



# UVC Camera Software Manual

Date	Version	Description
2023-10-19	V1.0	First Released



# 1 Description

- UVC cameras comply with UVC protocol and work with web-camera applications out-of-box
- UVC Cameras support windows, linux, MacOS Compatible with UVC drivers

## 1.1 What is UVC Camera

- UVC Camera is camera with a USB interface that meets the standards set for the USB Video Class. This means that every UVC Camera is a USB camera, but not all USB cameras are UVC Cameras, because they might adopt the USB interface without meeting the Video class requirements.
- Therefore, a major advantage of the UVC cameras is their universal compatibility and flexibility. As they meet the video class standard, you can easily use them on different platforms with a USB port without handling the driver issue, like the Raspberry Pi or a smartphone. It also makes it easier for you to migrate your applications from one platform to another.
- At present, our UVC cameras support Windows, Linux, MAC, and Android systems, but do not support the iPhone system.

# 2 Works on Windows

## 2.1 AMCAP

**AMCAP is a free and easily use UVC Camera test tools.**

### 2.1.1 Preview

**Open AMCAP.EXE, Select USB Camera From “Devices”, Select “Preview” from “Options”**

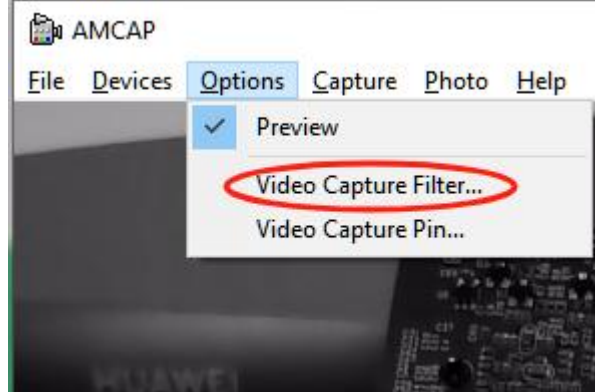


## UVC Camera SW Manual



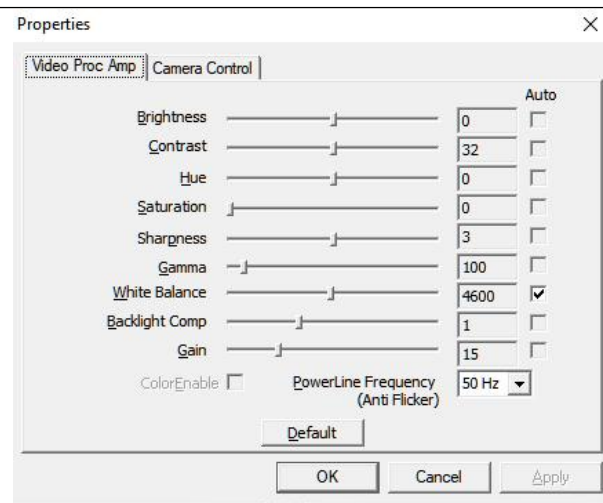
### 2.1.2 Video Capture Filter

You Can find most of Controllable Parameters from “Options”, “Video Capture Filter”.





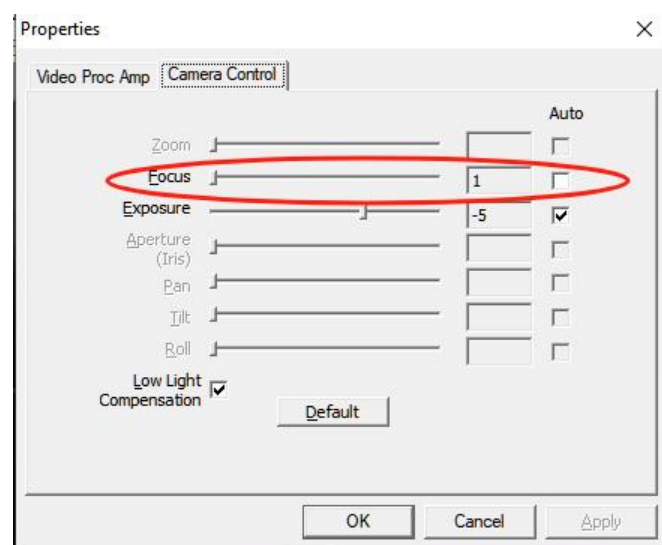
## UVC Camera SW Manual



Brightness, Contrast, Hue, Saturation, Sharpness, Gamma, White  
Balance, Backlight Comp, Gain, Exposure, PowerLine Frequency, Low Light  
Compensation

### 2.1.3 External Trigger Parameters

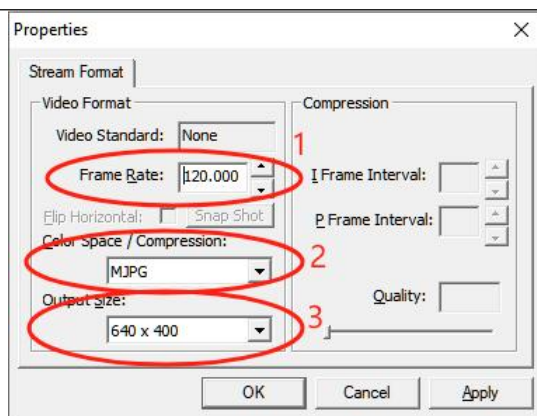
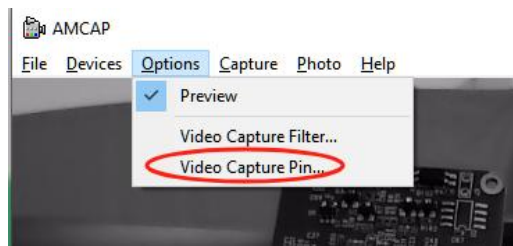
From “Video Capture Filter” “Camera Control” ,The “Focus” Parameter  
is for external trigger signal Enable.





