OpenMV 内部资料

2017/05/08

目录

1、OpenMV 快速参考

1.1、OpenMV 摄像头

通用控制

pyb

```
import pyb

pyb.delay(50) # wait 50 milliseconds
pyb.millis() # number of milliseconds since bootup
pyb.repl_uart(pyb.UART(3, 9600)) # duplicate REPL on UA
RT(3)
pyb.wfi() # pause CPU, waiting for interrupt
pyb.stop() # stop CPU, waiting for external interrupt
```

LED

pyb.LED <pyb.LED>

```
from pyb import LED

led = LED(1) # red led
led.toggle()
led.on()
led.off()
```

这是 LED 引脚排列：

- LED(1) -> 红色 LED
- LED(2) -> 绿色 LED
- LED(3) -> 蓝色 LED
- LED(4) -> 红外 LEDs

引脚和 GPIO

pyb.Pin <pyb.Pin>

```
from pyb import Pin

p_out = Pin('P7', Pin.OUT_PP)
```

```
p_out.high()
p_out.low()

p_in = Pin('P7', Pin.IN, Pin.PULL_UP)
p_in.value() # get value, 0 or 1
```

这是 GPIO 引脚分配：

- Pin('P0') -> P0 (PB15) - 5V 耐压，3.3V 输出，高达 25 mA 驱动。
- Pin('P1') -> P1 (PB14) - 5V 耐压，3.3V 输出，高达 25 mA 驱动。
- Pin('P2') -> P2 (PB13) - 5V 耐压，3.3V 输出，高达 25 mA 驱动。
- Pin('P3') -> P3 (PB12) - 5V 耐压，3.3V 输出，高达 25 mA 驱动。
- Pin('P4') -> P4 (PB10) - 5V 耐压，3.3V 输出，高达 25 mA 驱动。
- Pin('P5') -> P5 (PB11) - 5V 耐压，3.3V 输出，高达 25 mA 驱动。
- Pin('P6') -> P6 (PA5) - 5V 耐压，3.3V 输出，高达 25 mA 驱动。
- Pin('P7') -> P7 (PD12) - 5V 耐压，3.3V 输出，高达 25 mA 驱动。
- Pin('P8') -> P8 (PD13) - 5V 耐压，3.3V 输出，高达 25 mA 驱动。

在 OpenMV M7 上：

- Pin('P9') -> P9 (PD14) - 5V 耐压，3.3V 输出，最高可达 25 mA 驱动。

不要在所有 I / O 引脚上总共超过 120 mA。

舵机控制

pyb.Servo <pyb.Servo>

```
from pyb import Servo

s1 = Servo(1) # servo on position 1 (P7)
s1.angle(45) # move to 45 degrees
s1.angle(-60, 1500) # move to -60 degrees in 1500ms
s1.speed(50) # for continuous rotation servos
```

这是舵机引脚分配：

- Servo(1) -> P7 (PD12)
- Servo(2) -> P8 (PD13)

在 OpenMV M7 上:

- Servo(3) -> P9 (PD14)

外部中断

pyb.ExtInt <pyb.ExtInt>

```
from pyb import Pin, ExtInt

callback = lambda e: print("intr")
ext = ExtInt(Pin('P7'), ExtInt.IRQ_RISING, Pin.PULL_NON
E, callback)
```

这是 GPIO 引脚分配：

- Pin('P0') -> P0 (PB15)
- Pin('P1') -> P1 (PB14)
- Pin('P2') -> P2 (PB13)
- Pin('P3') -> P3 (PB12)
- Pin('P4') -> P4 (PB10)
- Pin('P5') -> P5 (PB11)
- Pin('P6') -> P6 (PA5)
- Pin('P7') -> P7 (PD12)
- Pin('P8') -> P8 (PD13)

在 OpenMV M7 上:

- Pin('P9') -> P9 (PD14)

计时器

pyb.Timer <pyb.Timer>

```
from pyb import Timer

tim = Timer(4, freq=1000)
tim.counter() # get counter value
tim.freq(0.5) # 0.5 Hz
```

```
tim.callback(lambda t: pyb.LED(1).toggle())
```

这是定时器引脚分配：

- Timer 1 Channel 3 Negative -> P0
- Timer 1 Channel 2 Negative -> P1
- Timer 1 Channel 1 Negative -> P2
- Timer 2 Channel 3 Positive -> P4
- Timer 2 Channel 4 Positive -> P5
- Timer 2 Channel 1 Positive -> P6
- Timer 4 Channel 1 Negative -> P7
- Timer 4 Channel 2 Negative -> P8

在 OpenMV M7 上：

- Timer 4 Channel 3 Positive -> P9

PWM（脉宽调制）

pyb.Pin <pyb.Pin>``pyb.Timer <pyb.Timer>

```
from pyb import Pin, Timer

p = Pin('P7') # P7 has TIM4, CH1
tim = Timer(4, freq=1000)
ch = tim.channel(1, Timer.PWM, pin=p)
ch.pulse_width_percent(50)
```

这是定时器引脚分配：

- Timer 1 Channel 3 Negative -> P0
- Timer 1 Channel 2 Negative -> P1
- Timer 1 Channel 1 Negative -> P2
- Timer 2 Channel 3 Positive -> P4
- Timer 2 Channel 4 Positive -> P5
- Timer 2 Channel 1 Positive -> P6
- Timer 4 Channel 1 Negative -> P7
- Timer 4 Channel 2 Negative -> P8

在 OpenMV M7 上：

- Timer 4 Channel 3 Positive -> P9

ADC（模数转换）

pyb.Pin <pyb.Pin>``pyb.ADC <pyb.ADC>

```
from pyb import Pin, ADC

adc = ADC('P6')
adc.read() # read value, 0-4095
```

这是 ADC 引脚分配：

- ADC('P6') -> P6 (PA5) - 只有 3.3V（非 5V）电压在这个模式！

DAC（数模转换）

pyb.Pin <pyb.Pin> pyb.DAC <pyb.DAC>

```
from pyb import Pin, DAC

dac = DAC('P6')
dac.write(120) # output between 0 and 255
```

这是 ADC 引脚分配：

- DAC('P6') -> P6 (PA5) - 只有 3.3V（非 5V）电压在这个模式！

UART（串口）

pyb.UART <pyb.UART>

```
from pyb import UART

uart = UART(3, 9600)
uart.write('hello')
uart.read(5) # read up to 5 bytes
```

这是 UART 引脚分配：

- UART 3 RX -> P5 (PB11)
- UART 3 TX -> P4 (PB10)

在 OpenMV M7 上：

- UART 1 RX -> P0 (PB15)
- UART 1 TX -> P1 (PB14)

SPI 总线

pyb.SPI <pyb.SPI>

```
from pyb import SPI

spi = SPI(2, SPI.MASTER, baudrate=200000, polarity=1, p
hase=0)
spi.send('hello')
spi.recv(5) # receive 5 bytes on the bus
spi.send_recv('hello') # send a receive 5 bytes
```

这是 SPI 引脚分配：

- SPI 2 MOSI (Master-Out-Slave-In) -> P0 (PB15)
- SPI 2 MISO (Master-In-Slave-Out) -> P1 (PB14)
- SPI 2 SCLK (Serial Clock) -> P2 (PB13)
- SPI 2 SS (Serial Select) -> P3 (PB12)

I2C 总线

pyb.I2C <pyb.I2C>

```
from pyb import I2C

i2c = I2C(2, I2C.MASTER, baudrate=100000)
i2c.scan() # returns list of slave addresses
i2c.send('hello', 0x42) # send 5 bytes to slave with ad
dress 0x42
i2c.recv(5, 0x42) # receive 5 bytes from slave
i2c.mem_read(2, 0x42, 0x10) # read 2 bytes from slave 0
x42, slave memory 0x10
i2c.mem_write('xy', 0x42, 0x10) # write 2 bytes to slav
e 0x42, slave memory 0x10
```

这是 I2C 引脚排列：

- I2C 2 SCL (Serial Clock) -> P4 (PB10)
- I2C 2 SDA (Serial Data) -> P5 (PB11)

在 OpenMV M7 上：

- I2C 4 SCL (Serial Clock) -> P7 (PD13)
- I2C 4 SDA (Serial Data) -> P8 (PD12)

2、OpenMV 教程

2.1、I/O 教程

在进入机器视觉主题之前，我们将讨论您的 OpenMV 摄像头上的 I / O 引脚控制。重要的是您知道如何切换 I / O 引脚，传输和接收串行数据，并将您的 OpenMV 进行睡眠，以便您可以创建一个完整的系统。

根据您的 OpenMV 摄像机的型号，您可以使用 9-10 个可用于低速数字输入和输出的通用 I/O 引脚。请注意，我们使用具有 5V 支持的 I/O 引脚的 STM32 处理器，因此您可以直接将 OpenMV 摄像机连接到 Arduino 或其他 5V 器件，而无需担心。I/O 引脚也非常强大，每个可以源或吸收高达 25 mA。

无论如何，不同的 I / O 引脚都有不同的特殊功能。您的 OpenMV 的 SPI 总线引脚可用于控制 SPI 器件，例如 P0-P3 。P4-P5 是您的 OpenMV Cam 的异步串行或 I2C 总线，可以串行或 I2C 通讯。P6 是您的 OpenMV Cam 的 ADC / DAC 引脚，用于 0V 至 3.3V 输入和输出。而 P7-P8（或 P7-P9）是您的 OpenMV Cam 的辅助 I / O 引脚。

PYB 模块

所有微控制器 I / O 功能都可以从 pyb （Python Board）模块获得。您只需要 import pyb 在脚本中访问它。一旦导入，您将可以访问 ADC，CAN，DAC，I2C，引脚，伺服，SPI 和 UART 类，以及能够控制电路板的功耗。

2.2、LED 控制

您的 OpenMV 摄像头有一个 RGB LED 和两个红外 LED 灯。您可以单独控制 RGB LED 的红色，绿色和蓝色区段，将两个 IR LED 控制为一个单位。要控制 LED，首先导入 pyb 模块。然后为要控制的特定 LED 创建一个 LED 类对象

```
import pyb

red_led = pyb.LED(0)
green_led = pyb.LED(1)
blue_led = pyb.LED(2)
ir_leds = pyb.LED(3)
```

pyb.LED(number) 创建一个 LED 对象，您可以使用它来控制 特定的 LED。通过 pyb.LED 0”控制红色 RGB LED 段，“1” 控制绿色 RGB LED 段，“2”控制蓝色 RGB LED 段， “3”控制两个红外 LED。

在创建如上所述的 LED 控制对象后，我强烈建议您 调用 off() 新的 LED 的方法将其置于已知状态。

不管怎样，有三种方法可以调用每个 LED， ，， off(), on(), 与 toggle() 三个选项。

与其他 MicroPython 板不同，OpenMV Cam 不支持 intensity() 允许对 LED 进行 PWM 调光的方法。我们重新设计了用于 LED 调光的定时器，用于生成时钟源为相机芯片供电。

最后，在脚本中使用 RGB LED 作指示。作为红外 LED，那些用于夜视。当您为我们的 IR 镜头（这是没有红外线滤镜的镜头）切换您的 OpenMV Cam 的常规镜头时，您可以打开红外 LED，以使您的 OpenMV Cam 可以在黑暗中看到。红外 LED 足够亮，可以在黑色的 OpenMV Cam 前面照亮 3 米左右。

2.3、OpenMV IDE 概述

现在来谈谈 OpenMV IDE。OpenMV IDE 是您将用于编程 OpenMV 的工具。它具有强大的文本编辑器，由 QtCreator，帧缓冲器查看器，直方图显示器和集成串行终端提供支持，用于从 OpenMV 进行调试输出。

无论如何，IDE 或多或少是直接使用。在 IDE 中进行流水线化的工作已经付出了很多努力。但是，IDE 中应该知道几件事情。

文件菜单和示例

在 OpenMV IDE 中的文件菜单下，您可以使用所有标准文本编辑器选项。新建，打开，保存，另存为，打印等。无论如何，当您创建新文件或打开文件时，它将显示在文本编辑器窗格中。请注意，您可以打开多个文件，您可以通过单击显示当前文件名称的组合框来选择。OpenMV IDE 使用 QtCreator 作为文本编辑器后端，实际上支持在具有水平和垂直分割功能的多个窗口中打开多个文件。但是，所有这些都是隐藏的，以便明确当您点击 Arduino IDE 中的运行按钮时，哪个脚本将运行。如果要在不同的编辑器中编辑脚本， 或者您需要一次打开多个文件，只需打开所选编辑器中的文件并对其进行处理即可。OpenMV IDE 会自动检测文件已更改，并询问您是否要在单击 OpenMV IDE 窗口时加载更改。

