**V4L2 Deep Learning**

V4L2 是 Video for Linux Two 的简称。是Video for Linux API的第二个版本。 提供为音频设备和实现image、video的捕获的视频设备的driver的接口。就是为radio, camera以及他们的功能对应的driver提供一个内核的接口。

早期的视频设备使用ad-hoc接口。后来被Alan Cox的V4L API取代，应用与Linux2.2 kernel中。在1999年，Bill Dirks启动V4L2的开发并且修正了部分V4L的代码，能够支持更多设备。后续V4L2的API也被多次的修正，兼容Linux更高版本的kernel。

V4L2的源码也是开源的，可以从 <git://linuxtv.org/v4l-utils.git> clone下来。

V4L2对一个设备的操作流程包括：

1. Opening the device
2. Changing device properties, selecting a video and audio input, video standard, picture brightness and so on.
3. Negotiating a data format.
4. Negotiating an input/output method.
5. Actual input/output loop.
6. Closing the device.

实际操作中，对不同的设备而言，部分上述步骤可以省略。

***打开和关闭设备***

V4L2 drivers作为模块插入在内核中。当一个外接设备第一次被打开时，V4L2模块会被自动加载。插入外接设备时，系统会为每个设备命名。**每个设备作为一个节点挂载在/dev/下，driver就会把不同节点对应的寄存器都load进来。**系统为每个节点进行编号命名，major No + minor No，major No为81，minor No为0-255之间的一个数。命名完全由系统决定。V4L2 driver模块会选择以’\_nr’作为后缀命名的设备。

例如：

insmod mydriver.o video\_nr=0,1 radio\_nr=0,1

那么在/etc/modules.conf中就要做出如下定义：

alias char-major-81-0 mydriver

alias char-major-81-1 mydriver

alias char-major-81-64 mydriver

options mydriver video\_nr=0,1 radio\_nr=0,1

当一个APP试图打开编号为81-0, 或者81-1或者81-64的设备时，就会自动load ‘mydriver’这个驱动。

V4L2设备可以被多次打开。如果驱动支持多次打开设备的话，那么就可以启动类似于面板的APP来控制亮度，音量等。

打开设备和关闭设备的函数为 open() close()

***查询功能***

V4L2覆盖了很多设备，但是即使对于同一种设备，也有着不同的功能，因此可以用一个ioctl VIDIOC\_QUERYCAP来检查kernel驱动是否与当前设备兼容。所有的V4L2的驱动都必须支持VIDIOC\_QEURYCAP这个ioctl，一个应用在打开设备以后应当首先调用这个ioctl。Ioctl()函数是设备驱动程序中对设备的I/O通道进行管理的函数。其中通过调用不同的命令，对设备的特性进行控制，例如波特率，转速，等等。

***Video Inputs and Outputs***

Video的输入和输出是设备的物理接口，radio设备没有video inputs and outputs。在代码中，可以用VIDIOC\_ENUMINPUT和VIDIOC\_ENUMOUTPUT这两个ioctl命令来枚举设备可用的inputs和outputs。只有video和VBI设备有输入，输出设备至少有一个输出。

***Audio Inputs and Outputs***

音频的输入和输出也是设备的物理接口，视频捕获设备有audio inputs，输出设备可有或没有outputs，radio设备既没有音频输入，也没有输出。

***视频标准***

视频设备通常支持多种视频标准。每一个视频的input和output通道都可能支持不同的标准。这些标准可以由VIDIOC\_ENUMINPUT和VIDIOC\_ENUMOUTPUT这两个ioctl命令进行查看。

V4L2为每一种现有的视频标准定义一个bit位。剩余的bit位用于标识驱动标准。可以用VIDIOC\_ENUMSTD ioctl查看设备支持的视频标准。

***用户控制接口***

用户在使用不同的设备时，会进行亮度，饱和度或者音量等的调节，因此，对不同的设备来说，需要为用户提供不同的控制交互界面。设备不同，同一个参数也会有不同的默认值和范围。因此，需要通过ioctl来发送不同的命令，获取不同设备的参数属性。这些都是通过不同的ID来获取的。V4L2中定义了不同的ID，用户也可以根据自己的需求，植入自己的ID。所有的ID都以V4L2\_CID\_作为前缀。这些ID可以帮助用户设置或者读取不同属性的当前值。

一个通用APP，必须向用户提供所有的可控制的ID。命名清晰。如果命名不能直接反应该ID的用途，则必须配有相关的user manual或者一个控制面板。例如下表：

|  |  |  |
| --- | --- | --- |
| ID | Type | Description |
| V4L2\_CID\_BASE |  | First predefined ID, equal to V4L2\_CID\_BRIGHTNESS |
| V4L2\_CID\_USER\_BASE |  | Synonym of V4L2\_CID\_BASE |
| V4L2\_CID\_BRIGHTNESS | integer | Picture brightness, or more precisely, the black level |
| V4L2\_CID\_CONTRAST | integer | Picture contrast or luma gain |
| V4L2\_CID\_SATURATION | integer | Picture color saturation or chroma gain |
| V4L2\_CID\_HUE | integer | Hue or color balance |

用户可以通过在应用中调用VIDIOC\_QUERYCTRL和VIDIOC\_QUERYMENU命令，枚举出可用的控制ID。通过VIDIOC\_G\_CTRL和VIDIOC\_G\_CTRL来设置或者读取一个ID对应的值。驱动中必须支持VIDIOC\_QUERYCTRL, VIDIOC\_G\_CTRL 和VIDIOC\_S\_CTRL命令的调用。

***数据格式***

***Data Format Begotiation***

不同的应用与设备之间需要交换不同的数据。比如说image,raw,或者VBI的数据等。尽管设备的驱动都会指定一个默认的数据格式，但是在每次与应用之间交换数据时，仍然需要进行数据格式的协商。协商意味着，应用程序向设备请求一个特殊的指定的数据格式，驱动控制硬件设备，尽量产生满足需求的数据格式。当然，应用程序也可以直接询问驱动当前选择的数据格式。

v4l2\_format这个结构体中，定义了现有的全部数据格式。通过VIDIOC\_G\_FMT和VIDIOC\_S\_FMT命令进行设置和查询。另外，VIDIOC\_TRY\_FMT命令可用于查看硬件设备所支持的数据格式，而不直接指定一个数据格式。

***图像格式枚举***

VIDIOC\_ENUM\_FMT命令可以用于枚举设备所支持的图像格式，所有驱动都必须支持这个命令的调用。驱动在kernel space中是不能够进行图像格式的转化，因此必须直接枚举出硬件设备所支持的全部格式。

Image Cropping , insertion and scaling

V4L2接口支持对图像的剪切，插入和缩放功能。

***输入和输出***

V4L2接口会自动调用经典I/O方法 read(), write()，在打开设备之后。其他的输入输出的方法就需要协商以后才能调用。