### 2.1.3 Video Capture Pin

You Can find most of Controllable Parameters from “Options”, “Video Capture Pin”.



- |   |                                     |
|---|-------------------------------------|
| 1 | You can chage Frame rate            |
| 2 | Choose Output format like MJPG/YUV2 |
| 3 | Choose Resolution Camera Support    |

### 2.1.4 Status Bar

You can find live frame Rate, Output Resolution

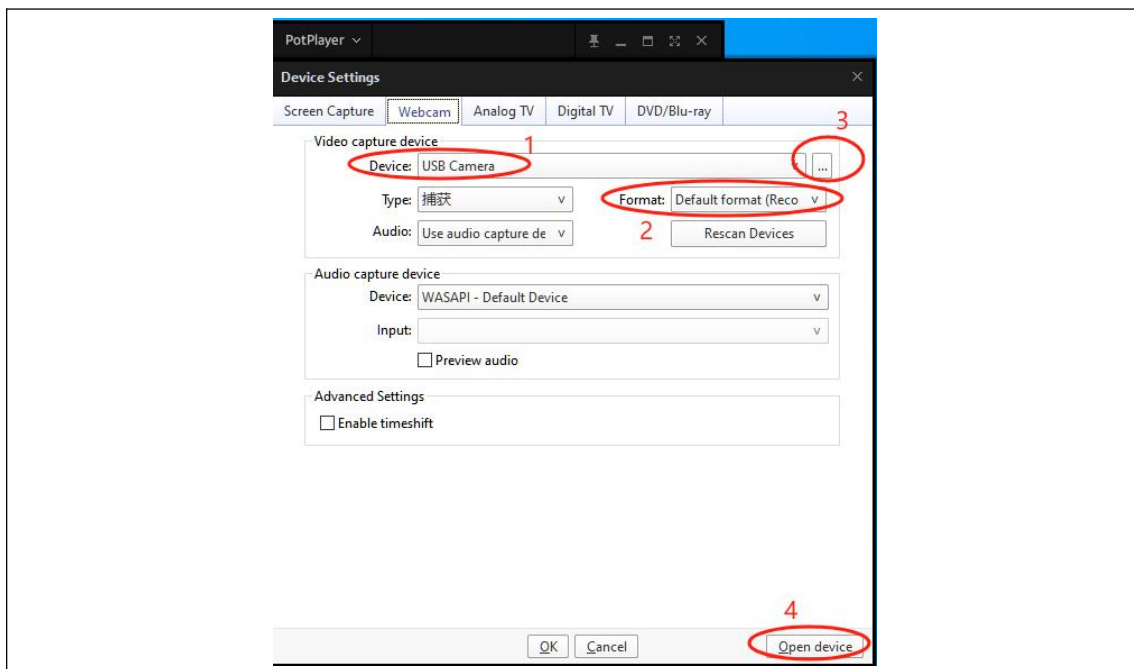


## 2.2 PotPlayer

Potplayer is another free Windows Tools which easily get video and images of UVC and U3V,UVC3.0 Cameras.