文字编辑

OpenMV IDE 具有由 QtCreator 后端供电的专业文本编辑器。您可以在所有打开的文件，空白可视化（MicroPython 重要），字体大小控制和最佳的 QtCreator 查找和替换之间进行无限的撤消和重做。查找和替换功能与正常表达式匹配，在查找和使用捕获的文本进行替换时捕获文本。此外，它也可以在更换时保留大小写。最后，查找和替换功能不仅在当前文件上起作用，而且可以在文件夹中的所有文件或 OpenMV IDE 中的所有打开文件中起作用。

连接到您的 OpenMV 摄像头

连接到您的 OpenMV Cam 在上一个 Hardware Setup 教程页面中有详细介绍。一旦您了解所有涉及编程固件等的启动问题，您只需点击 OpenMV IDE 左下角的连接按钮即可连接到 OpenMV Cam。

OpenMV IDE 对于连接并自动过滤不是 OpenMV Cam 的所有串行端口都很聪明。如果只有一个 OpenMV Cam 连接到您的计算机，它会找到它并立即连接。如果您有两个 OpenMV 摄像头，那么它会询问您连接哪个串行端口。请注意，OpenMV IDE 会记住您的选择，以便下次如果连接要连接的 OpenMV Cam 的串行端口已经被选中。

连接到您的 OpenMV Cam 的串行端口后，OpenMV IDE 将尝试确定与您的 OpenMV Cam 相关联的计算机上的 USB 闪存驱动器。OpenMV IDE 会在 USB 闪存驱动器上进行一些智能过滤，以尽可能自动连接到正确的。但是，它可能无法自动确定正确的，如果不能，请求您帮助它。像 OpenMV IDE 上的串行端口一样会记住你的选择，所以下次连接时，它会自动突出显示你以前的选择。

最后连接到 OpenMV 后，连接按钮将被断开按钮取代。单击断开按钮以断开与 OpenMV 的连接。请注意，断开连接将停止您的 OpenMV 上当前正在执行的脚本。您也可以从计算机上拔下 OpenMV 而不断开连接，OpenMV IDE 会检测到并自动断开与 OpenMV 的连接。如果您的 OpenMV 崩溃，OpenMV IDE 会检测到这一点，并与 OpenMV 断开连接。

运行脚本

完成编辑代码并准备运行脚本后，只需单击 OpenMV IDE 左下角的绿色运行按钮即可。然后，该脚本将被发送到您的 OpenMV Cam，以编译成 Python 字节码，并由 OpenMV Cam 执行。

如果您的脚本中有任何错误，您的 OpenMV Cam 将在串行终端中发回编译错误，OpenMV IDE 会自动解析查找错误。当 OpenMV IDE 检测到错误时，它将自动打开文件错误，突出显示错误的行以及显示一个漂亮的错误消息框。此功能可节省大量修复错误的时间。

无论如何，如果要停止脚本，只需单击停止按钮，该脚本将在脚本运行时替换运行按钮。请注意，脚本可能由于完成或编译错误而自动停止。在任一情况下，运行按钮将重新出现。

帧缓冲区查看器

什么使 OpenMV IDE 特别的是集成的帧缓冲区查看器。这样， 您可以轻松地查看 OpenMV 在处理代码时看到的内容。

帧缓冲区查看器在以前调用时显示您的 OpenMV Cam 的帧缓冲区中的任 sensor.snapshot()内容。

最后，您可以右键单击在帧缓冲区查看器中看到的任何图像，将该图像保存到磁盘。另外，如果通过单击并拖动来选择帧缓冲区中的区域，则可以将该区域保存到磁盘。请注意，您应该在尝试将帧缓冲区保存到磁盘之前停止脚本。否则，您可能无法获得所需的准确框架。

要取消选择帧缓冲区中的区域，只需单击任意位置即可拖动以删除选区。但是，可以在取消选择时创建一个像素选项，以便尝试单击帧缓冲区中的空白区域。

直方图显示

OpenMV IDE 中集成的直方图显示主要用于填充帧缓冲区查看器下的空白空间，并为您提供一些视图。然而，它也有助于获得关于房间照明质量的反馈，确定颜色跟踪设置，一般只是让您了解 OpenMV 所查看的图像的质量 。

您可以在直方图中选择四种不同的颜色空间。无论是 RGB，灰度，LAB 和 YUV。只有灰度和 LAB 可用于以编程方式控制您的 OpenMV 摄像头。RGB 是非常好看的眼睛糖果和 YUV 是在那里，因为我们使用它的 JPEG 压缩，并认为我们可以添加它。

无论如何，默认情况下，直方图显示有关整个图像的信息。但是，如果通过单击并拖动来选择帧缓冲区的区域，则直方图将仅显示该区域中颜色的分布。这种特性使得直方图显示超确定正确有用的，你需要在你的脚本中使用的灰度和 LAB 色彩通道设置 image.find_blobs 和 image.binary。

串口

要显示串行终端，请单击位于 OpenMV IDE 底部的串行终端按钮。串行端口内置在主窗口中以便于使用。它只是分开你的文本编辑窗口。

无论如何，所创建的 OpenMV Cam 的所有调试文本 print 将显示在串行终端中。除此之外还没有什么可说的。

请注意，串行终端或多或少将无限缓冲文本。它会将最后一百万行文本保留在 RAM 中。所以，你可以用它来缓冲大量的调试输出。另外，如果您 ctrl+f 在 Windows / Linux 或 Mac 上的等效快捷方式按下，您将能够搜索调试输出。最后，如果您想查看以前的调试输出，使其非常好用，串行终端就不够自动滚动。如果滚动到文本输出的底部，则自动滚动将再次打开。

状态栏

在状态栏上，OpenMV IDE 将显示您的 OpenMV 的固件版本，串行端口，驱动器和 FPS。固件版本标签实际上是一个按钮，您可以点击更新您的 OpenMV，如果您的 OpenMV 的固件已过期。串行端口标签只显示您的 OpenMV 的串行端口，没有其他。驱动器标签是另一个按钮，您可以点击它来更改链接到您的 OpenMV 摄像机的驱动器。最后，FPS 标签显示 FPS OpenMV IDE 从您的 OpenMV 获得。

工具

您将在 OpenMV IDE 中的工具菜单下找到 OpenMV Cam 的实用工具。特别是，Save open script to your OpenMV Cam 和 Reset OpenMV 工具用于开发时使用是 OpenMV 有用的应用程序。

接下来，在“工具”菜单下，您可以调用引导加载程序来重新编程 OpenMV。只有当 OpenMV IDE 与 OpenMV 断开连接时，才能调用引导加载程序。您可以给它一个二进制.bin 文件重新编程您的 OpenMV 或.dfu 文件。引导加载程序功能仅适用于计划更改默认 OpenMV 固件的高级用户。

打开终端

打开终端功能允许您使用 OpenMV IDE 创建新的串行终端，以便在未连接到 计算机的情况下远程调试 OpenMV 。开放终端功能也可用于编程任何 MicroPython 开发板。

机器视觉

机器视觉子菜单包含许多用于您的 OpenMV 摄像机的机器视觉工具。例如，您可以使用颜色阈值编辑器获得 最佳的颜色跟踪阈值 image.find_blobs()。我们将 定期提供新的机器视觉工具，使您的生活更轻松。

2.4、硬件设置

将 OpenMV 摄像机连接到计算机之前，首先要清洁摄像机 IC。

接下来，找到一个 micro-usb 电缆，并将 OpenMV 连接到您的计算机，然后启动 OpenMV IDE。

在 Windows 上，您应该会看到有关安装驱动程序的 Windows 的一些通知。等到 Windows 完成安装驱动程序之后，OpenMV 的 USB 闪存驱动器出现，OpenMV 的蓝色指示灯闪烁，然后再尝试连接 OpenMV IDE。

如果您的 OpenMV 摄像机在连接到计算机也没有响应的情况下显示无响应。如果在连接后没有看到绿灯闪烁，那么我们可以使用 DFU 重新编程 OpenMV 摄像头。如果您看到绿灯但没有 USB 闪存驱动器出现，那么我们仍然可以使用 DFU 重新编程。

接下来，启动 OpenMV IDE，然后单击连接按钮（OpenMV IDE 中的左角）。OpenMV IDE 应该自动连接到您的 OpenMV。

您的 OpenMV Cam 现在可以使用了！

USB 闪存驱动器说明

您的 OpenMV Cam 内置有一个 USB 闪存驱动器。当您的 OpenMV 插入计算机时，会出现此闪存驱动器。

3、固件升级与更新

3.1、更新固件

01 安装 DFU 软件 下载地址 http://pan.baidu.com/s/1slDBPGL

02 下载最新固件 https://github.com/openmv/openmv/releases

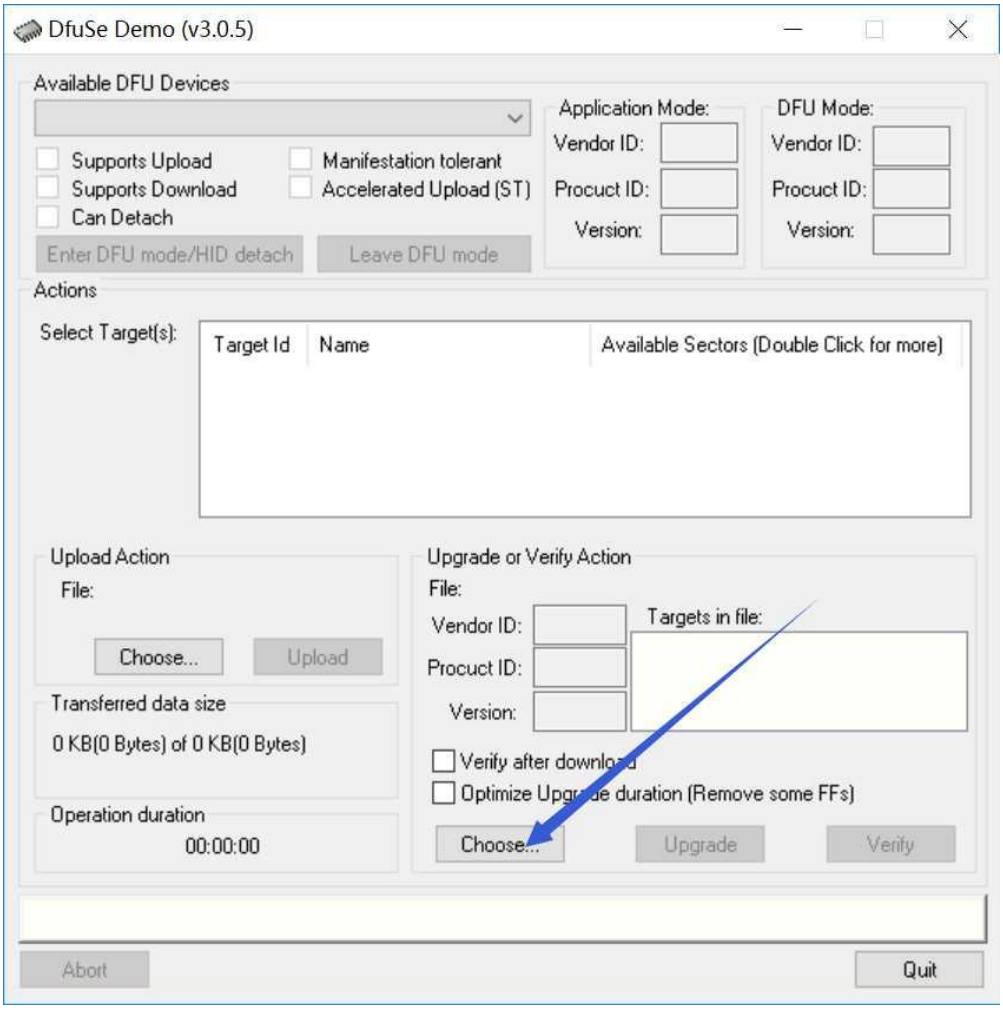
03 第三步，进入 DFU 模式 短接 Boot 引脚和 3.3V 引脚。（**如何进入 DFU 模式**）

