Compatible with NVIDIA Jetson Nano Camera IMX219-160 8-Megapixels Camera Module 3280 × 2464 Resolution 160 Degree Wide Angle of View with IMX219 Sensor

About this item

- IMX219-160 Camera, Supports NVIDIA Jetson Nano Developer Kit ,160° FOV.
- 3280 × 2464 High Resolution ,8 Megapixels Camera
- Sensor: IMX219
- Light and easy to use
- Combined with the Jetson Nano AI computer, this camera suits for AI projects such as:Face recognition,road mark detection,license plate recognition

Note: Products with electrical plugs are designed for use in the US. Outlets and voltage differ internationally and this product may require an adapter or converter for use in your destination. Please check compatibility before purchasing.

Product Description

Introduction
IMX219 Camera, 8 Megapixels Wide Angel of View IMX219 AI Camera for NVIDIA Jetson Nano Developer Kit. You can also use it with CM3/CM3+
expansion boards like Raspberry Pi Compute Module IO board, Compute Module IO Board Plus, Compute Module POE Board of Waveshare and the
StereoPi board.

Hardware connection
Connect the camera to CSI interface of Jetson Nano.Set the metal side of FFC into Heat-sink
Connect a HDMI LCD to Jetson NanoLens

specifications:
CMOS size: 1/4inch
Aperture (F): 2.35
Focal Length: 3.15mm
Angle of View (diagonal): 160 degree
Distortion: <14.3%4 screw holes
Used for attachment
Provides 3.3V power output
Dimension: 25mm × 24mm

Applications
Combined with the Jetson Nano AI computer, this camera suits for AI projects such as:
Face recognition
Road mark detection
License plate recognition

Package Content
IMX219-160 Camera x1
15-pin FFC (opposite sides contact) x1

Test with Jetson Nano

Hardware connection

- Insert the camera cable with the metal side facing the heatsink into the camera port on the Jetson Nano development kit.
    - Start the Jetson Nano.

- Test camera.
    - Open the terminal (press Ctrl+ALT+T shortcut on the keyboard to open the terminal), and enter the following command to test the camera.

```
DISPLAY=:0.0 nvgstcapture-1.0
```

- Test the dual camera.
    - If you need to test dual cameras, you can add sensor-id=x to select the camera. x can be 0 or 1.

```
#test video0
DISPLAY=:0.0 nvgstcapture-1.0 --sensor-id=0
#test video1
DISPLAY=:0.0 nvgstcapture-1.0 --sensor-id=1
```

- If the camera shooting effect is reddish, you can follow the steps below:

1. Download the camera-override.isp file and extract it to a specific folder:

```
wget https://files.waveshare.com/upload/e/eb/Camera_overrides.tar.gz
tar zxvf Camera_overrides.tar.gz
sudo cp camera_overrides.isp /var/nvidia/nvcam/settings/
```

2. Install files:

```
sudo chmod 664 /var/nvidia/nvcam/settings/camera_overrides.isp
sudo chown root:root /var/nvidia/nvcam/settings/camera_overrides.isp
```

【Notice】
- 12 of NV12 is a number, not a letter.

- The test screen is output to HDMI or DP screen, so when testing, first connect the screen to Jetson Nano.

- Opencv calls the camera

If you want to use the opencv library to call the camera, you can refer to the official jetcam program

Test with Compute Module

This IMX219 series camera cannot be used on the Raspberry Pi board due to the encryption, and can only be used with the carrier board on the computing module.
The use of IMX219 series cameras is the same as that of other Raspberry Pi cameras.

- Connect the camera to Compute Module Carrier board (like Compute Module 4 PoE Board), please take care of the direction.

Introduction

After the Bullseye version, the underlying Raspberry Pi driver for the Raspberry Pi image has been switched from Raspicam to libcamera. Libcamera is an open-source software stack (referred to as a driver here for ease of understanding) that is convenient for third-party porting and developing their own camera drivers. As of February 7, 2023, the official pycamera2 library has been provided for libcamera, making it easier for users to call Python demos.