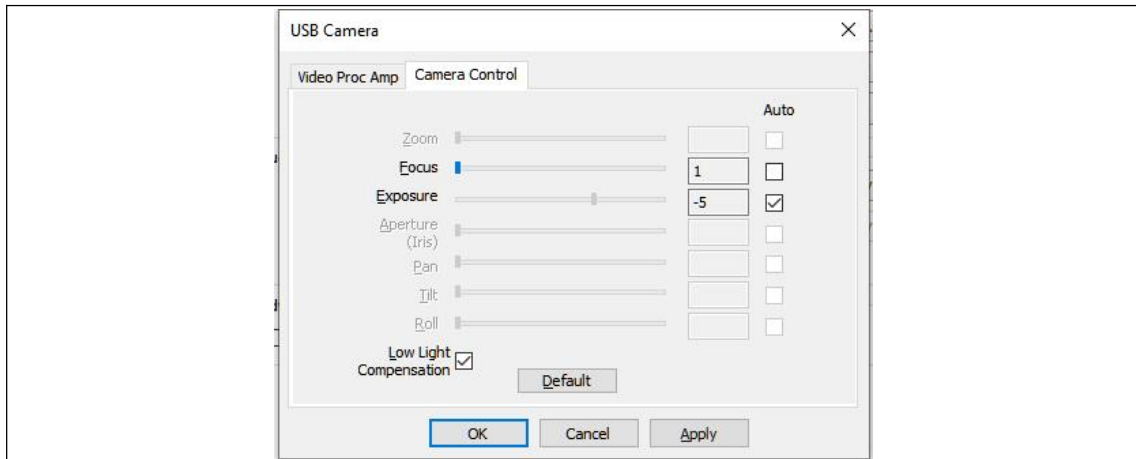
### 2.2.1 Open UVC Camera

Use Shortcut Key **ALT+D** open window as above





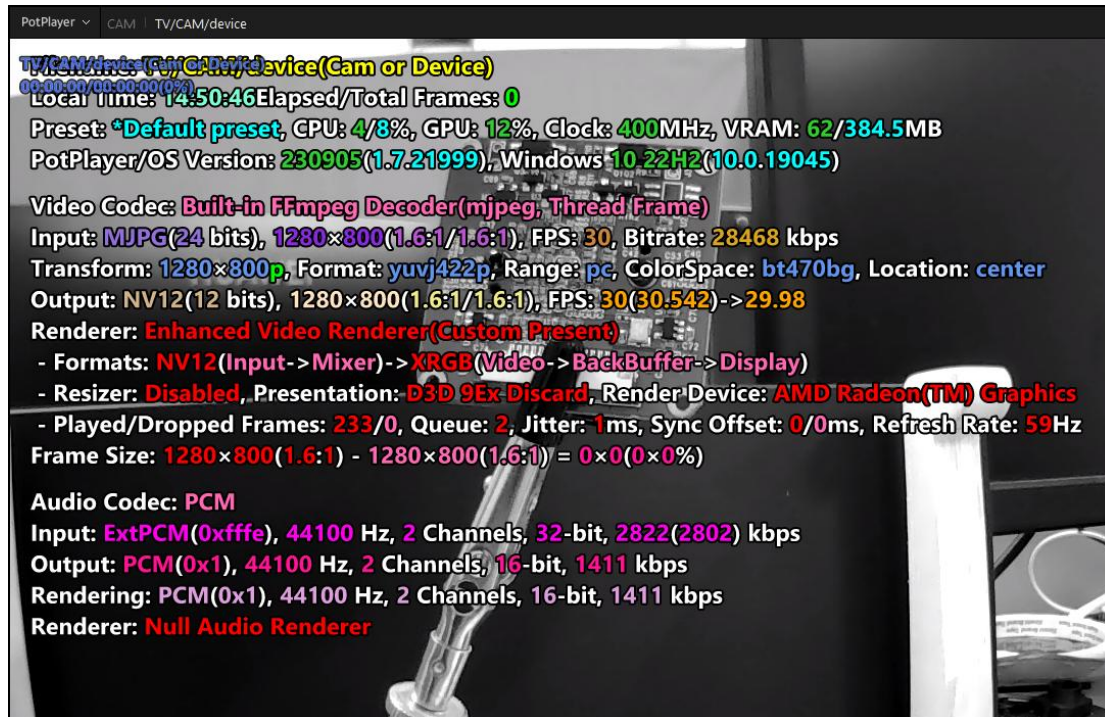
## UVC Camera SW Manual



1	Choose UVC Camera Device
2	Choose Output format ,resolution,frame rate
3	Camera Parameters Settings
4	Open Device

### 2.2.2 Live Working Status

Use shortcutkey TAB Open window as below



## 2.3 OpenCV Python

### 2.3.1 Install Python3

Download from below link, check from cmd.exe after install successfully

<https://www.python.org/downloads/release/>

```
python --version
```

```
pip --version
```

```
C:\Users\zhouj>python --version  
Python 3.11.6
```

```
C:\Users\zhouj>pip --version  
pip 23.3 from C:\Users\zhouj\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_x-ww\Python311\site-packages\pip (python 3.11)
```





### 2.3.2 Install numpy

```
pip install numpy
```

### 2.3.3 Install Opencv

```
pip install opencv-python
```

If you have error for installing, update your pip by below command:

```
python -m pip install --upgrade pip
```

### 2.3.4 Run OpenCV Python

Example1:

```
import cv2

cv2.namedWindow("preview")
vc = cv2.VideoCapture(0)

if vc.isOpened(): # try to get the first frame
    rval, frame = vc.read()
else:
    rval = False

while rval:
    cv2.imshow("preview", frame)
    rval, frame = vc.read()
    key = cv2.waitKey(20)
    if key == 27: # exit on ESC
        break

vc.release()
cv2.destroyAllWindows()
```



### Example2:

```
# import the opencv library
import cv2

# define a video capture object
vid = cv2.VideoCapture(0)

while(True):

    # Capture the video frame
    # by frame
    ret, frame = vid.read()

    # Display the resulting frame
    cv2.imshow('frame', frame)

    # the 'q' button is set as the
    # quitting button you may use any
    # desired button of your choice
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# After the loop release the cap object
vid.release()
# Destroy all the windows
cv2.destroyAllWindows()
```

## 2.3.5 Cited information

You can refer to the below link for any updates:

<https://stackoverflow.com/a/606154>

<https://www.geeksforgeeks.org/python-opencv-capture-video-from-camera/>



## 3 Works on Linux

### 3.1 Gvuvview

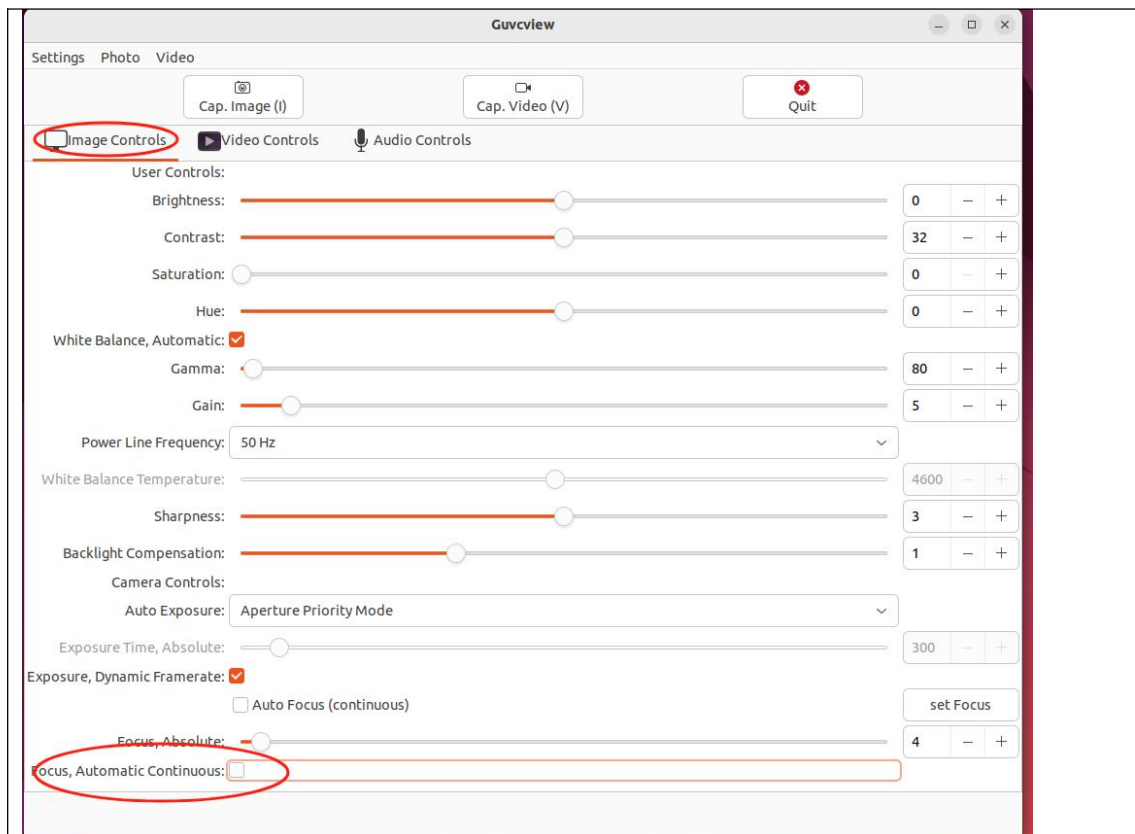
#### 3.1.1 Install

Gvuvview is free and easy operation tools for linux, Install and run :

```
sudo apt install guvview
```

```
sudo guvview
```

#### 3.1.2 Image Controls



You can find the control parameters from Image Controls.

#### 3.1.3 External Trigger Control

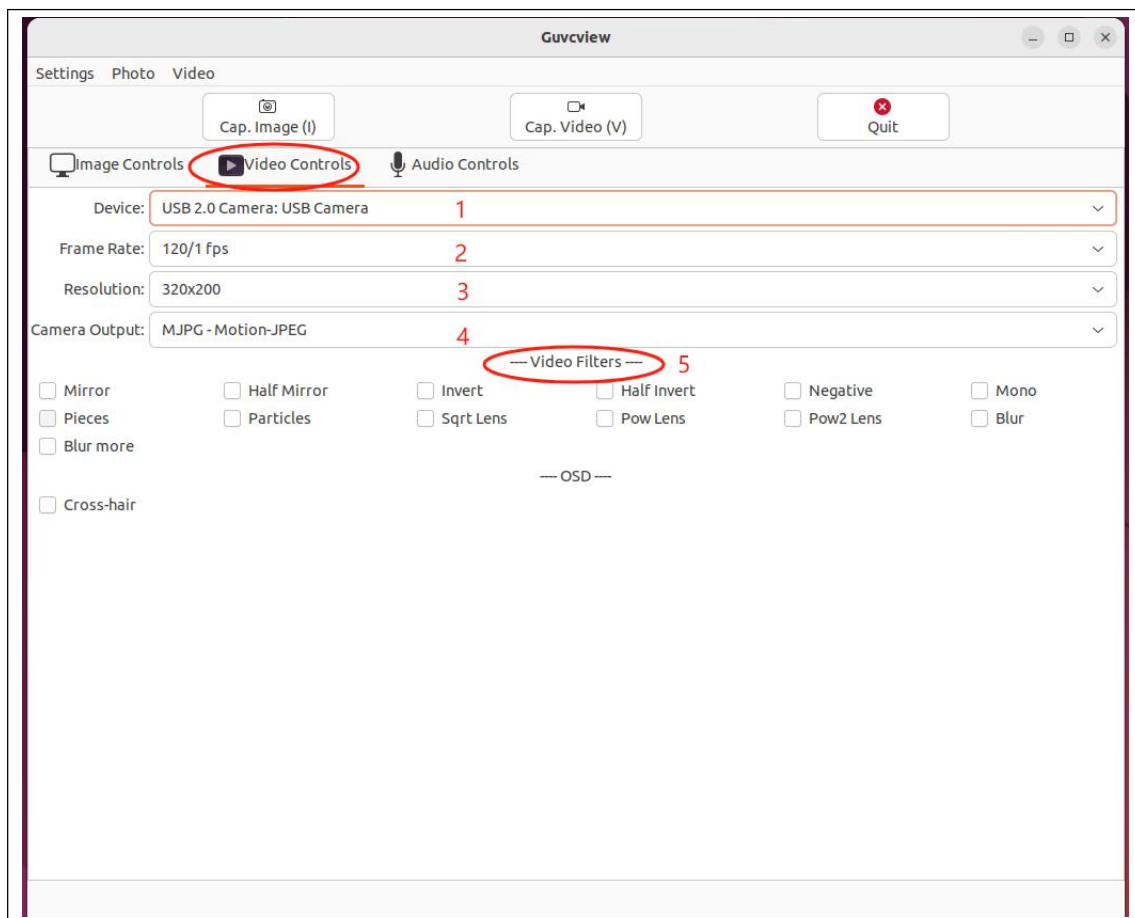


## UVC Camera SW Manual



Focus, Automatic Continuous is for external trigger. Uncheck it to enable external trigger mode.