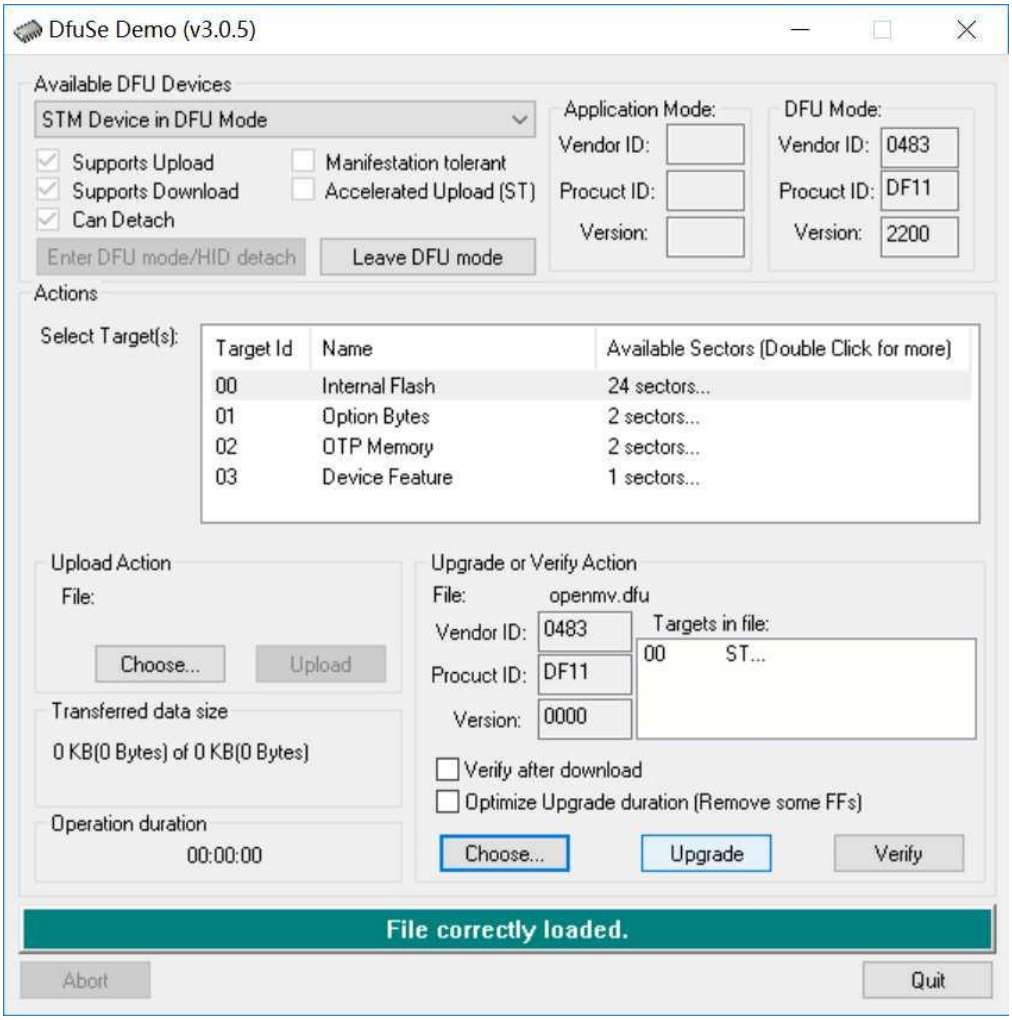
04 用 USB 先连接电脑后， 设备管理器会出现一个 DFU 设备。

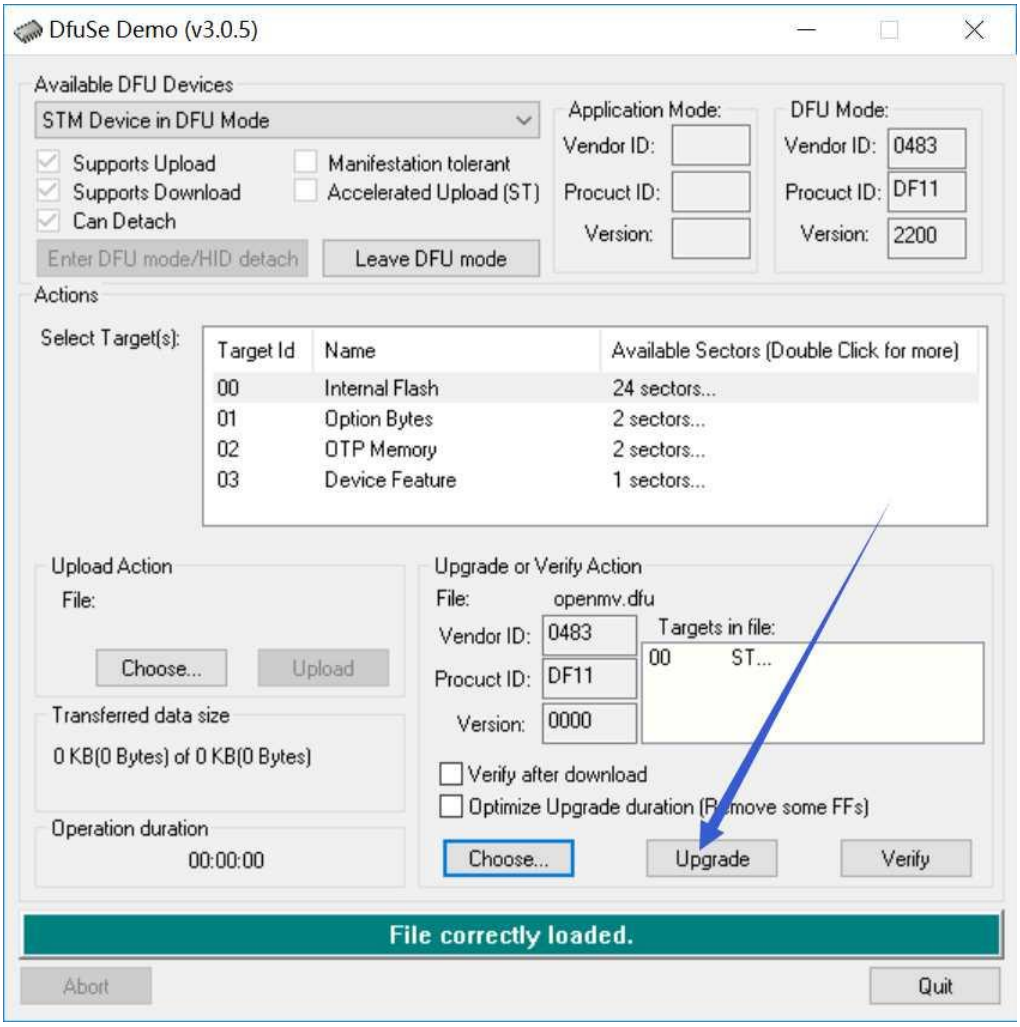
05 打开 DFU


STMicroelectronics
最近添加
Dfu file manager
最近添加
DfuSeDemo
最近添加
UM0412
最近添加

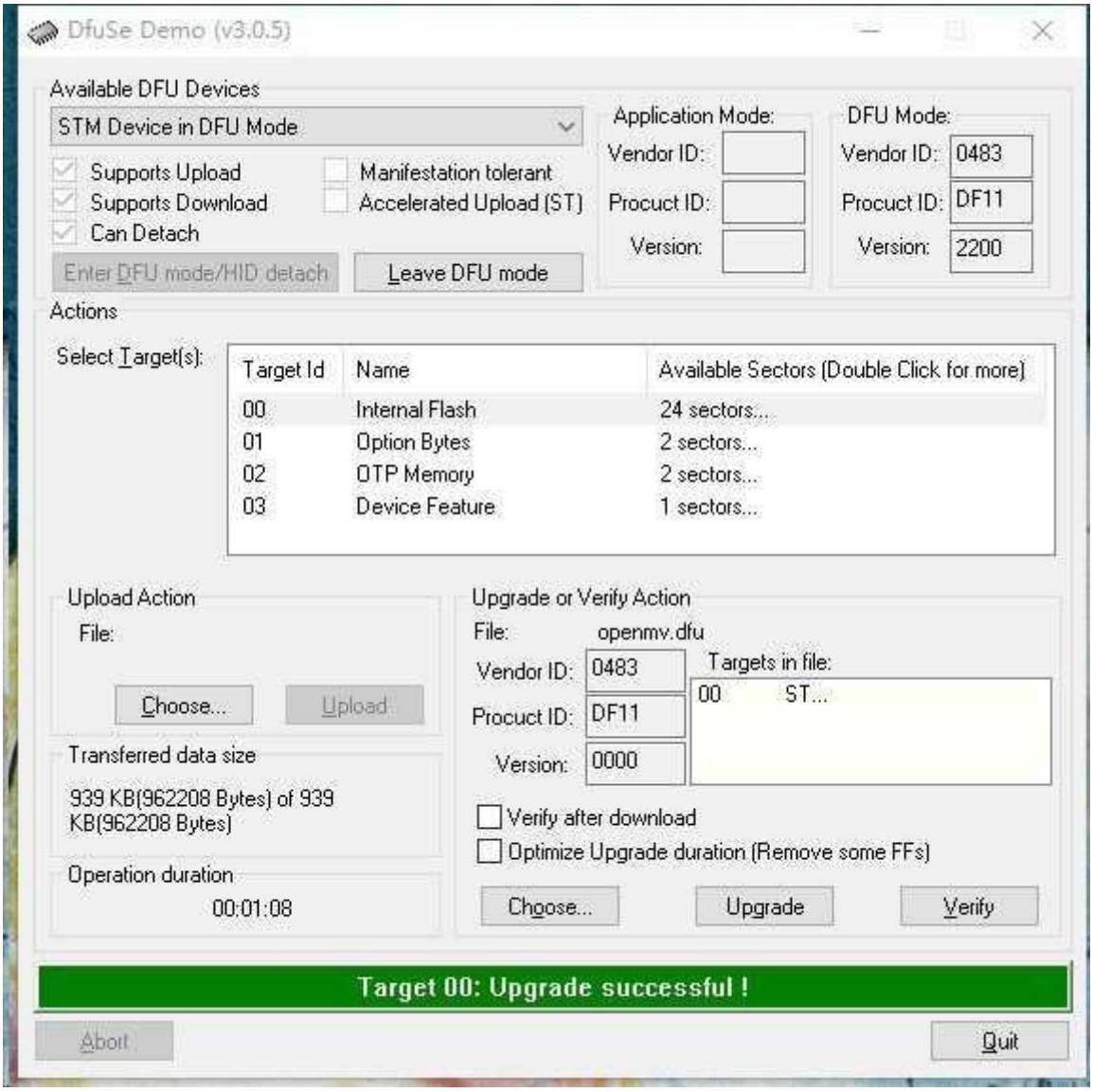
06 上传固件


DfuSe Demo (v3.0.5)
Available DFU Devices
Supports Upload
Supports Download
Can Detach
Manifestation tolerant
Accelerated Upload (ST)
Enter DFU mode/HID detach
Leave DFU mode
Application Mode:
Vendor ID:
Procuct ID:
Version:
DFU Mode:
Vendor ID:
Procuct ID:
Version:
Actions
Select Target(s):
Target Id
Name
Available Sectors (Double Click for more)
Upload Action
File:
Choose...
Upload
Transferred data size
0 KB(0 Bytes) of 0 KB(0 Bytes)
Operation duration
00:00:00
Upgrade or Verify Action
File:
Vendor ID:
Procuct ID:
Version:
Targets in file:
Verify after download
Optimize Upgrade duration (Remove some FFs)
Choose...
Upgrade
Verify
Abort
Quit


DfuSe Demo (v3.0.5)
Available DFU Devices
STM Device in DFU Mode
Supports Upload
Supports Download
Can Detach
Manifestation tolerant
Accelerated Upload (ST)
Enter DFU mode/HID detach
Leave DFU mode
Application Mode:
Vendor ID:
Procuct ID:
Version:
DFU Mode:
Vendor ID: 0483
Procuct ID: DF11
Version: 2200
Actions
Select Target(s):
Target Id
Name
Available Sectors (Double Click for more)
00 Internal Flash 24 sectors...
01 Option Bytes 2 sectors...
02 OTP Memory 2 sectors...
03 Device Feature 1 sectors...
Upload Action
File:
Choose...
Upload
Transferred data size
0 KB(0 Bytes) of 0 KB(0 Bytes)
Operation duration
00:00:00
Upgrade or Verify Action
File: openmv.dfu
Vendor ID: 0483
Procuct ID: DF11
Version: 0000
Targets in file:
00 ST...
Verify after download
Optimize Upgrade duration (Remove some FFs)
Choose...
Upgrade
Verify
File correctly loaded.
Abort
Quit