Call Camera

The libcamera software stack provides six instructions for users to preview and test the camera interface.

libcamera-hello

This is a simple "hello world" program that previews the camera and displays the camera image on the screen.
Example

```
libcamera-hello
```

This command will preview the camera on the screen for about 5 seconds. The user can use the "-t <duration>" parameter to set the preview time, where the unit of <duration> is milliseconds. If it is set to 0, it will keep previewing all the time. For example:

```
libcamerahello -t 0
```

Tune File
The libcamera driver of the Raspberry Pi will call a tuning file for different camera modules. The tuning file provides various parameters. When calling the camera, libcamera will call the parameters in the tuning file, and process the image in combination with the algorithm. The final output is the preview screen. Since the libcamera driver can only automatically receive the signal of the chip, the final display effect of the camera will also be affected by the entire module. The use of the tuning file is to flexibly handle the cameras of different modules and adjust to improve the image quality.
If the output image of the camera is not ideal after using the default tuning file, the user can

adjust the image by calling the custom tuning file. For example, if you are using the official NOIR version of the camera, the NOIR camera may require different white balance parameters compared with the regular Raspberry Pi Camera V2. In this case, you can switch by calling the tuning file.

```
libcamera-hello --tuning-file
/usr/share/libcamera/ipa/raspberrypi/imx219_noir.json
```

Users can copy the default tuning files and modify them according to their needs.
Note: The use of tuning files is applicable to other libcamera commands, and will not be introduced in subsequent commands.

Preview Window
Most libcamera commands will display a preview window on the screen. Users can customize the title information of the preview window through the --info-text parameter, and can also call some camera parameters through %directives and display them on the window.
For example, if you use HQ Camera: You can display the focal length of the camera on the window through --info-txe "%focus".

```
libcamera-hello --info-text "focus %focus".
```

Note: For more information on parameter settings, please refer to the following chapters.

libcamera-jpeg

libcamera-jpeg is a simple still picture shooting program, different from the complex functions of libcamera-still, libcamera-jpeg code is more concise and has many of the same functions to complete picture shooting.
Take JPEG image of full pixel

```
libcamera-jpeg -o test.jpg
```

This shooting command will display a preview serial port for about 5 seconds, and then shoot a full-pixel JPEG image and save it as test.jpg.
Users can set the preview time through the -t parameter and can set the resolution of the captured image through --width and --height. E.g:

```
libcamera-jpeg -o test.jpg -t 2000 --width 640 --height 480
```

Exposure control
All libcamera commands allow the user to set the shutter time and gain themselves, such as:

```
libcamera-jpeg -o test.jpg -t 2000 --shutter 20000 --gain 1.5
```

This command will capture an image with 20ms exposure and camera gain set to 1.5x. The gain parameter set will first set the analog gain parameter inside the photosensitive chip. If the set gain exceeds the maximum built-in analog gain value of the driver, the maximum analog gain of

the chip will be set first, and then the remaining gain multiples will be used as numbers. gain to take effect.
Remarks: The digital gain is realized by ISP (image signal processing), not directly adjusting the built-in register of the chip. Under normal circumstances, the default digital gain is close to 1.0, unless there are the following three situations.

1. Overall gain parameter requirements, that is, when the analog gain cannot meet the set gain parameter requirements, the digital gain will be needed for compensation.
2. One of the color gains is less than 1 (the color gain is achieved by digital gain), in this case, the final gain is stabilized at 1/min(red_gain, blue_gain), that is, a uniform number is actually applied gain, and is the gain value for one of the color channels (not the green channel).
3. AEC/AGC was modified. If there is a change in AEC/AGC, the numerical gain will also change to a certain extent, where does it come from to eliminate any fluctuations, but this change will be quickly restored to the "normal" value.

The Raspberry Pi's AEC/AGX algorithm allows the program to specify exposure compensation, which is to adjust the brightness of the image by setting the aperture value, for example:

```
libcamera-jpeg --ev -0.5 -o darker.jpg
libcamera-jpeg --ev 0 -o normal.jpg
libcamera-jpeg --ev 0.5 -o brighter.jpg
```

libcamera-still

libcamera-still and libcamera-jpeg are very similar, the difference is that libcamera inherits more functions of raspistill. As before, the user can take a picture with the following command.
Test Command

```
libcamera-still -o test.jpg
```

Encoder
libcamea-still supports image files in different formats, can support png and bmp encoding, and also supports saving binary dumps of RGB or YUV pixels as files without encoding or in any image format. If you save RGB or YUV data directly, the program must understand the pixel arrangement of the file when reading such files.

```
libcamera-still -e png -o test.png
libcamera-still -e bmp -o test.bmp
libcamera-still -e rgb -o test.data
libcamera-still -e yuv420 -o test.data
```

