on Root – We’re Losing the Fight

We must face a harsh reality: The "Root" aspect of Android is dying and Unlockable Bootloaders are rapidly becoming a thing of the past. We are losing the fight for control. OEMs are locking down bootloaders more aggressively every year, using "security" as an excuse to take away your right to own the hardware you paid for. They want you in their walled garden, using their processed, over-sharpened "AI" looks and be happy with whatever they give you, even if half-baked.

We need users to speak up. Don't just settle for what the OEM gives you. Demand that they unlock the Camera2API and give us what is rightfully ours. Demand that they provide all the RAW streams and mode available. And just as important, fight back against the creeping intention they have to make us relinquish absolute control of our devices. Although we shouldn't have to "hack" our phones to use the hardware we bought, we’re often left with no recourse; they aim to take even this away, as if the current state of affairs wasn’t bad enough. If we don't push back, the ability to do what we do and push beyond arbitrary hardware limiters may eventually be locked away forever.

Because this landscape is ever-evolving and highly device-specific, I won’t go too far into the specific “how-to” of these mods here. What works today on a Pixel might be different for a Xiaomi. Instead, I highly encourage you to join our community and follow the dedicated channels for your specific device. That is where the real “magic” is brewed, and where you can find the latest stable releases from these developers!

[11] Epilogue: Constant Learning & Change

If you've made it this far, I'd like to thank you for choosing this resource as your companion through this journey. You've officially graduated from being a casual user to becoming a knight of mobile imaging and videography. My goal throughout this guide has been to imbue all my knowledge and personality into a single, unified resource—with all of my capacities, knowledge, and even my limitations. You've seen the man behind the curtain, you've learned how to spot OEMs playing you from a mile away, and you now understand that your phone isn't just a multimedia device; it's a high-performance imaging sensor miracle monster that just happens to have a SIM card slot.

They always say: "Give a man fire and he'll be warm for a day; light a man on fire and he'll be warm for the rest of his life…" Or something deeply philosophical along those lines. Anyway, my goal hasn't been to give you a "magic settings" list. Those don't exist. My goal was to give you the intuition to look at a scene, understand the physics of the light hitting your sensor, and know exactly how to harness what the app can and cannot do. You now have the power to push your sensor beyond what ordinary camera users believe to be even remotely possible.

Whether you are leveraging the quality of RAW video, playing with log captures, experimenting with the power of Dual Conversion Gain, or navigating the high-wire act of Rooted mods, you are now operating with a level of control that 99% of smartphone users don't even know exists.

The world of MotionCam moves fast. While I have poured everything I know into these pages, the reality of Android development is that the ground is always shifting, and so is the app. A new security patch might break a long-standing mod, or a new sensor architecture breakthrough might render an old analogy obsolete. Likewise, I expect upcoming updates already underway as I write this—radical changes like real-time Direct Preview during capture, ISO/Shutter Priority, Float Audio, and ACES tonemapping—to necessitate aggressive revisions to this content.

None of this would be possible without the relentless, mad-scientist energy of the MotionCam development team. These are the people pulling rabbits out of hats that didn't even have bottoms, working tirelessly to turn our "consumer" gadgets into professional-grade cinema instruments. They are the architects of this revolution, and their commitment to the community is the reason we have these "impossible" features in our pockets.

Because of this, consider this manual a living document. I am committed to keeping the information presented as accurate and up to date as I can, but I am only one person. If you spot an error, find a bug in my logic, or discover a new breakthrough that needs to be shared, I want to hear from you!

This isn't a lecture; it's a peer-to-peer exchange. I fully intended this to be a fun how-to book loaded with theory but also readability and practical applications. If something is unclear, or if you have feedback or questions, reach out to me (ragusaucy96) on Discord or Reddit! Your feedback is what will keep this guide sharp and ensure we stay one step ahead of changes.

The foundation you've built by reading this manual will be essential for what's coming next. We are moving toward a future where the limitations of your phone are defined only by your skill, not by a corporate boardroom.

In spite of all this, I want to ensure you remember one thing at all times: At the end of the day, all the technical knowledge in the world doesn't matter if you aren't out there using it. Don't be afraid to fail. Don't be afraid of a little noise. Don't be afraid to create, and don't let the OEMs tell you what your hardware can or cannot do.

So, without further ado, let us stop talking—and let us start creating something meaningful!

-RaguSaucy, AKA Armando