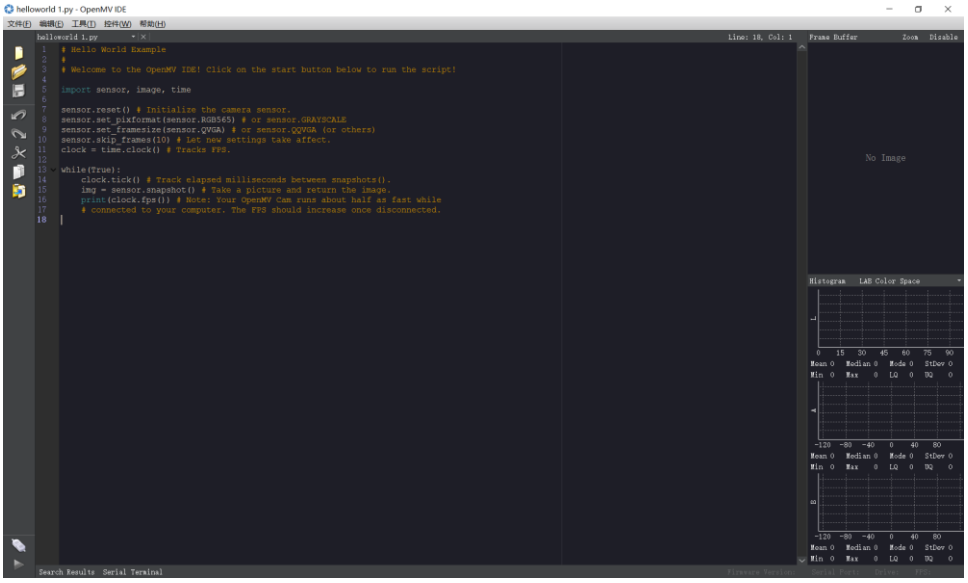
07 更新成功，重新插上 USB 线，运行 IDE 测试！



4、基本使用

4.1、软件与驱动安装

Windows 用户 下载 IDE 软件 https://openmv.io/pages/download
http://pan.baidu.com/s/1eRRUdjG



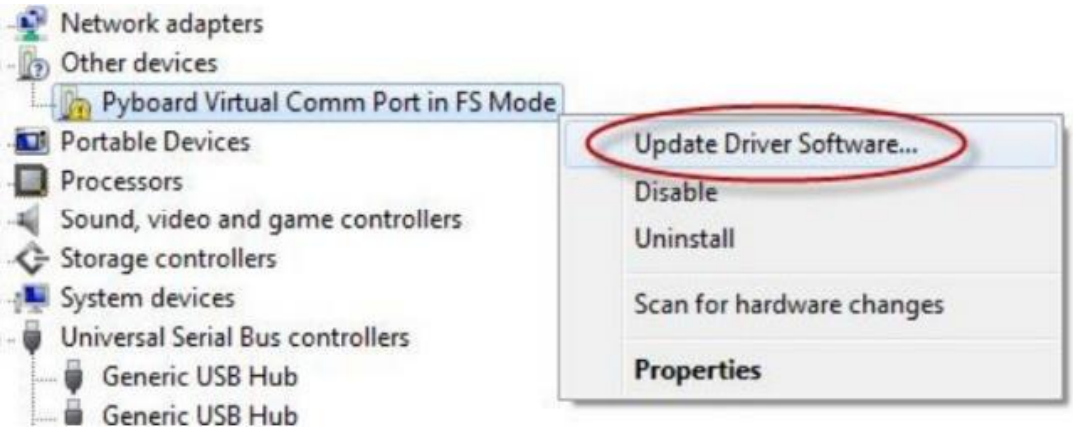
会自动安装驱动

若不能有效安装 请按如下步骤下载安装

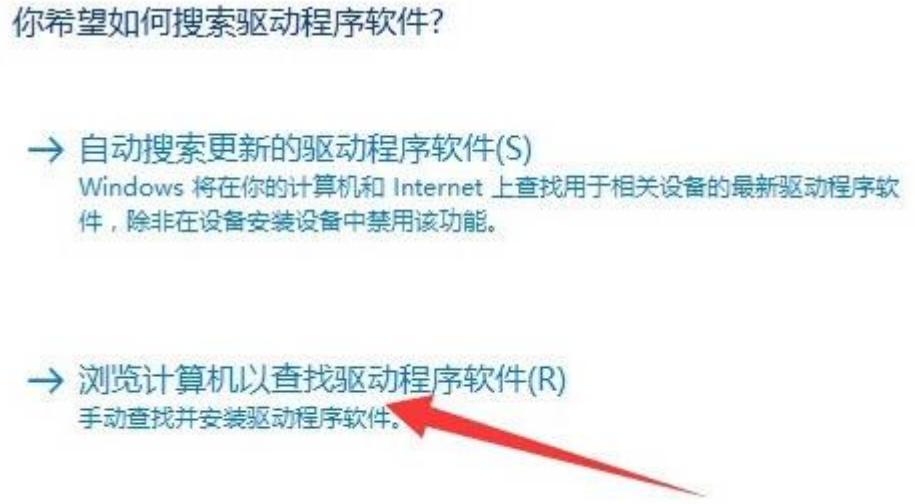
设备管理器中会出现叹号，则没有正常安装驱动。

首先下载驱动： http://pan.baidu.com/s/1qYVnF0S

解压到桌面，然后右键设备管理器未知设备，升级驱动：



选择 浏览计算机以查找驱动程序软件



成功的安装好驱动。

5、示例讲解

5.1、01-基本功能

5.1.1、helloworld

Hello World 示例 欢迎来到 OpenMV IDE！点击绿色运行箭头按钮运行脚本！

```
import sensor, image, time
sensor.reset()                      # 重置和初始化传感器。
sensor.set_pixformat(sensor.RGB565) # 设置像素格式为 RGB565（或灰度）
sensor.set_framesize(sensor.QVGA)   # 设置框架大小的 QVGA（320x240）
sensor.skip_frames(10)              # 等待设置生效
clock = time.clock()                # 创建一个时钟跟踪对象的 FPS。
while(True):
```

```
    clock.tick()                        # 更新帧时钟。
    img = sensor.snapshot()             # 拍照并返回图片。
    print(clock.fps())                  # 注：OpenMV 摄像头
运行约一半的快速连接时
                                        # 到 IDE。FPS 应该增
加一次断开。
```

5.1.2、基本示例

main

主要模块的例子 当 OpenMV 从您的计算机断开 main.py 脚本上的 SD 卡（如附件）或 main.py 脚本 你 OpenMV 的内部闪存驱动器。

```
import time, pyb

led = pyb.LED(3) # 红色 LED = 1, 绿色 LED = 2, 蓝色 LED
 = 3, 红外 LEDs = 4.
usb = pyb.USB_VCP()
```

这是一个允许你使用的串口对象与你的电脑通讯。虽然它没有打开下面的代码运行。

```
while(not usb.isconnected()):
    led.on()
    time.sleep(150)
    led.off()
    time.sleep(100)
    led.on()
    time.sleep(150)
    led.off()
    time.sleep(600)
led = pyb.LED(2) # 改用绿色 LED
while(usb.isconnected()):
    led.on()
    time.sleep(150)
    led.off()
    time.sleep(100)
    led.on()
    time.sleep(150)
    led.off()
    time.sleep(600)
```

5.2、02-基本驱动

5.2.1、ADC 读取示例

read_adc

本例展示了如何在 OpenMV 摄像机上读取 ADC。

```
import time
from pyb import ADC

adc = ADC("P6") # Must always be "P6".

while(True):
    # The ADC has 12-bits of resolution for 4096 value
s.
    print("ADC = %fv" % ((adc.read() * 3.3) / 4095))
    time.sleep(100)
```

5.2.2、DAC 控制示例

dac_control

此示例显示如何在 OpenMV 摄像机上使用 DAC 引脚输出。

```
import time
from pyb import DAC

dac = DAC("P6") # Must always be "P6".

while(True):
    # The DAC has 8-12 bits of resolution (default 8-bi
ts).
    for i in range(256):
        dac.write(i)
        time.sleep(20)
    for i in range(256):
        dac.write(255-i)
        time.sleep(20)
```

5.2.3、IIC 从机

arduino_i2c_slave

I2C，Arduino 作为主设备，OpenMV 作为从设备。 请将您的 OpenMV 连接到您的 Arduino，如下所示： OpenMV 主 I2C 数据（P5） -

Arduino Uno 数据（A4） OpenMV 主 I2C 时钟（P4） - Arduino Uno 时钟（A5） OpenMV 接地 - Arduino 接地

```
import pyb, ustruct

text = "Hello World!\n"
data = ustruct.pack("<%ds" % len(text), text)
```

使用“ustruct”构建要发送的数据包。 “

读我！！！ 请理解，当您的 OpenMV Cam 不是 I2C 主机时，它可能会错过响应 作为 I2C 从设备发送数据，无论是在中断回调中调用“i2c.send（）”还是在 main loop 下面。当这种情况发生时，Arduino 会得到一个 NAK，必须尝试从中读取 OpenMV Cam。注意，Arduino 和 OpenMV Cam I2C 驱动程序都不是很好 遇到任何 I2C 错误后不连接。在 OpenMV Cam 和 Arduino 上，你可以通过恢复 de-initing 然后重新启动 I2C 外设。

OpenMV Cam 的硬件 I2C 总线总是 I2C 总线 2。

```
bus = pyb.I2C(2, pyb.I2C.SLAVE, addr=0x12)
bus.deinit() #完全复位 I2C 设备...
bus = pyb.I2C(2, pyb.I2C.SLAVE, addr=0x12)
print("Waiting for Arduino...")
```

注意，为了同步正常工作，OpenMV Cam 必须在运行此脚本之前 Arduino 开始轮询 OpenMV Cam 数据。否则，I2C 字节成帧被弄乱了等，所以，保持 Arduino 在复位，直到 OpenMV 凸轮是“等待 Arduino ...”。

```
while(True):
    try:
        bus.send(ustruct.pack("<h", len(data)), timeout
=10000)
```

首先发送 len（16 位）。

```
        try:
            bus.send(data, timeout=10000) #第二次发送
数据。
            print("Sent Data!") #只到达时没有错误。
        except OSError as err:
            pass #不关心错误 - 所以通过。
```

请注意，有 3 个可能的错误。超时错误，通用错误或一个忙错误。“err.arg [0]”的错误代码分别为 116，5，16。 除了 OSError as err：

```
    except OSError as err:
        pass #不关心错误 - 所以通过。
```

请注意，有 3 个可能的错误。超时错误，通用错误或一个忙错误。“err.arg [0]”的错误代码分别为 116，5，16。

```
##############################################################
##############################################
# Arduino Code
##############################################################
##############################################
#
# #include <Wire.h>
# #define BAUD_RATE 19200
# #define CHAR_BUF 128
#
# void setup() {
#   Serial.begin(BAUD_RATE);
#   Wire.begin();
#   delay(1000); // Give the OpenMV Cam time to bootup.
# }
#
# void loop() {
#   int32_t temp = 0;
#   char buff[CHAR_BUF] = {0};
#
#   Wire.requestFrom(0x12, 2);
#   if(Wire.available() == 2) { // got length?
#
#     temp = Wire.read() | (Wire.read() << 8);
#     delay(1); // Give some setup time...
#
#     Wire.requestFrom(0x12, temp);
#     if(Wire.available() == temp) { // got full messag
e?
#
#       temp = 0;
#       while(Wire.available()) buff[temp++] = Wire.rea
d();
#
```

```
#     } else {
#       while(Wire.available()) Wire.read(); // Toss ga
rbage bytes.
#     }
#   } else {
#     while(Wire.available()) Wire.read(); // Toss garb
age bytes.
#   }
#
#   Serial.print(buff);
#   delay(1); // Don't loop to quickly.
# }
```

5.2.4、IIC 控制

I2C 控制

此示例显示如何通过将 内容转储到标准 EEPROM 上来在 OpenMV Cam 上使用 i2c 总线。要运行此示例，请将 Thermopile Shield 连接到 OpenMV Cam 或 I2C EEPROM 到您的 OpenMV Cam。

```
from pyb import I2C

i2c = I2C(2, I2C.MASTER) # The i2c bus must always be
2.
mem = i2c.mem_read(256, 0x50, 0) # The eeprom slave add
ress is 0x50.

print("\n[")
for i in range(16):
    print("\t[", end='')
    for j in range(16):
        print("%03d" % mem[(i*16)+j], end='')
        if j != 15: print(", ", end='')
    print("]," if i != 15 else "]")
print("]")
```

5.2.5、LED 控制示例

led_control

此示例显示如何控制 OpenMV Cam 的内置 LED。使用 智能手机的相机查看红外 LED。

```
import time
from pyb import LED

red_led   = LED(1)
green_led = LED(2)
blue_led  = LED(3)
ir_led    = LED(4)

def led_control(x):
    if   (x&1)==0: red_led.off()
    elif (x&1)==1: red_led.on()
    if   (x&2)==0: green_led.off()
    elif (x&2)==2: green_led.on()
    if   (x&4)==0: blue_led.off()
    elif (x&4)==4: blue_led.on()
    if   (x&8)==0: ir_led.off()
    elif (x&8)==8: ir_led.on()

while(True):
    for i in range(16):
        led_control(i)
        time.sleep(500)
```

5.2.6、PWM 控制示例

pwm_control

该示例显示如何使用 OpenMV Cam 进行 PWM。

```
import time
from pyb import Pin, Timer

tim = Timer(4, freq=1000) # Frequency in Hz
```