### 3.1.4 Video Controls



From Video Controls,

1	Select Device
2	Select Frame Rate
3	Select Resolution
4	Select Output format
5	Video Filters

## 3.2 qv4l2



## UVC Camera SW Manual

### 3.2.1 Install

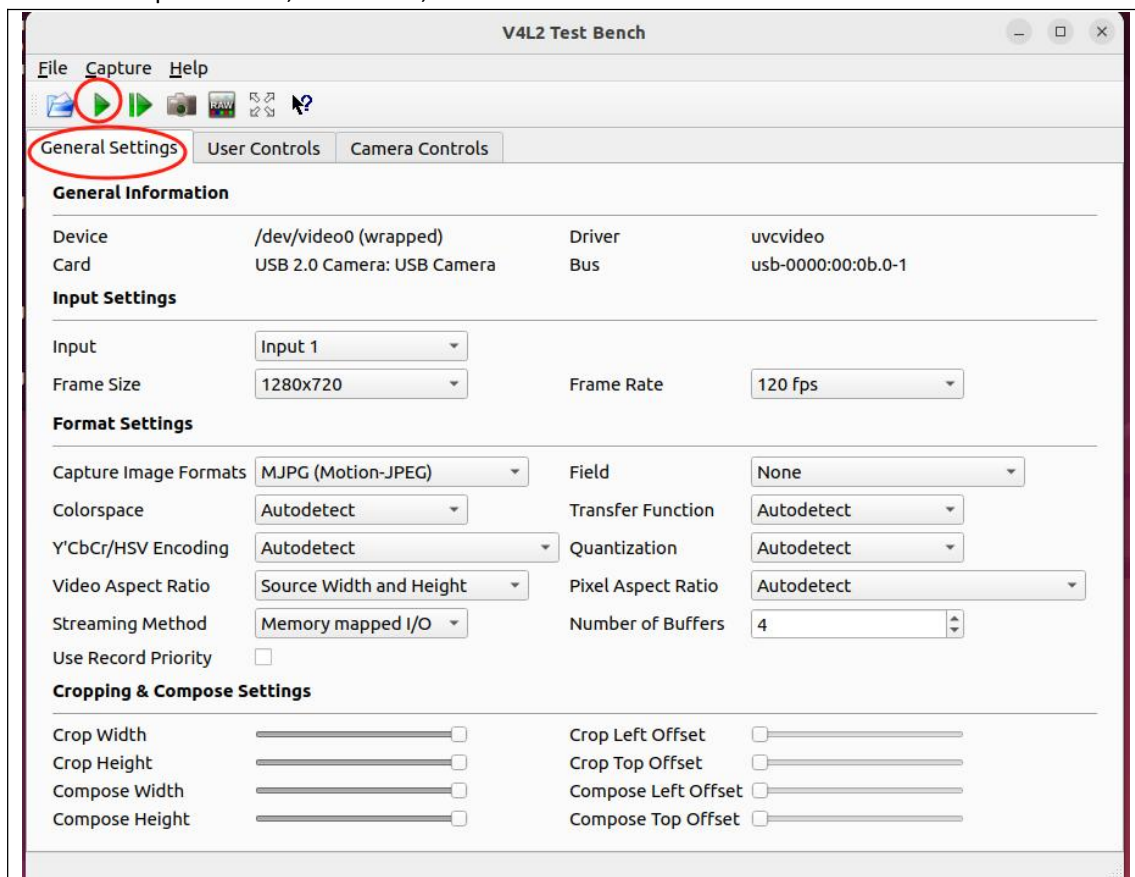
qv4l2 is free and easy operation tools for linux, Install and run :

```
sudo apt install qv4l2
```

```
sudo qv4l2
```