Note: The format of image saving is controlled by the -e parameter. If the -e parameter is not called, it will be saved in the format of the output file name by default.
Raw Image Capture
The raw image is the image output by the direct image sensor without any ISP or CPU processing. For color camera sensors, the output format of the raw image is generally Bayer. Note that the raw image is different from the bit-encoded RGB and YUV images we said earlier, and RGB and YUV are also ISP-processed images.
Instructions to take a raw image:

```
libcamera-still -r -o test.jpg
```

The original image is generally saved in DNG (Adobe Digital Negative) format, which is compatible with most standard programs, such as dcraw or RawTherapee. The original image will be saved as a file of the same name with the .dng suffix, for example, if you run the above command, it will be saved as a test.dng, and generate a jpeg file at the same time. The DNG file contains metadata related to image acquisition, such as white balance data, ISP color matrix, etc. The following is the metadata encoding information displayed by the exiftool:

```
File Name                       : test.dng
Directory                       : .
File Size                       : 24 MB
File Modification Date/Time     : 2021:08:17 16:36:18+01:00
File Access Date/Time           : 2021:08:17 16:36:18+01:00
File Inode Change Date/Time     : 2021:08:17 16:36:18+01:00
File Permissions                : rw-r--r--
File Type                       : DNG
File Type Extension             : dng
MIME Type                       : image/x-adobe-dng
Exif Byte Order                 : Little-endian (Intel, II)
Make                            : Raspberry Pi
Camera Model Name               : /base/soc/i2c0mux/i2c@1/imx477@1a
Orientation                     : Horizontal (normal)
Software                        : libcamera-still
Subfile Type                    : Full-resolution Image
Image Width                     : 4056
Image Height                    : 3040
Bits Per Sample                 : 16
Compression                     : Uncompressed
Photometric Interpretation      : Color Filter Array
Samples Per Pixel               : 1
Planar Configuration            : Chunky
CFA Repeat Pattern Dim          : 2 2
CFA Pattern 2                   : 2 1 1 0
Black Level Repeat Dim          : 2 2
Black Level                     : 256 256 256 256
White Level                     : 4095
DNG Version                     : 1.1.0.0
DNG Backward Version            : 1.0.0.0
Unique Camera Model             : /base/soc/i2c0mux/i2c@1/imx477@1a
Color Matrix 1                  : 0.8545269369 -0.2382823821 -0.09044229197 -
0.1890484985 1.063961506 0.1062747385 -0.01334283455 0.1440163847
0.2593136724
As Shot Neutral                 : 0.4754476844 1 0.413686484
Calibration Illuminant 1        : D65
Strip Offsets                   : 0
Strip Byte Counts               : 0
Exposure Time                   : 1/20
ISO                             : 400
CFA Pattern                     : [Blue, Green][Green, Red]
Image Size                      : 4056x3040
Megapixels                      : 12.3
Shutter Speed                   : 1/20
```

Long Exposure
If we want to take a super long exposure picture, we need to disable AEC/AGC and white balance, otherwise, these algorithms will cause the picture to wait for a lot of frame data when it converges. Disabling these algorithms requires another explicit value to be set. Additionally, the user can skip the preview process with the --immediate setting.
Here is the instruction to take an image with an exposure of 100 seconds:

```
libcamera-still -o long_exposure.jpg --shutter 100000000 --gain 1 --awbgains
1,1 --immediate
```

Remarks: Reference table for the longest exposure time of several official cameras.

Module	Maximum exposure time (s)
V1 (OV5647)	6
V2 (IMX219)	11.76
V3 (IMX708)	112
HQ (IMX477)	670

libcamera-vid

libcamera-vid is a video recording program that uses the Raspberry Pi hardware H.264 encoder by default. After the program runs, a preview window will be displayed on the screen, and the bitstream encoding will be output to the specified file. For example, record a 10s video.

```
libcamera-vid -t 10000 -o test.h264
```

If you want to view the video, you can use vlc to play it.

```
vlc test.h264
```

Note: The recorded video stream is unpackaged. Users can use --save-pts to set the output timestamp to facilitate the subsequent conversion of the bit stream to other video formats.

```
libcamera-vid -o test.h264 --save-pts timestamps.txt
```