在通道 1 和 2 上分别在 50%和 75%占空比的 TIM4 上产生 1KHz 方波。

```
ch1 = tim.channel(1, Timer.PWM, pin=Pin("P7"), pulse_wi
dth_percent=50)
ch2 = tim.channel(2, Timer.PWM, pin=Pin("P8"), pulse_wi
dth_percent=75)

while (True):
    time.sleep(1000)
```

5.2.7、SPI 从机

arduino_spi_slave

SPI 以 Arduino 为主设备，OpenMV 为从机。

请将您的 OpenMV Cam 连接到您的 Arduino，如下所示：

OpenMV MOSI (P0) - Arduino MOSI (11) OpenMV MISO (P1) - Arduino MISO (12) OpenMV SCK (P2) - Arduino SCK (13) OpenMV SS (P3) - Arduino SS (10) OpenMV Ground - Arduino Ground

```
import pyb, ustruct

text = "Hello World!\n"
data = ustruct.pack("<bi%ds" % len(text), 85, len(tex
t), text) # 85 is a sync char.
```

使用“ustruct”构建要发送的数据包。 “

零填充数据为 4 字节加 4 字节的倍数。

```
data += "\x00" * (4 + (len(data) % 4))
```

请注意，当您的 OpenMV 不是 SPI 主机时， 无论您是否在中断回调或 下面的主循环中调用“spi.send（）”，都可能会错过响应发送数据作为 SPI 从站。因此，必须绝对设计您的通信协议， 以便如果从设备（OpenMV） 在主机（Arduino）从 SPI 外设读取的垃圾数据的时候不会调用“spi.send（）” 扔了 为了完成这个，我们使用一个 85 字节（二进制 01010101）的同步字符，Arduino 将会读取第一个字节。 如果没有看到，那么它会中止 SPI 事务，并将重试。第二，为了清除 SPI 外设状态，我们总是发送四个字节的倍数和一个额外的四个零 字节，以确保 SPI 外设不会保留任何可能为 85 的过时数据。请注意 ，OpenMV Cam 将错过“spi.send（）”窗口随机，因为它必须服务 中断。中断在连接到 PC 时会发生更多的事情。

您的 OpenMV 的硬件 SPI 总线始终为 SPI 总线 2. 极性= 0 - >时钟空闲为低电平。 相位= 0 - >上升时钟沿的采样数据，在下降时钟沿输出数据。

```
spi = pyb.SPI(2, pyb.SPI.SLAVE, polarity=0, phase=0)
pin = pyb.Pin("P3", pyb.Pin.IN, pull=pyb.Pin.PULL_UP)
print("Waiting for Arduino...")
```

请注意，为了使同步正常工作，在 Arduino 开始轮询 OpenMV Cam 之前，OpenMV Cam 必须运行此脚本。否则 SPI 字节帧被弄乱 等等。所以，保持 Arduino 重置，直到 OpenMV Cam “等待 Arduino ...”。

```
while(True):
    while(pin.value()): pass
    try:
        spi.send(data, timeout=1000)
        # If we failed to sync up the first time we'll
sync up the next time.
        print("Sent Data!") # Only reached on no error.
    except OSError as err:
        pass # Don't care about errors - so pass.
        # Note that there are 3 possible errors. A time
out error, a general purpose error, or
        # a busy error. The error codes are 116, 5, 16
respectively for "err.arg[0]".
    while(not pin.value()): pass
```

Arduino 代码

```
#
# #include <SPI.h>
# #define SS_PIN 10
# #define BAUD_RATE 19200
# #define CHAR_BUF 128
#
# void setup() {
#   pinMode(SS_PIN, OUTPUT);
#   Serial.begin(BAUD_RATE);
#   SPI.begin();
#   SPI.setBitOrder(MSBFIRST);
#   SPI.setClockDivider(SPI_CLOCK_DIV16);
#   SPI.setDataMode(SPI_MODE0);
#   delay(1000); // Give the OpenMV Cam time to bootup.
# }
#
# void loop() {
#   int32_t temp = 0;
#   char buff[CHAR_BUF] = {0};
#   digitalWrite(SS_PIN, LOW);
#   delay(1); // Give the OpenMV Cam some time to setup
 to send data.
#
```

```
#   if(SPI.transfer(1) == 85) { // saw sync char?
#     SPI.transfer(&temp, 4); // get length
#     int zero_legnth = 4 + ((temp + 1) % 4);
#     if (temp) {
#       SPI.transfer(&buff, min(temp, CHAR_BUF));
#       temp -= min(temp, CHAR_BUF);
#     }
#     while (temp--) SPI.transfer(0); // eat any remain
ing bytes
#     while (zero_legnth--) SPI.transfer(0); // eat zer
os.
#   }
#
#   digitalWrite(SS_PIN, HIGH);
#   Serial.print(buff);
#   delay(1); // Don't loop to quickly.
# }
```

5.2.8、SPI 控制

spi_control

此示例显示如何在 OpenMV Cam 上直接使用 SPI 总线， 而不使用内置的 lcd 屏蔽驱动程序。您将需要 LCD 屏幕运行此示例。

```
import sensor, image, time
from pyb import Pin, SPI

cs  = Pin("P3", Pin.OUT_OD)
rst = Pin("P7", Pin.OUT_PP)
rs  = Pin("P8", Pin.OUT_PP)
```

您的 OpenMV Cam 的硬件 SPI 总线始终为 SPI 总线 2。

```
spi = SPI(2, SPI.MASTER, baudrate=int(1000000000/66), p
olarity=0, phase=0)

def write_command_byte(c):
    cs.low()
    rs.low()
    spi.send(c)
    cs.high()

def write_data_byte(c):
```

```
    cs.low()
    rs.high()
    spi.send(c)
    cs.high()

def write_command(c, *data):
    write_command_byte(c)
    if data:
        for d in data: write_data_byte(d)

def write_image(img):
    cs.low()
    rs.high()
    spi.send(img)
    cs.high()
```

复位 LCD。

```
rst.low()
time.sleep(100)
rst.high()
time.sleep(100)

write_command(0x11) # Sleep Exit
time.sleep(120)
```

内存数据访问控制

```
write_command(0x36, 0xC0)
```

接口像素格式

```
write_command(0x3A, 0x05)
```

显示开

```
write_command(0x29)

sensor.reset() # Initialize the camera sensor.
sensor.set_pixformat(sensor.RGB565) # must be this
sensor.set_framesize(sensor.QQVGA2) # must be this
sensor.skip_frames(10) # Let new settings take affect.
clock = time.clock() # Tracks FPS.
```

```
while(True):
    clock.tick() # Track elapsed milliseconds between s
napshots().
    img = sensor.snapshot() # Take a picture and return
 the image.

    write_command(0x2C) # Write image command...
    write_image(img)

    print(clock.fps()) # Note: Your OpenMV Cam runs abo
ut half as fast while
```

连接到您的计算机。一旦断开连接，FPS 应该增加。

5.2.9、串口

uart_control

此示例显示如何使用 OpenMV Cam 上的串行端口。 连接销 P4 串口液晶屏的串口输入看看“Hello World！” 印刷 在串行 LCD 显示屏上。

```
import time
from pyb import UART
```

始终通过 UART 3 获取 OpenMV Cam 的 UART 编号。 第二个参数是 UART 波特率。 用于更高级的 UART 控制 示例参见 BLE-Shield 驱动程序。

```
uart = UART(3, 19200)

while(True):
    uart.write("Hello World!\r")
    time.sleep(1000)
```

5.2.10、定时器控制示例

timer_control

此示例显示如何使用计时器回调。

```
import time
from pyb import Pin, Timer
```

```
def tick(timer):                # 当我们调用时，我们将收到定
时器对象
    print("Timer callback")

tim = Timer(4, freq=1)          # 使用定时器 4 创建一个定时器
对象 - 在 1Hz 触发
tim.callback(tick)              # 设置回调到我们的 tick 函数

while (True):
    time.sleep(1000)
```

5.2.11、引脚控制实例

pin_control

这个例子展示了如何使用 I/O 引脚 GPIO 的方式对你 OpenMV

```
from pyb import Pin
```

将开关连接到针脚 0，当开关闭合时将开关拉低 引脚 1 将置高电平

```
pin0 = Pin('P0', Pin.IN, Pin.PULL_UP)
pin1 = Pin('P1', Pin.OUT_PP, Pin.PULL_NONE)

while(True):
    pin1.value(not pin0.value())
```

5.2.12、舵机控制

servo_control

此示例显示如何使用您的 OpenMV Cam 来控制舵机。

```
import time
from pyb import Servo

s1 = Servo(1) # P7
s2 = Servo(2) # P8

while(True):
    for i in range(1000):
        s1.pulse_width(1000 + i)
        s2.pulse_width(1999 - i)
        time.sleep(10)
```

```
    for i in range(1000):
        s1.pulse_width(1999 - i)
        s2.pulse_width(1000 + i)
        time.sleep(10)
```

5.2.13、超频示例

overclocking

这个例子说明如何将 OMV2 摄像机的超频超频到 216MHz。 如果您需要保持此频率 从主脚本调用 set_frequency 功能，相机将保持超频，直到下一次重新设置。

警告：超频至 216MHz 应该是安全的，但使用您自担风险！

```
import cpufreq
```

打印当前 CPU 频率

```
print(cpufreq.get_frequency())
```

设定频率有效值为（120，144，168，192，216）

```
cpufreq.set_frequency(cpufreq.CPUFREQ_216MHZ)
```

打印当前 CPU 频率

```
print(cpufreq.get_frequency())
```

5.3、03-绘图

5.3.1、图像复制帧缓冲区

copy2fb

此示例显示如何将图像加载并复制到帧缓冲区进行测试。

```
import sensor, image
```

仍需要 init 传感器

```
sensor.reset()
# Set sensor settings
sensor.set_contrast(1)
```

```
sensor.set_gainceiling(16)
```

设置传感器像素格式

```
sensor.set_framesize(sensor.QQVGA)
sensor.set_pixformat(sensor.GRAYSCALE)
```

加载图像

```
img = image.Image("/image.pgm", copy_to_fb=True)
```

在这里添加图纸代码。 img.draw_line（...）

冲洗 FB

```
sensor.snapshot()
```

5.3.2、彩色绘制示例

color_drawing

此示例显示了您的 OpenMV Cam 的内置绘图功能。这个 例子最初是一个测试，但作为一个很好的参考代码。请将 IDE 置于非 JPEG 模式，以查看最佳绘图质量。

```
import sensor, image, time

sensor.reset()
sensor.set_framesize(sensor.QVGA)
```

所有绘图功能都使用相同的代码传递颜色。 所以我们只需要测试一个功能。

```
while(True):

    # Test Draw Line (GRAYSCALE)
    sensor.set_pixformat(sensor.GRAYSCALE)
    for i in range(10):
        img = sensor.snapshot()
    for i in range(img.width()):
        c = ((i * 255) + (img.width()/2)) / img.width()
        img.draw_line([i, 0, i, img.height()-1], color
= int(c))
    sensor.snapshot()
```

```
    time.sleep(1000)
```

测试绘制线（RGB565）

```
    sensor.set_pixformat(sensor.RGB565)
    for i in range(10):
        img = sensor.snapshot()
    for i in range(img.width()):
        c = ((i * 255) + (img.width()/2)) / img.width()
        img.draw_line([i, 0, i, img.height()-1], color
= [int(c), 0, 0])
    sensor.snapshot()
    time.sleep(1000)
```

测试绘制线（RGB565）

```
    sensor.set_pixformat(sensor.RGB565)
    for i in range(10):
        img = sensor.snapshot()
    for i in range(img.width()):
        c = ((i * 255) + (img.width()/2)) / img.width()
        img.draw_line([i, 0, i, img.height()-1], color
= [0, int(c), 0])
    sensor.snapshot()
    time.sleep(1000)
```

测试绘制线（RGB565）

```
    sensor.set_pixformat(sensor.RGB565)
    for i in range(10):
        img = sensor.snapshot()
    for i in range(img.width()):
        c = ((i * 255) + (img.width()/2)) / img.width()
        img.draw_line([i, 0, i, img.height()-1], color
= [0, 0, int(c)])
    sensor.snapshot()
    time.sleep(1000)
```

5.3.3、绘制各种标识

[TOC]

crazy_drawing

此示例显示了您的 OpenMV Cam 的内置绘图功能。这个 例子最初是一个测试，但作为一个很好的参考代码。请将 IDE 置于非 JPEG 模式，以查看最佳绘图质量。

```
import pyb, sensor, image, math

sensor.reset()
sensor.set_framesize(sensor.QVGA)

while(True):
```