### 3.2.2 General Settings

Choose Output Devices, Resolution, Frame Rate

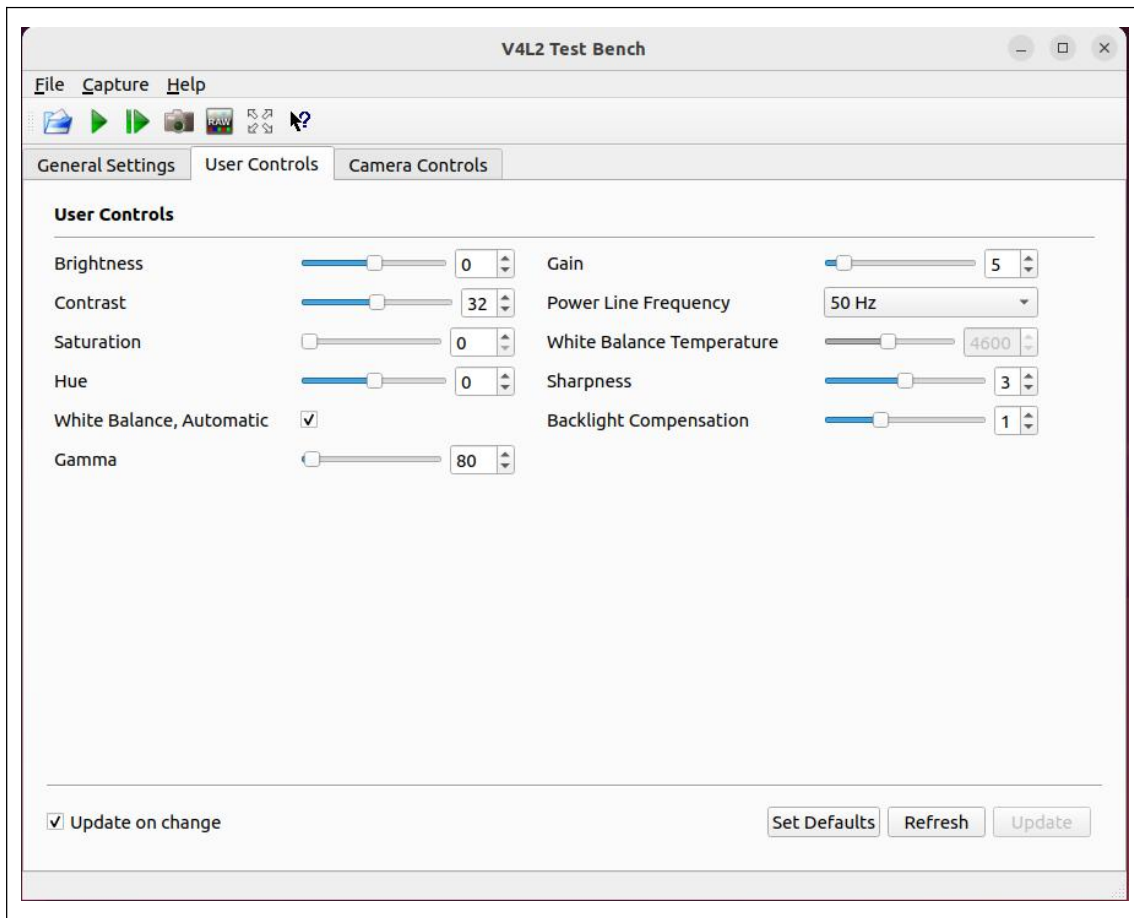


### 3.2.3 User Controls

control parameters

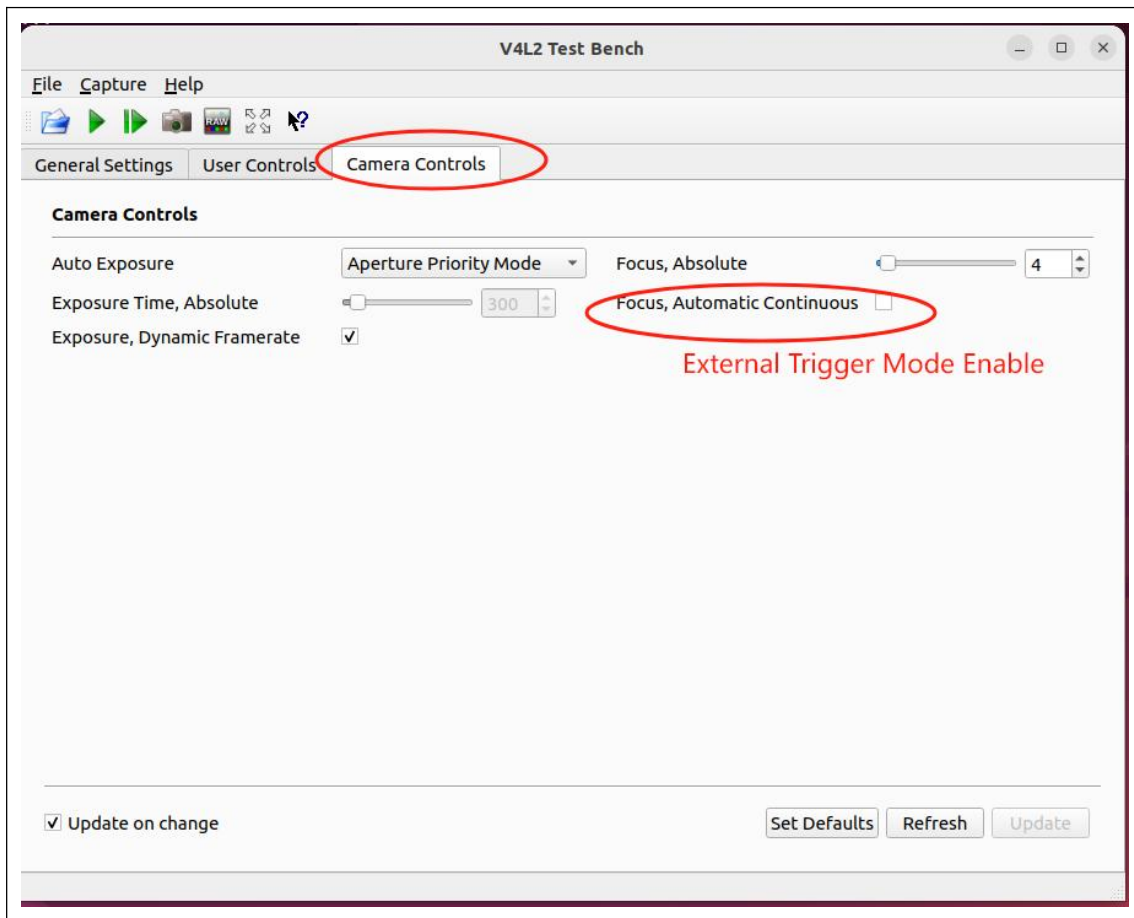


## UVC Camera SW Manual



### 3.2.4 Camera Controls

You can uncheck the External Trigger from this options.



## 3.3 V4L utility Tools

### 3.3.1 Install V4L utility packages

```
sudo apt-get update  
sudo apt-get install v4l-utils
```

### 3.3.2 List UVC devices

```
v4l2-ctl --list-devices  
  
joez@joez-VirtualBox:~$ v4l2-ctl --list-devices  
USB Camera: USB Camera (usb-0000:00:0c.0-2):  
    /dev/video0  
    /dev/video1  
    /dev/media0
```



### 3.3.3 List the supported formats

```
v4l2-ctl --list-formats -d
```

```
joez@joez-VirtualBox:~$ v4l2-ctl --list-formats -d 0
ioctl: VIDIOC_ENUM_FMT
Type: Video Capture

[0]: 'MJPG' (Motion-JPEG, compressed)
[1]: 'YUYV' (YUYV 4:2:2)
```