If you want to output the mkv file, you can use the following command:

```
mkvmerge -o test.mkv --timecodes 0:timestamps.txt test.h264
```

Encoder
Raspberry Pi supports JPEG format and YUV420 without compression and format:

```
libcamera-vid -t 10000 --codec mjpeg -o test.mjpeg
libcamera-vid -t 10000 --codec yuv420 -o test.data
```

The --codec option sets the output format, not the output file extension.
Use the --segment parameter to split the output file into segments (unit is ms), which is suitable for JPEG files that need to split the JPEG video stream into separate short (about 1ms) JPEG files.

```
libcamera-vid -t 10000 --codec mjpeg --segment 1 -o test%05d.jpeg
```

UDP Video Streaming Transmission
UDP can be used for video streaming, and the Raspberry Pi runs (server):

```
libcamera-vid -t 0 --inline -o udp://<ip-addr>:<port>
```

Among them, <ip-addr> needs to be replaced with the actual client IP address or multicast address. On the client (client), enter the following commands to obtain and display the video stream (you can use one of the two commands);

```
vlc udp://@:<port> :demux=h264
vlc udp://@:<port> :demux=h264
```

Note: The port needs to be the same as the one you set on the Raspberry Pi.
TCP Video Streaming Transmission
You can use TCP for video streaming, and the Raspberry Pi runs (server):

```
libcamera-vid -t 0 --inline --listen -o tcp://0.0.0.0:<port>
```

The client runs:

```
vlc tcp/h264://<ip-addr-of-server>:<port> #Select one of the two commands
ffplay tcp://<ip-addr-of-server>:<port> -vf "setpts=N/30" -fflags nobuffer -
flags low_delay -framedrop
```

RTSP Video Streaming Transmission
On the Raspberry Pi, vlc is usually used to process the RTSP video stream:

```
libcamera-vid -t 0 --inline -o - | cvlc stream:///dev/stdin --sout
'#rtp{sdp=rtsp://:8554/stream1}' :demux=h264
```

On the playback side, you can run any of the following commands:

```
vlc rtsp://<ip-addr-of-server>:8554/stream1
ffplay rtsp://<ip-addr-of-server>:8554/stream1 -vf "setpts=N/30" -fflags
nobuffer -flags low_delay -framedrop
```

In all preview commands, if you want to turn off the preview window on the Raspberry Pi, you can use the parameter -n (--nopreview) to set it. Also, pay attention to the setting of the --inline parameter. Changing the setting will force the header information of the video stream to be included in each I (intra) frame. This setting allows the client to correctly parse the video stream even if the video header is lost.
High Frame Rate Mode

If you use the libcamera-vid command to record high frame rate video (generally higher than 60fps) while reducing frame loss, you need to pay attention to the following points:

1. The target level of H.264 needs to be set to 4.2, which can be set with the **--level 4.2** parameter.
2. The color noise reduction function must be turned off, you can use the **--denoise cdn_off** parameter setting.
3. If the set frame rate is higher than 100fps, close the preview window to release more CPU resources and avoid frame loss. Can be set using the parameter **-n**.
4. It is recommended to add the setting force_turbo=1 in the /boot/config.txt file to ensure that the CPU clock will not be limited during the video stream or in the middle.
5. Adjust the ISP output resolution, use **-width 1280 --height 720** to set the resolution, or set it to a lower resolution, depending on the camera model.
6. If you are using Pi 4 or this higher-performance model, you can add the setting gpu_freq=550 or higher in the /boot/config.txt file to overclock the motherboard GPU to achieve higher performance.

For example, record 1280x720 120fps video.

```
libcamera-vid --level 4.2 --framerate 120 --width 1280 --height 720 --save-
pts timestamp.pts -o video.264 -t 10000 --denoise cdn_off -n
```

libcamera-raw

Libcamera-raw is similar to a video recording program. In different places, libcamera-raw records the Bayer format data output by the direct sensor, that is, the original image data. Libcamera-raw doesn't show a preview window. For example, record a 2-second clip of raw data.

```
libcamera-raw -t 2000 -o test.raw
```