测试绘制点

```
    sensor.set_pixformat(sensor.GRAYSCALE)
    for i in range(10):
        img = sensor.snapshot()
        for j in range(100):
            x = (pyb.rng() % (2*img.width())) - (img.wi
dth()//2)
            y = (pyb.rng() % (2*img.height())) - (img.h
eight()//2)
            img.set_pixel(x, y, 255)
    sensor.set_pixformat(sensor.RGB565)
    for i in range(10):
        img = sensor.snapshot()
        for j in range(100):
            x = (pyb.rng() % (2*img.width())) - (img.wi
dth()//2)
            y = (pyb.rng() % (2*img.height())) - (img.h
eight()//2)
            img.set_pixel(x, y, (255, 255, 255))
```

测试绘制线

```
    sensor.set_pixformat(sensor.GRAYSCALE)
    for i in range(10):
        img = sensor.snapshot()
        for j in range(100):
            x0 = (pyb.rng() % (2*img.width())) - (img.w
idth()//2)
            y0 = (pyb.rng() % (2*img.height())) - (img.
height()//2)
```

```
            x1 = (pyb.rng() % (2*img.width())) - (img.w
idth()//2)
            y1 = (pyb.rng() % (2*img.height())) - (img.
height()//2)
            img.draw_line([x0, y0, x1, y1])
    sensor.set_pixformat(sensor.RGB565)
    for i in range(10):
        img = sensor.snapshot()
        for j in range(100):
            x0 = (pyb.rng() % (2*img.width())) - (img.w
idth()//2)
            y0 = (pyb.rng() % (2*img.height())) - (img.
height()//2)
            x1 = (pyb.rng() % (2*img.width())) - (img.w
idth()//2)
            y1 = (pyb.rng() % (2*img.height())) - (img.
height()//2)
            img.draw_line([x0, y0, x1, y1])
```

测试绘制矩形

```
    sensor.set_pixformat(sensor.GRAYSCALE)
    for i in range(10):
        img = sensor.snapshot()
        for j in range(100):
            x = (pyb.rng() % (2*img.width())) - (img.wi
dth()//2)
            y = (pyb.rng() % (2*img.height())) - (img.h
eight()//2)
            w = (pyb.rng() % img.width())
            h = (pyb.rng() % img.height())
            img.draw_rectangle([x, y, w, h])
    sensor.set_pixformat(sensor.RGB565)
    for i in range(10):
        img = sensor.snapshot()
        for j in range(100):
            x = (pyb.rng() % (2*img.width())) - (img.wi
dth()//2)
            y = (pyb.rng() % (2*img.height())) - (img.h
eight()//2)
            w = (pyb.rng() % img.width())
            h = (pyb.rng() % img.height())
            img.draw_rectangle([x, y, w, h])
```

测试绘制圆

```
    sensor.set_pixformat(sensor.GRAYSCALE)
    for i in range(10):
        img = sensor.snapshot()
        for j in range(100):
            x = (pyb.rng() % (2*img.width())) - (img.wi
dth()//2)
            y = (pyb.rng() % (2*img.height())) - (img.h
eight()//2)
            r = (pyb.rng() % (img.width() if (img.width
() > img.height()) else img.height()))
            img.draw_circle(x, y, r)
    sensor.set_pixformat(sensor.RGB565)
    for i in range(10):
        img = sensor.snapshot()
        for j in range(100):
            x = (pyb.rng() % (2*img.width())) - (img.wi
dth()//2)
            y = (pyb.rng() % (2*img.height())) - (img.h
eight()//2)
            r = (pyb.rng() % (img.width() if (img.width
() > img.height()) else img.height()))
            img.draw_circle(x, y, r)
```

测试绘制字符串

```
    sensor.set_pixformat(sensor.GRAYSCALE)
    for i in range(10):
        img = sensor.snapshot()
        for j in range(100):
            x = (pyb.rng() % (2*img.width())) - (img.wi
dth()//2)
            y = (pyb.rng() % (2*img.height())) - (img.h
eight()//2)
            img.draw_string(x, y, "Hello\nWorld!")
    sensor.set_pixformat(sensor.RGB565)
    for i in range(10):
        img = sensor.snapshot()
        for j in range(100):
            x = (pyb.rng() % (2*img.width())) - (img.wi
dth()//2)
```

```
            y = (pyb.rng() % (2*img.height())) - (img.h
eight()//2)
            img.draw_string(x, y, "Hello\nWorld!")
```

测试绘制十字架

```
    sensor.set_pixformat(sensor.GRAYSCALE)
    for i in range(10):
        img = sensor.snapshot()
        for j in range(100):
            x = (pyb.rng() % (2*img.width())) - (img.wi
dth()//2)
            y = (pyb.rng() % (2*img.height())) - (img.h
eight()//2)
            img.draw_cross(x, y)
    sensor.set_pixformat(sensor.RGB565)
    for i in range(10):
        img = sensor.snapshot()
        for j in range(100):
            x = (pyb.rng() % (2*img.width())) - (img.wi
dth()//2)
            y = (pyb.rng() % (2*img.height())) - (img.h
eight()//2)
            img.draw_cross(x, y)
```

测试绘制关键点

```
    sensor.set_pixformat(sensor.GRAYSCALE)
    for i in range(10):
        img = sensor.snapshot()
        for j in range(100):
            x = (pyb.rng() % (2*img.width())) - (img.wi
dth()//2)
            y = (pyb.rng() % (2*img.height())) - (img.h
eight()//2)
            a = (pyb.rng() % (2*math.pi))
            img.draw_keypoints([(x, y, a)])
    sensor.set_pixformat(sensor.RGB565)
    for i in range(10):
        img = sensor.snapshot()
        for j in range(100):
            x = (pyb.rng() % (2*img.width())) - (img.wi
dth()//2)
```

```
            y = (pyb.rng() % (2*img.height())) - (img.h
eight()//2)
            a = (pyb.rng() % (2*math.pi))
            img.draw_keypoints([(x, y, a)])
```

5.4、04-图像过滤器

5.4.1、中值滤波器

median_filter

此示例显示中值滤波。中值滤波使用 NxN 邻域的中值替换每个像素。中值滤波对于消除图像中的噪声是有好处的，同时保留边缘。

```
import sensor, image, time

sensor.reset() # Initialize the camera sensor.
sensor.set_pixformat(sensor.RGB565) # or sensor.GRAYSCA
LE
sensor.set_framesize(sensor.QQVGA) # or sensor.QVGA (or
 others)
sensor.skip_frames(10) # Let new settings take affect.
clock = time.clock() # Tracks FPS.

while(True):
    clock.tick() # Track elapsed milliseconds between s
napshots().
    img = sensor.snapshot() # Take a picture and return
 the image.
```

中值滤波器的第一个参数是内核大小，分别对于 1x1,3x3 或 5x5 内核可以是 0,1 或者 2。第二个参数“百分位数”是从 NxN 附近选择的百分位数。0.5 是中位数，0.25 是下四分位数，0.75 是上四分位数。

```
    img.median(1, percentile=0.5)

    print(clock.fps()) # Note: Your OpenMV Cam runs abo
ut half as fast while
```

连接到您的计算机。一旦断开连接，FPS 应该增加。

5.4.2、中点过滤器

midpoint_filter

此示例显示中点过滤。中点过滤将每个 像素替换为 NxN 邻域的最小和最大像素值的平均值。

```
import sensor, image, time

sensor.reset() # Initialize the camera sensor.
sensor.set_pixformat(sensor.RGB565) # or sensor.GRAYSCA
LE
sensor.set_framesize(sensor.QQVGA) # or sensor.QVGA (or
 others)
sensor.skip_frames(10) # Let new settings take affect.
clock = time.clock() # Tracks FPS.

while(True):
    clock.tick() # Track elapsed milliseconds between s
napshots().
    img = sensor.snapshot() # Take a picture and return
 the image.
```

第一个参数是内核大小。N 对应于（（N *2）+1）^ 2 内核大小。例如 1* 3x3 内核，2 == 5x5 内核等。注意：您不应该使用大于 2 的值。“偏差”参数可让您在最小和最大混合之间进行选择。0.5 ==中点过滤器，0.0 ==最小过滤器和 1.0 ==最大过滤器。请注意，最小过滤器使图像变暗，而最大过滤器使图像更轻。

```
    img.midpoint(1, bias=0.5)

    print(clock.fps()) # Note: Your OpenMV Cam runs abo
ut half as fast while
```

连接到您的计算机。一旦断开连接，FPS 应该增加。

5.4.3、侵蚀和扩张的

erode_and_dilate

此示例显示了可以在二进制图像上运行以消除噪点的侵蚀和扩展功能。这个例子最初是一个测试，但它有助于炫耀这些功能如何工作。

```
import pyb, sensor, image

sensor.reset()
sensor.set_framesize(sensor.QVGA)
```

```
grayscale_thres = (170, 255)
rgb565_thres = (70, 100, -128, 127, -128, 127)

while(True):

    sensor.set_pixformat(sensor.GRAYSCALE)
    for i in range(20):
        img = sensor.snapshot()
        img.binary([grayscale_thres])
        img.erode(2)
    for i in range(20):
        img = sensor.snapshot()
        img.binary([grayscale_thres])
        img.dilate(2)

    sensor.set_pixformat(sensor.RGB565)
    for i in range(20):
        img = sensor.snapshot()
        img.binary([rgb565_thres])
        img.erode(2)
    for i in range(20):
        img = sensor.snapshot()
        img.binary([rgb565_thres])
        img.dilate(2)
```

5.4.4、基本图像帧区别

basic_frame_differencing

注意：您需要一张 SD 卡才能运行此示例。

本示例演示了如何使用 OpenMV Cam 进行帧差分。这 被称为基本帧差异，因为没有背景图像更新。 所以，随着时间的流逝，背景图片可能会改变，导致问题。

```
import sensor, image, pyb, os, time

sensor.reset() # Initialize the camera sensor.
sensor.set_pixformat(sensor.RGB565) # or sensor.GRAYSCA
LE
sensor.set_framesize(sensor.QVGA) # or sensor.QQVGA (or
 others)
sensor.skip_frames(10) # Let new settings take affect.
```

```
sensor.set_auto_whitebal(False) # Turn off white balanc
e.
clock = time.clock() # Tracks FPS.

if not "temp" in os.listdir(): os.mkdir("temp") # Make
a temp directory

print("About to save background image...")
sensor.skip_frames(60) # Give the user time to get read
y.
sensor.snapshot().save("temp/bg.bmp")
print("Saved background image - Now frame differencing!
")

while(True):
    clock.tick() # Track elapsed milliseconds between s
napshots().
    img = sensor.snapshot() # Take a picture and return
 the image.
```

用“abs（NEW-OLD）”框架更换图像。 img.difference（“temp / bg.bmp”） img.difference("temp/bg.bmp")

```
    print(clock.fps()) # Note: Your OpenMV Cam runs abo
ut half as fast while
```

连接到您的计算机。一旦断开连接，FPS 应该增加。

5.4.5、平均过滤器

mean_filter

此示例显示平均值过滤。平均过滤是 NxN 邻域中的标准平均过滤器。平均过滤通过模拟所有内容来消除图像中的噪点。但是，它是最快的内核过滤器操作。

```
import sensor, image, time

sensor.reset() # Initialize the camera sensor.
sensor.set_pixformat(sensor.RGB565) # or sensor.GRAYSCA
LE
sensor.set_framesize(sensor.QQVGA) # or sensor.QVGA (or
 others)
sensor.skip_frames(10) # Let new settings take affect.
```

```
clock = time.clock() # Tracks FPS.

while(True):
    clock.tick() # Track elapsed milliseconds between s
napshots().
    img = sensor.snapshot() # Take a picture and return
 the image.
```

唯一的参数是内核大小。N 对应于（（N * 2）+1）^ 2 内核大小。例如 1 == 3x3 内核，2 == 5x5 内核等。注意：你不应该使用大于 2 的值。

```
    img.mean(1)

    print(clock.fps()) # Note: Your OpenMV Cam runs abo
ut half as fast while
```

连接到您的计算机。一旦断开连接，FPS 应该增加。

5.4.6、彩色二进制过滤器

color_binary_filter

该脚本显示二进制图像过滤器。这个脚本最初是一个测试脚本，但它可以用于显示如何使用二进制文件。

```
import pyb, sensor, image, math

sensor.reset()
sensor.set_framesize(sensor.QVGA)
sensor.set_pixformat(sensor.RGB565)

red_threshold = (0,100,    0,127,    0,127) # L A B
green_threshold = (0,100,    -128,0,    0,127) # L A B
blue_threshold = (0,100,    -128,127,    -128,0) # L A B

while(True):
```

测试红色阈值

```
    for i in range(100):
        img = sensor.snapshot()
        img.binary([red_threshold])
```

测试绿色阈值

```
    for i in range(100):
        img = sensor.snapshot()
        img.binary([green_threshold])
```

测试蓝色阈值

```
    for i in range(100):
        img = sensor.snapshot()
        img.binary([blue_threshold])
```

测试非红色阈值

```
    for i in range(100):
        img = sensor.snapshot()
        img.binary([red_threshold], invert = 1)
```

