云实验室摄像头应用案例

本文档涉及案例包括：

- 相机信息检测
- 相机显示
- 双相机并列显示
- 双相机叠加显示
- 相机内容的本地保存
- 修改相机输出分辨率
- 相机内容的旋转
- 相机输出的元素空间转换
- USB 相机显示
- 相机+NPU 案例

执行步骤：

1 预定板子：

打开云实验室网页，点击右上角登陆按钮输入账号密码。

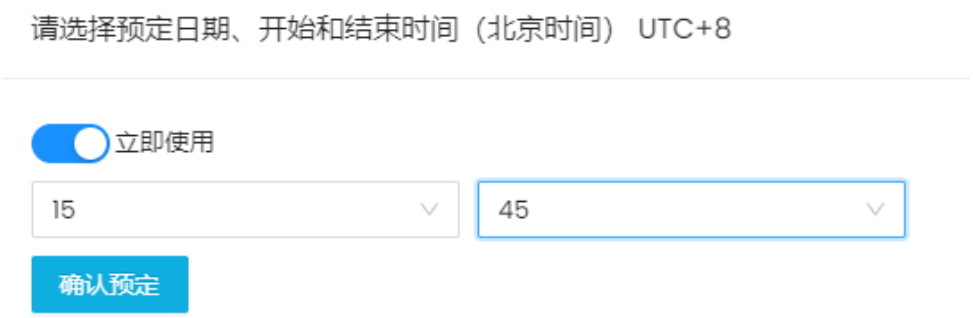
```
https://aiotcloud.nxp.com.cn/
```

登陆后依次点击硬件 -> i.MX 8 系列开发板

找到 i.MX 8M Plus “可立即使用” 状态的板子，点击“8MPLUSLPD4-PEVK“进入。

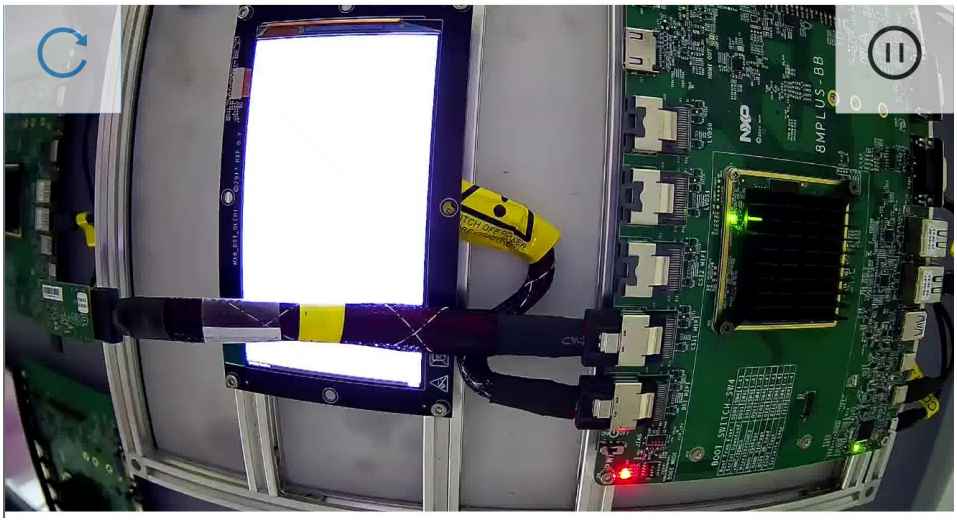
然后点击“立即预定“按钮：

选择立即使用，填写使用结束时间



然后进入我的预定，

点击右侧蓝色“调试”按钮，之后就进入到板子的实物页面和系统启动 log 页面。到此为止，板子预定并且启动成功。



默认用户名为“root”，默认无密码。

2 案例执行

2.1 相机信息检测

通过 v4l2-ctl 可以列出当前摄像头所能够支持的所有输出格式，比如对于 ov5640，命令和打印出可以支持的输出格 log 如下

```
v4l2-ctl --list-formats -d /dev/video3
```