The program will directly dump the original frame without format information, the program will directly print the pixel format and image size on the terminal, and the user can view the pixel data according to the output data.
By default, the program will save the original frame as a file, the file is usually large, and the user can divide the file by the --segement parameter.

```
libcamera-raw -t 2000 --segment 1 -o test%05d.raw
```

If the memory is large (such as using SSD), libcamera-raw can write the official HQ Camera data (about 18MB per frame) to the hard disk at a speed of about 10 frames per second. In order to achieve this speed, the program writes the unformatted raw frames, there is no way to save them as DNG files like libcamera-still does. If you want to ensure that there are no dropped frames, you can use --framerate to reduce the frame rate.

```
libcamera-raw -t 5000 --width 4056 --height 3040 -o test.raw --framerate 8
```

Common Command Setting Options

Common command setting options apply to all libcamera commands:

```
--help, -h
```

Print program help information, you can print the available setting options for each program command, and then exit.

```
--version
```

Print the software version, print the software version of libcamera and libcamera-app, then exit.

```
--list-cameras
```

Displays the recognized supported cameras. for example:

```
Available cameras
-----------------
0 : imx219 [3280x2464] (/base/soc/i2c0mux/i2c@1/imx219@10)
    Modes: 'SRGGB10_CSI2P': 640x480 [206.65 fps - (1000, 752)/1280x960 crop]
                            1640x1232 [41.85 fps - (0, 0)/3280x2464 crop]
                            1920x1080 [47.57 fps - (680, 692)/1920x1080
crop]
                            3280x2464 [21.19 fps - (0, 0)/3280x2464 crop]
           'SRGGB8' : 640x480 [206.65 fps - (1000, 752)/1280x960 crop]
                      1640x1232 [41.85 fps - (0, 0)/3280x2464 crop]
                      1920x1080 [47.57 fps - (680, 692)/1920x1080 crop]
                      3280x2464 [21.19 fps - (0, 0)/3280x2464 crop]
1 : imx477 [4056x3040] (/base/soc/i2c0mux/i2c@1/imx477@1a)
    Modes: 'SRGGB10_CSI2P': 1332x990 [120.05 fps - (696, 528)/2664x1980 crop]
           'SRGGB12_CSI2P': 2028x1080 [50.03 fps - (0, 440)/4056x2160 crop]
                            2028x1520 [40.01 fps - (0, 0)/4056x3040 crop]
                            4056x3040 [10.00 fps - (0, 0)/4056x3040 crop]
```

According to the printed information, the IMX219 camera has a suffix of 0, and the IM new 477 camera has a suffix of 1. When calling the camera, you can specify the corresponding suffix.

```
--camera
```

Specify the camera, and the corresponding suffix can refer to the print information of the command **--list-camera**.
For example: libcamera-hello -c config.txt
In the setting file, set parameters one line at a time, in the format of key=value:

```
timeout=99000
verbose=
--config,      -c
```

Under normal circumstances, we can directly set the camera parameters through commands. Here we use the --config parameter to specify the setting file and directly read the setting parameters in the file to set the camera preview effect.

```
--timeout, -t
```

The "-t" option sets the runtime of the libcamera demo. If the video recording command is run, the timeout option sets the recording duration. If the image capture command is run, the timeout sets the preview time before the image is captured and output.
If the timeout is not set when running the libcamera demo, the default timeout value is 5000 (5 seconds). If the timeout is set to 0, the demo will continue to run.
Example: *libcamera-hello -t 0*

`--preview, -p`

"-p" sets the size and position of the preview window (the qualified settings are valid in both X and DRM version windows), and the format is --preview <x. y, w, h> where "x, y" sets the preview window coordinate, "w" and "h" set the width and length of the preview window.
The settings of the preview serial port will not affect the resolution and aspect ratio of the camera image preview. The demo will scale the preview image to display in the preview window and adapt it according to the original image aspect ratio.
Example: *libcamera-hello -p 100,100,500,500*

`--fullscreen, -f`

The "-f" option sets the preview window full screen, the preview window, and the border in full-screen mode. Like "-p", it does not affect the resolution and aspect ratio, and will automatically adapt.
Example: *libcamera-still -f -o test.jpg*