### 3.3.4 List resolutions and frame

```
v4l2-ctl --list-formats-ext -d 0
```

```
joez@joez-VirtualBox:~$ v4l2-ctl --list-formats-ext -d 0
ioctl: VIDIOC_ENUM_FMT
Type: Video Capture

[0]: 'MJPG' (Motion-JPEG, compressed)
    Size: Discrete 640x480
        Interval: Discrete 0.033s (30.000 fps)
        Interval: Discrete 0.033s (30.000 fps)
    Size: Discrete 800x600
        Interval: Discrete 0.033s (30.000 fps)
    Size: Discrete 1024x768
        Interval: Discrete 0.033s (30.000 fps)
    Size: Discrete 1280x720
        Interval: Discrete 0.033s (30.000 fps)
    Size: Discrete 1920x1080
        Interval: Discrete 0.033s (30.000 fps)
[1]: 'YUYV' (YUYV 4:2:2)
    Size: Discrete 1920x1080
        Interval: Discrete 0.200s (5.000 fps)
    Size: Discrete 640x480
        Interval: Discrete 0.033s (30.000 fps)
    Size: Discrete 800x600
        Interval: Discrete 0.050s (20.000 fps)
        Interval: Discrete 0.067s (15.000 fps)
        Interval: Discrete 0.100s (10.000 fps)
        Interval: Discrete 0.200s (5.000 fps)
    Size: Discrete 1024x768
        Interval: Discrete 0.200s (5.000 fps)
    Size: Discrete 1280x720
        Interval: Discrete 0.100s (10.000 fps)
        Interval: Discrete 0.200s (5.000 fps)
    Size: Discrete 1280x1024
        Interval: Discrete 0.200s (5.000 fps)
```

### 3.3.5 List Control parameters





```
v4l2-ctl -d /dev/video0 -list
```

```
joez@joez-VirtualBox:~$ v4l2-ctl -d /dev/video0 -list
Video input set to 0 (Input 1: Camera, ok)

User Controls

          brightness 0x00980900 (int)      : min=-64 ma
          contrast 0x00980901 (int)      : min=0 max=
          saturation 0x00980902 (int)      : min=0 max=
          hue 0x00980903 (int)           : min=-180 m
    white_balance_automatic 0x0098090c (bool) : default=1
          gamma 0x00980910 (int)         : min=100 ma
          gain 0x00980913 (int)          : min=1 max=
    power_line_frequency 0x00980918 (menu) : min=0 max=
    white_balance_temperature 0x0098091a (int) : min=2800 m
          sharpness 0x0098091b (int)      : min=0 max=
    backlight_compensation 0x0098091c (int) : min=0 max=

Camera Controls

          auto_exposure 0x009a0901 (menu)  : min=0 max=
    exposure_time_absolute 0x009a0902 (int) : min=50 max=
    exposure_dynamic_framerate 0x009a0903 (bool) : default=0
```

### 3.3.6 Set User/Camera controls

For example, set camera brightness to 64

```
v4l2-ctl -d /dev/video0 --set-ctrl=brightness=64
```

```
joez@joez-VirtualBox:~$ v4l2-ctl -d /dev/video0 --set-ctrl=brightness=64
```

## 3.4 OpenCV Python

### 3.4.1 Install Opencv-Python

Check python pip version



## UVC Camera SW Manual

```
python3 --version
```

```
pip --version
```

Run below command if not find the pip.

```
joez@joez-VirtualBox:~$ pip --version
Command 'pip' not found, but can be installed with:
sudo apt install python3-pip
```

```
sudo apt install python3-pip
```

Install opencv-python

```
sudo pip install OpenCV-python
```

**\* If you en count download errors**

```
sudo pip install opencv-python -i https://pypi.tuna.tsinghua.edu.cn/simple
```

### 3.4.2 Set user controls parameters.

Below code sample set brightness as 64, contrast as 0

```
import cv2

# open video0
cap = cv2.VideoCapture(0)

# The control range can be viewed through v4l2-ctl -L
cap.set(cv2.CAP_PROP_BRIGHTNESS, 64)
cap.set(cv2.CAP_PROP_CONTRAST, 0)

while(True):
    # Capture frame-by-frame
    ret, frame = cap.read()
    # Display the resulting frame
    cv2.imshow('frame', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()
```



**SAVE File name as 1.py, then run**

```
sudo python3 1.py
```

### 3.4.3 Controlling values through code

```
import cv2
import time
# open video0
cap = cv2.VideoCapture(0)
cap.grab()

cap.set(cv2.CAP_PROP_AUTOFOCUS, 1)
time.sleep(2)
cap.set(cv2.CAP_PROP_AUTOFOCUS, 0)
time.sleep(2)
cap.set(cv2.CAP_PROP_FOCUS, 123)

cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)

while(True):
    # Capture frame-by-frame
    ret, frame = cap.read()
    # Display the resulting frame
    cv2.imshow('frame', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()
```

**SAVE File name as 2.py, then run**



```
sudo python3 2.py
```

### 3.4.4 Controlling values through UI interface

```
import cv2
import argparse
import configparser
from pathlib import Path
import time

parser = argparse.ArgumentParser()
parser.add_argument("-v", "--vid", default="0", help="Video source, default 0")
parser.add_argument(
    "-f", "--auto_focus", action="store_true", default=False, help="Turn on auto focus"
)
parser.add_argument(
    "-c",
    "--config",
    default="focus.ini",
    help="Focus config file, default focus.ini",
)
args = parser.parse_args()

try:
    vid = int(args.vid)
except ValueError:
    vid = args.vid

config_path = (Path(__file__).parent / Path(args.config)).resolve().absolute()
print("config file :", config_path)

config = configparser.ConfigParser()

config.read(config_path, encoding="utf-8")

cap = cv2.VideoCapture(vid)
cap.grab()
cap.set(cv2.CAP_PROP_AUTOFOCUS, 1)
```