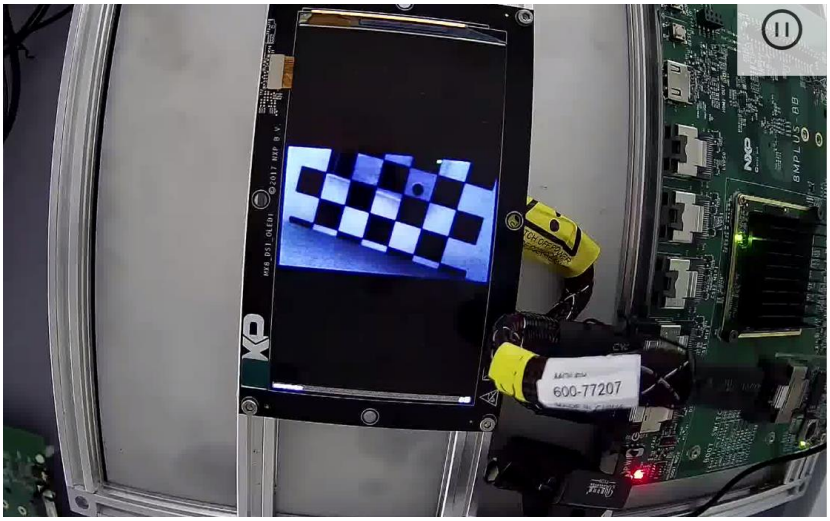
```
root@imx8mpevk:~# v4l2-ctl --list-formats -d /dev/video3
ioctl: VIDIOC_ENUM_FMT
        Type: Video Capture Multiplanar

        [0]: 'RGBP' (16-bit RGB 5-6-5)
        [1]: 'RGB3' (24-bit RGB 8-8-8)
        [2]: 'BGR3' (24-bit BGR 8-8-8)
        [3]: 'YUYV' (YUYV 4:2:2)
        [4]: 'YUV4' (32-bit A/XYUV 8-8-8-8)
        [5]: 'NV12' (Y/UV 4:2:0)
        [6]: 'NM12' (Y/UV 4:2:0 (N-C))
        [7]: 'YM24' (Planar YUV 4:4:4 (N-C))
        [8]: 'XR24' (32-bit BGRX 8-8-8-8)
        [9]: 'AR24' (32-bit BGRA 8-8-8-8)
```

2.2 相机显示

如下的 pipeline 可以被用来 preview 相机。v4l2src 的参数 device 指定相机的挂载设备，相机输出的格式和分辨率则通过 format，width 和 height 参数进行指定，最终输出到 wayland 桌面

```
   gst-launch-1.0 v4l2src device=/dev/video3 ! video/x-
raw,format=NV12,width=1920,height=1080 ! queue !  waylandsink
enable-tile=true sync=false
```