`--qt-preview`

Using the preview window based on the QT framework, this setting is not recommended under normal circumstances, because the preview demo will not use zero-copy buffer sharing and GPU acceleration, which will occupy more resources. The QT preview window supports X forwarding (the default preview program does not).
The Qt preview serial port does not support the "--fullscreen" setting option. If the user wants to use the Qt preview, it is recommended to keep a small preview window to avoid excessive resource usage and affecting the normal operation of the system.
Example: *libcamera-hello --qt-preview*

`--nopreview, -n`

Images are not previewed. This setting will turn off the image preview function.
Example: *libcamera-hello -n*

`--info-text`

Set the title and information display of the preview window (only available in the X graphics window) using the format --info-text <string>. When calling this option, there are multiple parameters that can be set, and the parameters are usually called in the % command format. The demo will call the corresponding value in the graphics metadata according to the instruction.
If no window info is specified, the default --info-text is set to *"#%frame (%fps fps) exp %exp ag*

%ag dg %dg"
Example: *libcamera-hello --info-test "Focus measure: %focus*
Available parameters:

Instructions	**Instructions**
%frame	Frame sequence number
%fps	Instantaneous frame rate
%exp	The shutter speed when capturing the image, in ms
%ag	Image analog gain controlled by the sensor chip
%dg	Image value gain controlled by ISP
%rg	Gain of the red component of each pixel
%bg	The gain of the blue component of each pixel
%focus	The corner measurement of the image, the larger the value, the clearer the image
%lp	Diopter of the current lens (1/distance in meters)
%afstate	Autofocus state (idle, scanning, focused, failed)

```
--width
--height
```

These two parameters set the width and height of the image, respectively. For libcamera-still, libcamera-jpeg and libcamera-vid commands, these two parameters can set the resolution of the output image/video.
If the libcamera-raw command is used, these two parameters affect the size of the obtained metadata frame. The camera has a 2 x 2 block reading mode. If the set resolution is smaller than the split mode, the camera will obtain the metadata frame according to the 2 x 2 block size. libcamera-hello cannot specify the resolution.
Example:
libcamera-vid -o test.h264 --width 1920 --height 1080 Record 1080p video
libcamera-still -r -o test.jpg --width 2028 --height 1520 Takes a 2028 x 1520 JPEG image.

```
--viewfinder-width
--viewfinder-height
```

This setting option is also used to set the resolution of the image, the difference is only the image size of the preview. It does not affect the final output image or video resolution. The size of the preview image will not affect the size of the preview window and will be adapted according to the window.
Example: *libcamera-hello --viewfinder-width 640 --viewfinder-height 480.*

```
--rawfull
```

This setting forces the sensor to use the --width and --height settings to output still images and video in full-resolution read mode. This setting libcamera-hello has no effect.
With this setting, the framerate is sacrificed. In full-resolution mode, frame reading will be slower.
Example: *libcamera-raw -t 2000 --segment 1 --rawfull -o test%03d.raw* The example command captures multiple full-resolution images Metadata frames in rate mode. If you are using an HQ camera. The size of each frame is 18MB, and if --rawfull is not set, the HQ camera defaults to 2 x 2 mode, and the data size of each frame is only 4.5MB.

```
--mode
```

This parameter is more general than rawfull. It is used to set the camera mode. When using it, you need to specify the width, height, bit depth, and packing mode, and separate them with colons. The set value does not have to be completely accurate, the system will automatically match the closest value, and the bit depth and packing mode can be set (the default is 12 and P means packing).

- **4056:3040:12:P** - 4056x3040 resolution, 12bit per pixel, packed. Packing means that the original image data will be packed in the buffer. In this case, two pixels will only occupy 3 bytes, which can save memory.
- **1632:1224:10** - 1632x1224 resolution, 10bit per pixel, packed by default. In 10-bit packing mode, 4-pixel data will occupy 5 bytes.
- **2592:1944:10:U** -2592x1944 resolution, 10 bits per pixel, no packing. In the case of unpacking, each speed limit will occupy 2bytes of memory, in this case, the highest 6 bits will be set to 0.
- **3262:2448** -3264x2448 resolution, 12bits, and packing mode are used by default. However, if the camera model, such as Camera V2 (IMX219), does not support 12bits mode, the system will automatically select 10bits mode.

```
--viewfinder-mode       #Specify sensor mode, given as <width>:<height>:<bit-depth>:<packing>
```

The --mode parameter is used to set the camera mode when recording video and shooting still images. If you want to set it when previewing, you can use the --viewfinder-mode parameter.

```
--lores-width
--lores-height
```