## UVC Camera SW Manual

```
if not args.auto_focus and config.has_section("Focus"):
```

```
    auto_focus = (  
        config.getint("Focus", "auto_focus")  
        if config.has_option("Focus", "auto_focus")  
        else 1  
    )  
    focus = (  
        config.getint("Focus", "focus")  
        if config.has_option("Focus", "focus")  
        else int(cap.get(cv2.CAP_PROP_FOCUS))  
    )
```

```
else:
```

```
    auto_focus = 1  
    focus = None
```

```
print("config auto_focus = %s" % auto_focus)  
print("config focus = %s" % focus)  
print("*" * 10)
```

```
if not auto_focus:
```

```
    cap.set(cv2.CAP_PROP_AUTOFOCUS, 0)
```

```
time.sleep(2)
```

```
if focus:
```

```
    cap.set(cv2.CAP_PROP_FOCUS, focus)
```

```
cv2.namedWindow("frame")
```

```
def set_auto_focus(x):
```

```
    cap.set(cv2.CAP_PROP_AUTOFOCUS, x)
```

```
cv2.createTrackbar(  
    "0: OFF\r\n 1: ON\r\nauto_focus",  
    "frame",  
    int(cap.get(cv2.CAP_PROP_AUTOFOCUS)),  
    1,  
    set_auto_focus,  
)
```



## UVC Camera SW Manual

```
def set_focus(x):
    cap.set(cv2.CAP_PROP_FOCUS, x)

cv2.createTrackbar("focus", "frame", int(cap.get(cv2.CAP_PROP_FOCUS)), 1023, set_focus)

while cap.isOpened():
    # cap frame-by-frame
    ret, frame = cap.read()
    if not ret:
        break
    focus = int(cap.get(cv2.CAP_PROP_FOCUS))
    cv2.setTrackbarPos("focus", "frame", focus)

    af = int(cap.get(cv2.CAP_PROP_AUTOFOCUS))
    cv2.setTrackbarPos("0: OFF\r\n 1: ON\r\nauto_focus", "frame", af)

    cv2.imshow("frame", frame)

    if cv2.waitKey(1) & 0xFF == ord("q"):
        break

# When everything done, release the cap
cap.release()
cv2.destroyAllWindows()

if not config.has_section("Focus"):
    config.add_section("Focus")

print("set auto_focus = 0")
config.set("Focus", "auto_focus", "0")

print("set focus = %s" % focus)
config.set("Focus", "focus", str(focus))

config.write(open(config_path, "w"))
```

**SAVE File name as cvtui.py, then run**

```
sudo python3 cvtui.py
```



### 3.5 Gstreamer

GStreamer becomes a popular and powerful open-source multimedia framework to help users to build their own video streaming, playback, editing applications with various codec and functionalities on top of its high-level APIs.

#### 3.5.1 Set Video Output Format

##### MJPEG

```
gst-launch-1.0 v4l2src device=/dev/video0 ! \
    image/jpeg,width=1920,height=1080,framerate=30/1 ! \
    decodebin ! autovideosink
```

```
joez@joez-VirtualBox:~/Desktop$ gst-launch-1.0 v4l2src device=/dev/video0 ! \
    image/jpeg,width=1920,height=1080,framerate=30/1 ! \
    decodebin ! autovideosink
```

##### YUV

```
gst-launch-1.0 -vv v4l2src device=/dev/video0 ! \
    video/x-raw,format=YUY2,width=1280,height=720,framerate=10/1 ! \
    videoconvert ! autovideosink
```

```
joez@joez-VirtualBox:~/Desktop$ gst-launch-1.0 -vv v4l2src device=/dev/video0 ! \
    video/x-raw,format=YUY2,width=1280,height=720,framerate=10/1 ! \
    videoconvert ! autovideosink
```

#### 3.5.2 Streaming

##### MJPEG

```
# server
gst-launch-1.0 v4l2src device=/dev/video0 ! \
    image/jpeg,width=1280,height=720,framerate=30/1 ! \
    tcpserver sink host=0.0.0.0 port=5001

# client
# change xxx.xxx.xxx.xxx to the actual ip address
```



## UVC Camera SW Manual

```
gst-launch-1.0 -v tcpclientsrc host=xxx.xxx.xxx.xxx port=5001 ! \
    decodebin ! autovideosink
```

### Save Video

```
gst-launch-1.0 v4l2src device=/dev/video0 !
image/jpeg,width=1280,height=720,framerate=30/1 ! jpegdec ! qtmux ! filesink
location=test.mp4 -e
```

### Save Image

```
gst-launch-1.0 v4l2src device=/dev/video0 num-buffers=1 ! jpegenc ! filesink sync=false
location=file.jpg
```

### Preview

```
gst-launch-1.0 v4l2src device=/dev/video0 !
image/jpeg,width=1280,height=720,framerate=30/1 ! jpegdec ! autovideosink
```

## 3.6 Read Serial Number

When you need to use multiple cameras, we need to use unique serial ID.

### 3.6.1 Linux udev

```
sudo udevadm info --query=all /dev/video0 | grep 'VENDOR_ID\|MODEL_ID\|SERIAL_SHORT'
```

```
joez@joez-VirtualBox: ~/Desktop$ sudo udevadm info --query=all /dev/video0 | grep
'VENDOR_ID\|MODEL_ID\|SERIAL_SHORT'
E: ID_VENDOR_ID=0bda
E: ID_MODEL_ID=3035
E: ID_SERIAL_SHORT=200901010001
```