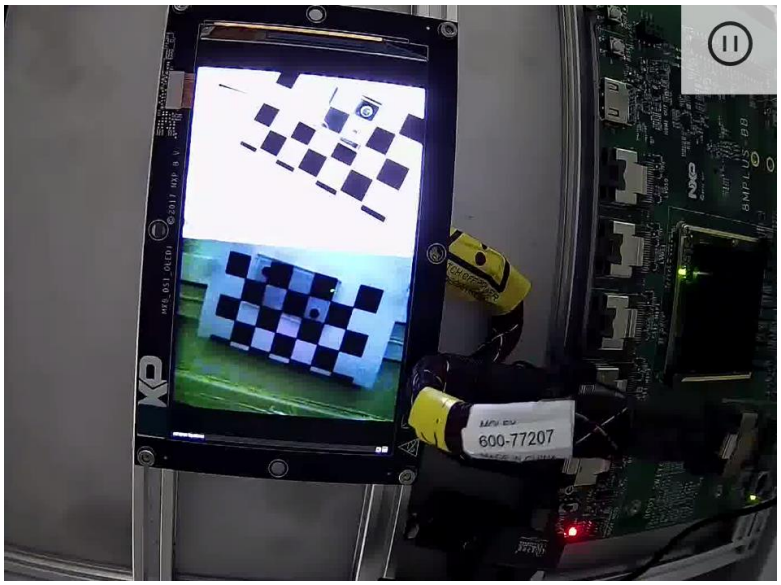


2.3 双相机并列显示

视频输入包含两路输入，一路为 ov5640 相机（/dev/video3，经由 MIPI-CSI 接口连接板子），另一路为 USB 相机（/dev/video4）。ov5640 相机使用 sink_0，其左上角的坐标为（0，0），USB 相机使用 sink_1，左上角位置为(0,480)。两路输入使用插件 imxcompositor_g2d，经由 2d gpu 叠加后上下并列放置，pipeline 和图像如下图所示

```
   gst-launch-1.0 imxcompositor_g2d name=comp \
sink_0::xpos=0 sink_0::ypos=0 sink_0::width=640 sink_0::height=480
\
```

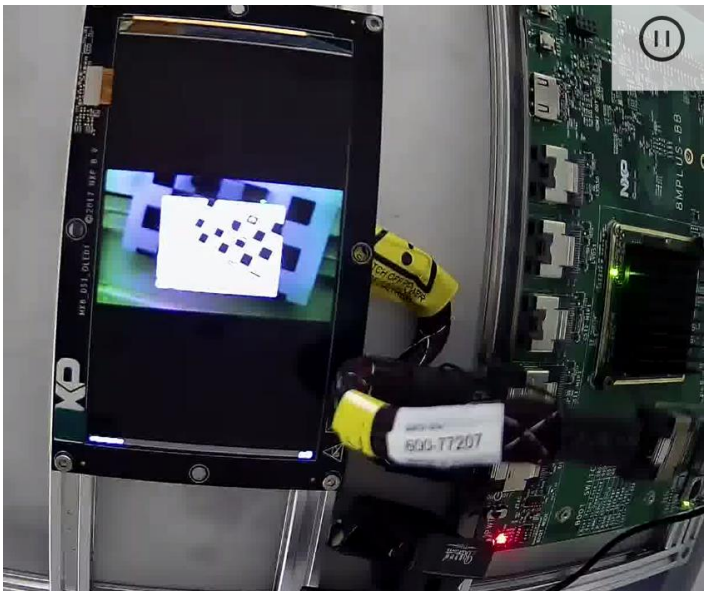
```
sink_1::xpos=0 sink_1::ypos=480 sink_1::width=640
sink_1::height=480 ! \
waylandsink \
v4l2src device=/dev/video3 ! video/x-
raw,format=NV12,width=640,height=480 ! queue ! comp.sink_0 \
v4l2src device=/dev/video4 ! video/x-
raw,format=YUY2,width=640,height=480 ! queue ! comp.sink_1
```



2.4 双相机叠加显示

视频输入包含两路输入，一路为 ov5640 相机（/dev/video3，经由 MIPI-CSI 接口连接板子），另一路为 USB 相机（/dev/video4）。ov5640 相机使用 sink_0，其左上角的坐标为（0，0），分辨率为 1280x720；USB 相机使用 sink_1，左上角位置为(320,120)，分辨率为 640x480。两路输入使用插件 imxcompositor_g2d，经由 2d gpu 叠加，其中 sink_0 图层位于下层（zorder=1），sink_1 图层位于上层（zorder=2），pipeline 和输出图像如图所示：

```
   gst-launch-1.0 -v imxcompositor_g2d name=comp sink_0::zorder=1
sink_1::zorder=2 \
sink_0::xpos=0 sink_0::ypos=0 sink_0::width=1280 sink_0::height=720
\
sink_1::xpos=320 sink_1::ypos=120 sink_1::width=640
sink_1::height=480 ! \
waylandsink \
v4l2src device=/dev/video3 ! video/x-
raw,format=NV12,width=1280,height=720! queue ! comp.sink_0 \
v4l2src device=/dev/video4 ! video/x-
raw,format=YUY2,width=640,height=480 ! queue ! comp.sink_1
```