测试非绿色阈值

```
    for i in range(100):
        img = sensor.snapshot()
        img.binary([green_threshold], invert = 1)
```

测试非蓝色阈值

```
    for i in range(100):
        img = sensor.snapshot()
        img.binary([blue_threshold], invert = 1)
```

5.4.7、模式过滤器

mode_filter

此示例显示关闭模式过滤。模式滤波是一种高度非线性的操作，它以每个像素的 NxN 邻域的模式替代每个像素。避免在 RGB565 图像上使用模式滤镜。它会在图像边缘造成伪像...

```
import sensor, image, time

sensor.reset() # Initialize the camera sensor.
sensor.set_pixformat(sensor.GRAYSCALE) # or sensor.RGB5
65
```

```
sensor.set_framesize(sensor.QQVGA) # or sensor.QVGA (or
 others)
sensor.skip_frames(10) # Let new settings take affect.
clock = time.clock() # Tracks FPS.

while(True):
    clock.tick() # Track elapsed milliseconds between s
napshots().
    img = sensor.snapshot() # Take a picture and return
 the image.
```

中值滤波器的唯一参数是内核大小，分别对于 1x1,3x3 或 5x5 内核可以是 0,1 或者 2。

```
    img.mode(1)

    print(clock.fps()) # Note: Your OpenMV Cam runs abo
ut half as fast while
```

连接到您的计算机。一旦断开连接，FPS 应该增加。

5.4.8、灰度二进制过滤器

grayscale_binary_filter

该脚本显示二进制图像过滤器。这个脚本最初是一个测试脚本，但它可以用于显示如何使用二进制文件。

```
import pyb, sensor, image, math

sensor.reset()
sensor.set_framesize(sensor.QVGA)
sensor.set_pixformat(sensor.GRAYSCALE)

low_threshold = (0, 50)
high_threshold = (205, 255)

while(True):
```

测试低阈值

```
    for i in range(100):
        img = sensor.snapshot()
        img.binary([low_threshold])
```

测试高阈值

```
    for i in range(100):
        img = sensor.snapshot()
        img.binary([high_threshold])
```

测试非低阈值

```
    for i in range(100):
        img = sensor.snapshot()
        img.binary([low_threshold], invert = 1)
```

Test not high threshold

```
    for i in range(100):
        img = sensor.snapshot()
        img.binary([high_threshold], invert = 1)
```

5.4.9、灰度过滤器

grayscale_filter

传感器模块可以在读取图像时进行一些基本的图像处理。此示例显示了如何应用灰度阈值。 未来！如果可能，请使用二进制函数进行图像分割。

```
import sensor, image, time

sensor.reset() # Initialize the camera sensor.
sensor.set_pixformat(sensor.RGB565) # or sensor.GRAYSCA
LE
sensor.set_framesize(sensor.QVGA) # or sensor.QQVGA (or
 others)
sensor.skip_frames(10) # Let new settings take affect.
clock = time.clock() # Tracks FPS.
```

按照阈值分割图像。这种分割是在读取图像时完成的，因此不会花费任何额外的时间...

```
sensor.set_image_filter(sensor.FILTER_BW, lower=128, up
per=255)

while(True):
```

```
    clock.tick() # Track elapsed milliseconds between s
napshots().
    img = sensor.snapshot() # Take a picture and return
 the image.
    print(clock.fps()) # Note: Your OpenMV Cam runs abo
ut half as fast while
    # connected to your computer. The FPS should increa
se once disconnected.
```

5.4.10、线性滤波

line_filter

传感器模块可以在图像读取期间进行一些基本图像处理，而无需额外的开销。这个例子说明了如何在 Python 中应用一些基本的行过滤器。

警告 - 在 Python 中执行行预处理时，此功能在 M4 上不起作用。在将来这可能会以某种方式固定，现在你会看到一个局部帧缓冲区。

```
import sensor, image, time
```

初始化相机传感器。

```
sensor.reset()
sensor.set_pixformat(sensor.GRAYSCALE)
sensor.set_framesize(sensor.QQVGA)
clock = time.clock() # Tracks FPS.
```

将源复制到目的地。 注意源是 YUYV 目的地是 1BPP 灰度

```
def line_filter_copy(src, dst):
    for i in range(0, len(dst), 1):
        dst[i] = src[i<<1]
```

按照阈值分割图像。 注意源是 YUYV 目的地是 1BPP 灰度

```
def line_filter_bw(src, dst):
    for i in range(0, len(dst), 1):
        if (src[i<<1] > 200 and src[i<<1] < 255):
            dst[i] = 0xFF
        else:
            dst[i] = 0x00
```

```
while(True):
    clock.tick() # Track elapsed milliseconds between s
napshots().
    lines = 0
    img = sensor.snapshot(line_filter = line_filter_cop
y) # Take a picture and return the image.

print(clock.fps()) # Note: Your OpenMV Cam runs about h
alf as fast while
```

连接到您的计算机时，OpenMV Cam 的运行速度大约一半。一旦断开连接，FPS 应该增加。

5.4.11、边缘检测

edge_detection

该示例演示了如何使用图像上的变形函数进行边缘 检测，然后对该图像进行阈值处理和过滤。

```
import sensor, image, time

kernel_size = 1 # kernel width = (size*2)+1, kernel hei
ght = (size*2)+1
kernel = [-1, -1, -1,\
          -1, +8, -1,\
          -1, -1, -1]
```

这是一个高通滤波器内核。看到更多的内核：
http://www.fmwconcepts.com/imagemagick/digital_image_filtering.pdf

```
thresholds = [(100, 255)] # grayscale thresholds

sensor.reset() # Initialize the camera sensor.
sensor.set_pixformat(sensor.GRAYSCALE) # or sensor.RGB5
65
sensor.set_framesize(sensor.QQVGA) # or sensor.QVGA (or
 others)
sensor.skip_frames(10) # Let new settings take affect.
clock = time.clock() # Tracks FPS.
```

在 OV7725 传感器上， 通过设置锐度/边沿寄存器可以显着增强边缘检测。 注意：稍后将作为功能实现。

```
if (sensor.get_id() == sensor.OV7725):
    sensor.__write_reg(0xAC, 0xDF)
    sensor.__write_reg(0x8F, 0xFF)

while(True):
    clock.tick() # Track elapsed milliseconds between s
napshots().
    img = sensor.snapshot() # Take a picture and return
 the image.

    img.morph(kernel_size, kernel)
    img.binary(thresholds)
```

使用 3x3 图像内核的小于 2 个邻居的 Erode 像素

```
    img.erode(1, threshold = 2)

    print(clock.fps()) # Note: Your OpenMV Cam runs abo
ut half as fast while
```

连接到您的计算机。一旦断开连接，FPS 应该增加。

5.4.12、锐化滤镜

sharpen_filter

此示例演示如何使用变体来锐化图像。

```
import sensor, image, time

kernel_size = 1 # kernel width = (size*2)+1, kernel hei
ght = (size*2)+1
kernel = [-1, -1, -1,\
          -1, +9, -1,\
          -1, -1, -1]
```

这是一个锐化的过滤器内核。

```
sensor.reset() # Initialize the camera sensor.
sensor.set_pixformat(sensor.GRAYSCALE) # or sensor.RGB5
65
sensor.set_framesize(sensor.QQVGA) # or sensor.QVGA (or
 others)
sensor.skip_frames(10) # Let new settings take affect.
```

```
clock = time.clock() # Tracks FPS.

while(True):
    clock.tick() # Track elapsed milliseconds between s
napshots().
    img = sensor.snapshot() # Take a picture and return
 the image.
```

在图像的每个像素上运行内核。

```
    img.morph(kernel_size, kernel)

    print(clock.fps()) # Note: Your OpenMV Cam runs abo
ut half as fast while
```

连接到您的计算机。一旦断开连接，FPS 应该增加。

5.4.13、高级图像帧区别

advanced_frame_differencing

注意：您需要一张 SD 卡才能运行此示例。

本示例演示了如何使用 OpenMV Cam 进行帧差分。这个 例子是高级的，因为它预设了一个后台更新来处理 反向绘制的图像超时。

```
import sensor, image, pyb, os, time

BG_UPDATE_FRAMES = 50 # How many frames before blendin
g.
BG_UPDATE_BLEND = 128 # How much to blend by... ([0-25
6]==[0.0-1.0]).

sensor.reset() # Initialize the camera sensor.
sensor.set_pixformat(sensor.GRAYSCALE) # or sensor.RGB5
65
sensor.set_framesize(sensor.QVGA) # or sensor.QQVGA (or
 others)
sensor.skip_frames(10) # Let new settings take affect.
sensor.set_auto_whitebal(False) # Turn off white balanc
e.
clock = time.clock() # Tracks FPS.
```

```
if not "temp" in os.listdir(): os.mkdir("temp") # Make
a temp directory

print("About to save background image...")
sensor.skip_frames(60) # Give the user time to get read
y.
sensor.snapshot().save("temp/bg.bmp")
print("Saved background image - Now frame differencing!
")

frame_count = 0
while(True):
    clock.tick() # Track elapsed milliseconds between s
napshots().
    img = sensor.snapshot() # Take a picture and return
 the image.

    frame_count += 1
    if frame_count > BG_UPDATE_FRAMES:
        frame_count = 0
```

混合在新的框架。我们在这里做 256-alpha，因为我们想 将新的框架融入背景。不是背景进入 新的框架，只是 alpha。混合通过（（新（alpha））+（OLD（256-alpha））） / 256 替换每个像素。因此，低 Alpha 会导致 新图像的混合不足，而高 Alpha 会导致 新图像的高混合。我们需要扭转这一更新。

```
        img.blend("temp/bg.bmp", alpha=(256-BG_UPDATE_B
LEND))
        img.save("temp/bg.bmp")
```

用“abs（NEW-OLD）”框架更换图像。 img.difference（“temp / bg.bmp”） img.difference("temp/bg.bmp")

```
    print(clock.fps()) # Note: Your OpenMV Cam runs abo
ut half as fast while
```

连接到您的计算机。一旦断开连接，FPS 应该增加。

5.5、05-拍照

5.5.1、人脸检测快照

snapshot_on_face_detection

注意：您需要一张 SD 卡才能运行此示例。

本示例演示如何使用 OpenMV Cam 上的脸部跟踪进行 拍摄。

```
import sensor, image, pyb

RED_LED_PIN = 1
BLUE_LED_PIN = 3

sensor.reset() # Initialize the camera sensor.
sensor.set_pixformat(sensor.GRAYSCALE)
sensor.set_framesize(sensor.HQVGA) # or sensor.QQVGA (o
r others)
sensor.skip_frames(10) # Let new settings take affect.
```

加载人脸检测 HaarCascade。这是您的 OpenMV 可以使用下面的 find_features （） 方法来检测脸部的对象。您的 OpenMV 内置 HaarCascade。默认情况下，HaarCascade 的所有阶段都被加载。但是，您可以调整阶段数，以加快处理的准确性。正面 HaarCascade 有 25 个阶段。

```
face_cascade = image.HaarCascade("frontalface", stages=
25)

while(True):

    pyb.LED(RED_LED_PIN).on()
    print("About to start detecting faces...")
    sensor.skip_frames(60) # Give the user time to get
ready.

    pyb.LED(RED_LED_PIN).off()
    print("Now detecting faces!")
    pyb.LED(BLUE_LED_PIN).on()

    diff = 10 # We'll say we detected a face after 10 f
rames.
    while(diff):
        img = sensor.snapshot()
```

阈值可以在 0.0 和 1.0 之间。更高的阈值导致更高的检测率，具有更多的假阳性。比例值控制匹配比例，允许您检测较小的面。

```
        faces = img.find_features(face_cascade, thresho
ld=0.5, scale_factor=1.5)

        if faces:
            diff -= 1
            for r in faces:
                img.draw_rectangle(r)

    pyb.LED(BLUE_LED_PIN).off()
    print("Face detected! Saving image...")
    sensor.snapshot().save("snapshot-%d.jpg" % pyb.rng
()) # Save Pic.
```

5.5.2、快照

snapshot

注意：您需要一张 SD 卡才能运行此示例。

您可以使用 OpenMV Cam 来保存图像文件。 import sensor, image, pyb

```
RED_LED_PIN = 1
BLUE_LED_PIN = 3

sensor.reset() # Initialize the camera sensor.
sensor.set_pixformat(sensor.RGB565) # or sensor.GRAYSCA
LE
sensor.set_framesize(sensor.QVGA) # or sensor.QQVGA (or
 others)
sensor.skip_frames(10) # Let new settings take affect.

pyb.LED(RED_LED_PIN).on()
sensor.skip_frames(30) # Give the user time to get read
y.

pyb.LED(RED_LED_PIN).off()
pyb.LED(BLUE_LED_PIN).on()

print("You're on camera!")
sensor.snapshot().save("example.jpg") # or "example.bmp
" (or others)

pyb.LED(BLUE_LED_PIN).off()
```

```
print("Done! Reset the camera to see the saved image.")
```