These two options set low-resolution images. The low-resolution data stream compresses the image, causing the aspect ratio of the image to change. When using libcamera-vid to record video, if a low resolution is set, functions such as color denoising will be disabled.
Example: *libcamera-hello --lores-width 224 --lores-height 224* Note that low-resolution settings are often used in conjunction with image postprocessing, otherwise it has little effect.

```
--hflip #Flip the image horizontally
--vflip #Flip the image vertically
```

```
--rotation #Flip the image horizontally or vertically according to the given
angle <angle>
```

These three options are used to flip the image. The parameters of --rotation currently only support 0 and 180, which are actually equivalent to --hflip and --vflip.
Example: *libcamera-hello --vflip --hflip*

```
--roi #Crop image <x, y, w, h>
```

"--roi" allows the user to crop the image area they want according to the coordinates from the complete image provided by the sensor, that is, digital scaling, paying attention to the coordinate value if it is in the valid range. For example --roi 0, 0, 1, 1 is an invalid instruction.
Example: *libcamera-hello --roi 0.25,0.25,0.5,0.5*
The example command will crop 1/4 of the image from the center of the image.

```
--hdr Run the camera in HDR mode (supported cameras only)
```

The hdr parameter is used to set the wide dynamic mode of the camera. This setting will only take effect if the camera supports a wide dynamic range. You can use --list-camera to see if the camera supports hdr mode.

```
--sharpness #Set the sharpness of the image <number>
```

Adjust the sharpness of the image by the value of <number>. If set to 0, no sharpening is applied. If you set a value above 1.0, an extra sharpening amount will be used.
Example: *libcamera-still -o test.jpg --sharpness 2.0*

```
--contrast #Set image contrast <number>
```

Example: *libcamera-still -o test.jpg --contrast 1.5*

```
--brightness #Set image brightness <number>
```

The setting range is -1.0 ~ 1.0
Example: *libcamera-still -o test.jpg --brightness 0.2*

```
--saturation #Set image color saturation <number>
```

Example: *libcamera-still -o test.jpg --saturation 0.8*

```
--ev #Set EV compensation <number>
```

Set the EV compensation of the image in aperture units, the setting range is -10 ~ 10, the default value is 0. The program works by improving the target method of the AEC/AGC algorithm.
Example: *libcamera-still -o test.jpg --ev 0.3*

```
--shutter #Set the exposure time, the unit is ms <number>
```

Note: If the frame rate of the camera is too fast, it may not work according to the set shutter time. If this happens, you can try to use --framerate to reduce the frame rate.
Example: *libcamera-hello --shutter 30000*

```
--gain #Set gain value (combination of numerical gain and analog gain)
<number>
--analoggain #--gain synonym
```

--analoggain is the same as --gain, the use of analoggain is only for compatibility with raspicam programs.

```
--metering #Set metering mode <string>
```

Set the metering mode of the AEC/AGC algorithm, the available parameters are:

- centre - Center metering (default)
- spot - spot metering
- average - average or full frame metering
- custom - custom metering mode, can be set via tuning file

Example: *libcamera-still -o test.jpg --metering spot*

```
--exposure #Set exposure profile <string>
```

The exposure mode can be set to normal or sport. The report profile for these two modes does not affect the overall exposure of the image, but in the case of sports mode, the program will shorten the exposure time and increase the justice to achieve the same exposure effect.
Example: *libcamera-still -o test.jpg --exposure sport*

```
--awb #Set white balance mode <string>
```

Available white balance modes:

Mode	Color Temperature
auto	2500K ~ 8000K
incadescent	2500K ~ 3000K
tungsten	3000K ~3500K
fluorescent	4000K ~ 4700K
indoor	3000K ~ 5000K
daylight	5500K ~ 6500K
cloudy	7000K ~ 8500K

custom Custom range, set via tuning file

Example: *libamera-still -o test.jpg --awb tungsten*

```
--awbgains #Set a fixed color gain <number,number>
```

Set red and blue gain.
Example: *libcamera-still -o test.jpg --awbgains 1.5, 2.0*

```
--denoise #Set denoising mode <string>
```

Supported denoising modes:

- auto - default mode, use standard spatial denoising, if it is video, it will use fast color noise reduction and use high-quality color noise reduction when taking still pictures. The preview image will not use any color denoising.
- off - turn off spatial denoising and color denoising.
- cdn_off - turn off color denoising.
- cdn_fast - use fast color denoising.
- cdn_hq - use high-quality color denoising, not suitable for video recording.