2.5 相机内容的本地保存

方法一为 gstreamer pipeline 的方式，借助 filesink 插件，并将 location 参数配置成输出文件的名称，pipeline 如下：

```
    gst-launch-1.0 v4l2src device=/dev/video3 ! video/x-
raw,format=NV12,width=1280,height=720 ! queue ! filesink
location=output.raw
```

方法二为直接借助 v4l2-ctl，“-d”参数指定相机的设备号，“--stream-to”指定输出的文件名，“--stream-count”可以指定保存的帧数。命令如下：

```
    v4l2-ctl --set-fmt-video=width=640,height=480,pixelformat=YUY2
--stream-mmap -d  /dev/video4 --stream-to=output.raw --stream-
count=10
```

log 如下，可以看到文件被保存到了 output.raw 中。

```
root@imx8mpevk:~# gst-launch-1.0 v4l2src device=/dev/video3 ! video/x-raw,format=NV12,width=1280,hei
ght=720 ! queue ! filesink location=output.raw
Setting pipeline to PAUSED ...
Pipeline is live and does not need PREROLL ...
Pipeline is PREROLLED ...
Setting pipeline to PLAYING ...
New clock: GstSystemClock
[  184.568368] bypass csc
[  184.570755] input fmt YUV4
[  184.573463] output fmt NM12
Redistribute latency...
^Chandling interrupt.
Interrupt: Stopping pipeline ...
Execution ended after 0:00:04.790558125
Setting pipeline to NULL ...
Freeing pipeline ...
root@imx8mpevk:~# ls
gstshark_2023-03-03_09:51:05  output.raw
```


2.6 修改相机输出分辨率

更改相机的分辨率包含两种情况。

在第一种情况下，我们期待的分辨率，如 640x480，对于 ov5640 相机而言是默认可以支持的，此时我们只需要将紧跟 v4l2src 后的 width 和 height 参数配置为目标分辨率即可。具体 pipeline 如下：

```
gst-launch-1.0 v4l2src device=/dev/video3 ! video/x-raw,format=NV12,width=640,height=480 ! queue ! waylandsink enable-tile=true sync=false
```

如果目标分辨率，如 100x100，对于 ov5640 相机而言无法直接输出，我们则需要借助插件 imxvideoconvert_g2d，使用 2d gpu 对相机的输入进行 resize，此时需要将紧跟在 imxvideoconvert_g2d 后的参数 width 和 height 配置成我们需要的分辨率即可。具体 pipeline 如下：

```
gst-launch-1.0 v4l2src device=/dev/video3 ! video/x-raw,format=NV12,width=640,height=480 ! imxvideoconvert_g2d ! video/x-raw,width=100,height=100 !queue ! waylandsink enable-tile=true sync=false
```

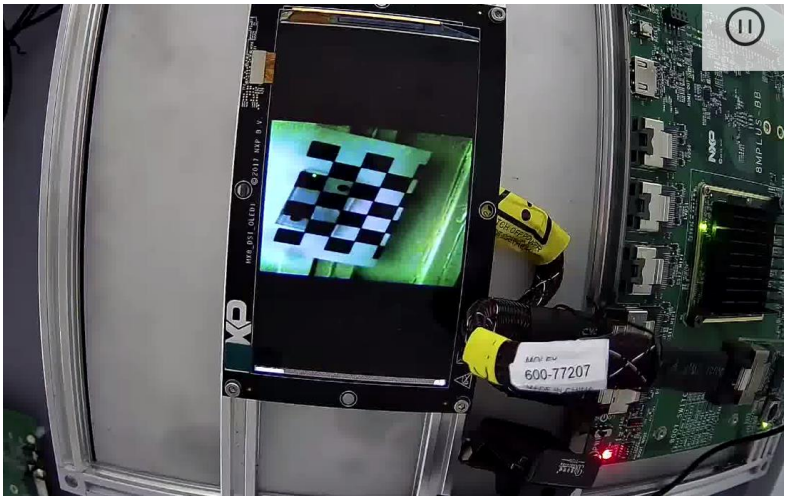
2.7 相机内容的旋转

imxvideoconvert_g2d 插件的参数 rotation 可以对相机的输入进行旋转操作。其具体参数的意义如下

```
rotation            : Rotation that shall be applied to output frames
                      flags: readable, writable
                      Enum "ImxVideoConvertRotationMode" Default: 0, "none"
                         (0): none             - No rotation
                         (1): rotate-90        - Rotate 90 degrees
                         (2): rotate-180       - Rotate 180 degrees
                         (3): rotate-270       - Rotate 270 degrees
                         (4): horizontal-flip  - Flip horizontally
                         (5): vertical-flip    - Flip vertically
```

当我们需要对相机输入进行 270°旋转时，具体 pipeline 如下：

```
gst-launch-1.0 v4l2src device=/dev/video3 ! video/x-raw,format=NV12,width=640,height=480 ! queue ! imxvideoconvert_g2d rotation=3 ! queue ! waylandsink enable-tile=true sync=false
```



2.8 相机输出的元素空间转换

常见的 CSC 包含两种。第一种是借助 ISI(Image Sensor Interface)进行 CSC，这种只局限于 MIPI-CSI 接口的相机。将紧跟在 v4l2src 后的 format 参数配置成我们需要的格式即可。具体 pipeline 为：

```
   gst-launch-1.0 v4l2src device=/dev/video3 ! video/x-
raw,format=RGB16,width=1280,height=720 !  waylandsink enable-
tile=true sync=false
```

在 ISI 不支持目标转换的情况下，需要借助第二种，插件 imxvideoconvert_g2d 即 2d GPU 的方式进行 CSC。需要将紧跟在 imxvideoconvert_g2d 后的参数 format 配置成我们需要的格式即可。具体 pipeline 为：

```
   gst-launch-1.0 v4l2src device=/dev/video3 ! video/x-
raw,format=NV12,width=1280,height=720 !  queue max-size-
buffers=0 ! imxvideoconvert_g2d ! video/x-
raw,format=RGB16,width=1280,height=720 !  queue ! waylandsink
enable-tile=true sync=false
```

2.9 USB 相机显示

USB 相机一般也被挂载在/dev 目录下。一般选择使用如下命令检测 USB 相机可以支持的输出类型

```
   v4l2-ctl --list-formats -d /dev/video4
```

可能得到的 log 如下：

```
root@imx8mpevk:~# v4l2-ctl --list-formats -d /dev/video4
ioctl: VIDIOC_ENUM_FMT
        Type: Video Capture

        [0]: 'YUYV' (YUYV 4:2:2)
```

即表示该 USB 相机只支持 YUYV 格式，该格式在 gstreamer 中被表示为 YUV2。因此 preview 该相机的具体 pipeline 为：

```
   gst-launch-1.0 v4l2src device=/dev/video4 ! video/x-
raw,format=YUY2,width=640,height=480 ! queue !  waylandsink enable-
tile=true sync=false
```

对于某些支持 MJPG 格式的相机，则需要使用 jpegdec 插件进行 JPEG 格式的解码。具体 pipeline 为：

```
   gst-launch-1.0 v4l2src device=/dev/video4 ! jpegdec!
imxvideoconvert_g2d !  waylandsink
```