Example: *libcamera-vid -o test.h264 --denoise cdn_off*

```
--tuning-file #Specify camera tuning file <string>
```

For more instructions on tuning files, you can refer to official tutorial
Example: *libcamera-hello --tuning-file ~/my~camera-tuning.json*

```
--autofocus-mode                    Specify the autofocus mode <string>
```

Set the autofocus mode.

- default - By default, the camera will use continuous autofocus mode, unless --lens-position or --autofocus-on-capture manual focus is set.
- manual - manual focus mode, the focus position can be set by --lens-position.
- auto - focus will only be done once when the camera is turned on, and will not be adjusted in other cases. (If you use the libcamera-still command, only when --autofocus-on-capture is used, it will focus once before taking a photo).
- continuous - The camera will automatically adjust the focus position according to the scene changes.

```
--autofocus-range   Specify the autofocus range <string>
```

Set the autofocus range.

- normal -- the default item, from nearest to infinity.
- macro - macro mode, only focus on nearby objects.

- full - full distance mode, adjusted to infinity for the closest object.

```
--autofocus-speed   Specify the autofocus speed <string>
```

Set the focus speed.

- normal - default item, normal speed.
- fast - fast focus mode.

```
--autofocus-window   --autofocus-window
```

To display the focus window, you need to set x, y, width, height, and the coordinate value setting is based on the ratio of the image. For example **--autofocus-window 0.25,0.25,0.5,0.5** would set a window half the size of the image and centered.

```
--lens-position    Set the lens to a given position <string>
```

Set the focus position.

- 0.0 -- set the focus position to infinity
- number -- set the focus position to 1/number number is any value you set, for example, if you set 2, it means that it will focus on the position of 0.5m.
- default -- focus on the default position relative to the hyperfocal distance of the lens.

```
--output, -o #output file name <string>
```

Set the filename of the output image or video. In addition to setting the file name, you can also specify the output udp or tcp server address to output the image to the server. If you are interested, you can check the relevant setting instructions of the subsequent tcp and udp.
Example: *libcamera-vid -t 100000 -o test.h264*

```
--wrap #Wrap the output file counter <number>
```

Example: *libcamera-vid -t 0 --codec mjpeg --segment 1 --wrap 100 -o image%d.jpg*

```
--flush # Flush the output file immediately
```

--flush will immediately update each frame of the image to the hard disk at the same time as it is written, reducing latency.
Example: *libcamera-vid -t 10000 --flush -o test.h264*

Still Photo Shooting Setting Parameters

```
--quality, -q            #Set JPEG image quality <0 ~ 100>
--exif, -x               #Add extra EXIF flags
--timelapse              #Time interval of time-lapse photography, the unit
is ms
--framestart             #Start value of frame count
--datetime               #Name output file with date format
--timestamp              #Name the output file with the system timestamp
```

```
-- restart                #Set the JPEG restart interval
--keypress, -k            #Set the enter button photo mode
--signal, -s              #Set the signal to trigger the photo
--thumb                   #Set thumbnail parameters <w:h:q>
--ebcoding, -e            #Set the image encoding type. jpg/png/bmp/rgb/yuv420
--raw, -r                 #Save raw image
--latest                  #Associate symbols to the latest saved file
--autofocus-on-capture    #Set to do a focus action before taking a photo
```

Video Recording Image Setting Parameters

```
--quality, -q   #Set JPEG commands <0 - 100>
--bitrate, -b   #Set H.264 bitrate
--intra, -g     #Set the internal frame period (only supports H.264)
--profile       #Set H.264 configuration
--level         #Set H.264 level
--codec         #Set encoding type h264 / mjpeg / yuv420
--keypress, -k  #Set carriage return to pause and record
--signal, -s    #Set signal pause and record
--initial       #Start the program in the recording or paused state
--split         #Split video and save to another file
--segment       #Split video into multiple video segments
--circular      #Write video to the circular buffer
--inline        #Write header in each I frame (H.264 only)
--listen        #Wait for a TCP connection
--frames        #Set the number of frames recorded
```