2.10 相机+NPU 案例

在新版本的 Linux BSP 中，用户可以通过点击左上角的 NXP logo 来测试 object detection

demo。同时用户也可以通过如下命令对该 demo 进行拆解。首先需要在本地下载一些模型文件，Windows 操作系统下可以直接复制链接地址到浏览器中：

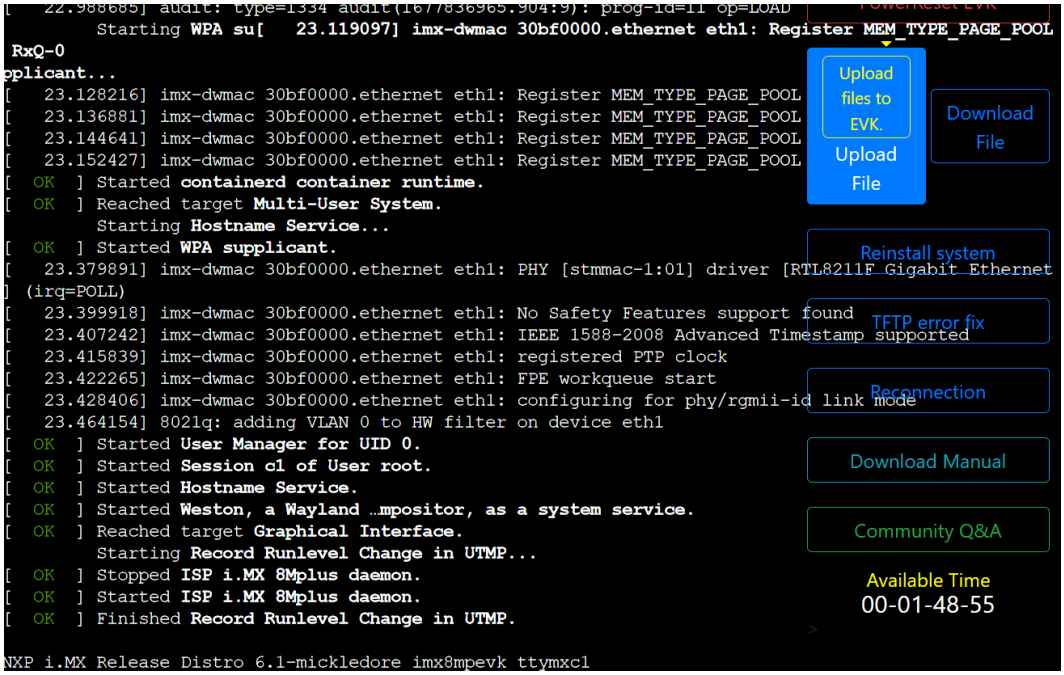
文件 1：

```
   https://github.com/google-
coral/test_data/raw/master/ssd_mobilenet_v2_coco_quant_postprocess
.tflite
```

文件 2：

```
   https://github.com/google-
coral/test_data/raw/master/coco_labels.txt
```

下载到本地 PC 后，通过右侧的 Upload File 按钮，上传至板子



随后在 Linux console 中，使用 export 命令，为 MODEL 和 LABELS 两个系统变量赋值：

```
   export
MODEL=$(pwd)/ssd_mobilenet_v2_coco_quant_postprocess.tflite \
   export LABELS=$(pwd)/coco_labels.txt
```

最后输入如下 pipeline 执行 object detection demo：

```
   gst-launch-1.0 --no-position v4l2src device=/dev/video3 ! \
video/x-raw,width=640,height=480,framerate=30/1 ! \
tee name=t t. ! queue max-size-buffers=2 leaky=2 ! \
imxvideoconvert_g2d ! \
video/x-raw,width=300,height=300,format=RGBA ! \
videoconvert ! video/x-raw,format=RGB ! \
```

```
tensor_converter ! \
tensor_filter framework=tensorflow-lite model=${MODEL}
custom=Delegate:NNAPI ! \
tensor_decoder mode=bounding_boxes option1=tf-ssd
option2=${LABELS} \
option3=0:1:2:3,50 option4=640:480 option5=300:300 ! \
mix. t. ! queue max-size-buffers=2 ! \
imxcompositor_g2d name=mix sink_0::zorder=2 sink_1::zorder=1 !
waylandsink
```