5.5.3、浮雕快照

emboss_snapshot

注意：您需要一张 SD 卡才能运行此示例。

您可以使用 OpenMV Cam 保存修改后的图像文件。

```
import sensor, image, pyb

RED_LED_PIN = 1
BLUE_LED_PIN = 3

sensor.reset() # Initialize the camera sensor.
sensor.set_pixformat(sensor.RGB565) # or sensor.GRAYSCA
LE
sensor.set_framesize(sensor.QVGA) # or sensor.QQVGA (or
 others)
sensor.skip_frames(10) # Let new settings take affect.

pyb.LED(RED_LED_PIN).on()
sensor.skip_frames(30) # Give the user time to get read
y.

pyb.LED(RED_LED_PIN).off()
pyb.LED(BLUE_LED_PIN).on()

print("You're on camera!")
img = sensor.snapshot()

img.morph(1, [+2, +1, +0,\
              +1, +1, -1,\
              +0, -1, -2]) # Emboss the image.

img.save("example.jpg") # or "example.bmp" (or others)

pyb.LED(BLUE_LED_PIN).off()
print("Done! Reset the camera to see the saved image.")
```

5.5.4、移动监测快照

snapshot_on_movement

注意：您需要一张 SD 卡才能运行此示例。

此示例演示如何使用 OpenMV 进行帧差分进行 运动检测。检测到运动后，您的 OpenMV 将拍摄照片。

```
import sensor, image, pyb, os

RED_LED_PIN = 1
BLUE_LED_PIN = 3

sensor.reset() # Initialize the camera sensor.
sensor.set_pixformat(sensor.RGB565) # or sensor.GRAYSCA
LE
sensor.set_framesize(sensor.QVGA) # or sensor.QQVGA (or
 others)
sensor.skip_frames(10) # Let new settings take affect.
sensor.set_auto_whitebal(False) # Turn off white balanc
e.

if not "temp" in os.listdir(): os.mkdir("temp") # Make
a temp directory

while(True):

    pyb.LED(RED_LED_PIN).on()
    print("About to save background image...")
    sensor.skip_frames(60) # Give the user time to get
ready.

    pyb.LED(RED_LED_PIN).off()
    sensor.snapshot().save("temp/bg.bmp")
    print("Saved background image - Now detecting motio
n!")
    pyb.LED(BLUE_LED_PIN).on()

    diff = 10 # We'll say we detected motion after 10 f
rames of motion.
    while(diff):
        img = sensor.snapshot()
        img.difference("temp/bg.bmp")
        stats = img.statistics()
        # Stats 5 is the max of the lighting color chan
nel. The below code
```

```
        # triggers when the lighting max for the whole
image goes above 20.
        # The lighting difference maximum should be zer
o normally.
        if (stats[5] > 20):
            diff -= 1

    pyb.LED(BLUE_LED_PIN).off()
    print("Movement detected! Saving image...")
    sensor.snapshot().save("temp/snapshot-%d.jpg" % py
b.rng()) # Save Pic.
```

5.6、06-视频录制

5.6.1、GIF 录制人脸检测

gif_on_face_detection

注意：您需要一张 SD 卡才能运行此示例。

您可以使用 OpenMV Cam 来记录 gif 文件。您可以为 记录器对象提供 RGB565 帧或灰度帧。使用 像 GIMP 这样的照片编辑软件可以在将 GIF 上传到网页之前对其进行压缩和优化。

此示例演示如何使用 OpenMV Cam 上的脸部跟踪来取得 gif。

```
import sensor, image, time, gif, pyb

RED_LED_PIN = 1
BLUE_LED_PIN = 3

sensor.reset() # Initialize the camera sensor.
sensor.set_pixformat(sensor.GRAYSCALE) # or sensor.
sensor.set_framesize(sensor.QQVGA) # or sensor.HQVGA (o
r others)
sensor.skip_frames(10) # Let new settings take affect.
```

加载人脸检测 HaarCascade。这是您的 OpenMV Cam 可以使用下面的 find_features（）方法来检测脸部的对象。您的 OpenMV Cam 内置了 HaarCascade。默认情况下， HaarCascade 的所有阶段都被加载。但是，您可以调整阶段数，以加快 处理的准确性。正面 HaarCascade 有 25 个 阶段。

```
face_cascade = image.HaarCascade("frontalface", stages=
25)

while(True):

    pyb.LED(RED_LED_PIN).on()
    print("About to start detecting faces...")
    sensor.skip_frames(60) # Give the user time to get
ready.

    pyb.LED(RED_LED_PIN).off()
    print("Now detecting faces!")
    pyb.LED(BLUE_LED_PIN).on()

    diff = 10 # We'll say we detected a face after 10 f
rames.
    while(diff):
        img = sensor.snapshot()
```

阈值可以在 0.0 和 1.0 之间。更高的阈值导致 更高的检测率，具有更多的假阳性。比例值 控制匹配比例，允许您检测较小的面。

```
        faces = img.find_features(face_cascade, thresho
ld=0.5, scale_factor=1.5)

        if faces:
            diff -= 1
            for r in faces:
                img.draw_rectangle(r)

    g = gif.Gif("example-%d.gif" % pyb.rng(), loop=Tru
e)

    clock = time.clock() # Tracks FPS.
    print("You're on camera!")
    for i in range(100):
        clock.tick()
```

clock.avg（）返回帧之间的毫秒 - gif delay

```
        g.add_frame(sensor.snapshot(), delay=int(clock.
avg()/10)) # centiseconds.
        print(clock.fps())
```

```
    g.close()
    pyb.LED(BLUE_LED_PIN).off()
    print("Restarting...")
```

5.6.2、GIF 视频录制示例

gif

注意：您需要一张 SD 卡才能运行此示例。

您可以使用 OpenMV Cam 来记录 gif 文件。您可以为 记录器对象提供 RGB565 帧或灰度帧。使用 像 GIMP 这样的照片编辑软件可以在将 GIF 上传到网页之前对其进行压缩和优化。

```
import sensor, image, time, gif, pyb

RED_LED_PIN = 1
BLUE_LED_PIN = 3

sensor.reset() # Initialize the camera sensor.
sensor.set_pixformat(sensor.RGB565) # or sensor.GRAYSCA
LE
sensor.set_framesize(sensor.QQVGA) # or sensor.QVGA (or
 others)
sensor.skip_frames(10) # Let new settings take affect.
clock = time.clock() # Tracks FPS.

pyb.LED(RED_LED_PIN).on()
sensor.skip_frames(30) # Give the user time to get read
y.

pyb.LED(RED_LED_PIN).off()
pyb.LED(BLUE_LED_PIN).on()

g = gif.Gif("example.gif", loop=True)

print("You're on camera!")
for i in range(100):
    clock.tick()
```

clock.avg（）返回帧之间的毫秒 - gif delay

```
    g.add_frame(sensor.snapshot(), delay=int(clock.avg
()/10)) # centiseconds.
```

```
    print(clock.fps())

g.close()
pyb.LED(BLUE_LED_PIN).off()
print("Done! Reset the camera to see the saved recordin
g.")
```

5.6.3、MJPEG 录像面部检测

mjpeg_on_face_detection

注意：您需要一张 SD 卡才能运行此示例。

您可以使用 OpenMV Cam 来记录 mjpeg 文件。您可以为 记录仪对象 JPEG 或 RGB565 /灰度帧提供。一旦你完成 录制 MJPEG 文件，您可以使用 VLC 播放。如果你在 Ubuntu，那么 内置的视频播放器也可以工作。

此示例演示如何在您的 OpenMV Cam 上使用脸部跟踪进行 mjpeg。

```
import sensor, image, time, mjpeg, pyb

RED_LED_PIN = 1
BLUE_LED_PIN = 3

sensor.reset() # Initialize the camera sensor.
sensor.set_pixformat(sensor.GRAYSCALE) # or sensor.
sensor.set_framesize(sensor.QQVGA) # or sensor.HQVGA (o
r others)
sensor.skip_frames(10) # Let new settings take affect.
```

加载人脸检测 HaarCascade。这是您的 OpenMV Cam 可以使用下面的 find_features （） 方法来检测脸部的对象。您的 OpenMV Cam 内置了 HaarCascade。默认情况下， HaarCascade 的所有阶段都被加载。但是，您可以调整阶段数，以加快 处理的准确性。正面 HaarCascade 有 25 个 阶段。

```
face_cascade = image.HaarCascade("frontalface", stages=
25)

while(True):

    pyb.LED(RED_LED_PIN).on()
    print("About to start detecting faces...")
```

```
    sensor.skip_frames(60) # Give the user time to get
ready.

    pyb.LED(RED_LED_PIN).off()
    print("Now detecting faces!")
    pyb.LED(BLUE_LED_PIN).on()

    diff = 10 # We'll say we detected a face after 10 f
rames.
    while(diff):
        img = sensor.snapshot()
```

阈值可以在 0.0 和 1.0 之间。更高的阈值导致 更高的检测率，具有更多的假阳性。比例值 控制匹配比例，允许您检测较小的面。

```
        faces = img.find_features(face_cascade, thresho
ld=0.5, scale_factor=1.5)

        if faces:
            diff -= 1
            for r in faces:
                img.draw_rectangle(r)

    m = mjpeg.Mjpeg("example-%d.mjpeg" % pyb.rng())

    clock = time.clock() # Tracks FPS.
    print("You're on camera!")
    for i in range(200):
        clock.tick()
        m.add_frame(sensor.snapshot())
        print(clock.fps())

    m.close(clock.fps())
    pyb.LED(BLUE_LED_PIN).off()
    print("Restarting...")
```

5.6.4、MJPEG 视频录制示例

mjpeg

注意：您需要一张 SD 卡才能运行此演示。

您可以使用 OpenMV Cam 来记录 mjpeg 文件。您可以为 记录仪对象 JPEG 或 RGB565 /灰度帧提供。一旦你完成 录制 MJPEG 文件，您可以

使用 VLC 播放。如果你在 Ubuntu，那么 内置的视频播放器也可以工作。

```
import sensor, image, time, mjpeg, pyb

RED_LED_PIN = 1
BLUE_LED_PIN = 3

sensor.reset() # Initialize the camera sensor.
sensor.set_pixformat(sensor.RGB565) # or sensor.GRAYSCA
LE
sensor.set_framesize(sensor.QVGA) # or sensor.QQVGA (or
 others)
sensor.skip_frames(10) # Let new settings take affect.
clock = time.clock() # Tracks FPS.

pyb.LED(RED_LED_PIN).on()
sensor.skip_frames(30) # Give the user time to get read
y.

pyb.LED(RED_LED_PIN).off()
pyb.LED(BLUE_LED_PIN).on()

m = mjpeg.Mjpeg("example.mjpeg")

print("You're on camera!")
for i in range(200):
    clock.tick()
    m.add_frame(sensor.snapshot())
    print(clock.fps())

m.close(clock.fps())
pyb.LED(BLUE_LED_PIN).off()
print("Done! Reset the camera to see the saved recordin
g.")
```

5.6.5、MJPEG 视频录制移动示例

mjpeg_on_movement

注意：您需要一张 SD 卡才能运行此示例。

您可以使用 OpenMV Cam 来记录 mjpeg 文件。您可以为 记录仪对象 JPEG 或 RGB565 /灰度帧提供。一旦你完成 录制 MJPEG 文件，您可以

使用 VLC 播放。如果你在 Ubuntu，那么 内置的视频播放器也可以工作。

此示例演示如何使用 OpenMV Cam 进行帧差分进行 运动检测。检测到运动后，您的 OpenMV 摄像头将拍摄视频。

```
import sensor, image, time, mjpeg, pyb, os

RED_LED_PIN = 1
BLUE_LED_PIN = 3

sensor.reset() # Initialize the camera sensor.
sensor.set_pixformat(sensor.RGB565) # or sensor.GRAYSCA
LE
sensor.set_framesize(sensor.QVGA) # or sensor.QQVGA (or
 others)
sensor.skip_frames(10) # Let new settings take affect.
sensor.set_auto_whitebal(False) # Turn off white balanc
e.

if not "temp" in os.listdir(): os.mkdir("temp") # Make
a temp directory

while(True):

    pyb.LED(RED_LED_PIN).on()
    print("About to save background image...")
    sensor.skip_frames(60) # Give the user time to get
ready.

    pyb.LED(RED_LED_PIN).off()
    sensor.snapshot().save("temp/bg.bmp")
    print("Saved background image - Now detecting motio
n!")
    pyb.LED(BLUE_LED_PIN).on()

    diff = 10 # We'll say we detected motion after 10 f
rames of motion.
    while(diff):
        img = sensor.snapshot()
        img.difference("temp/bg.bmp")
        stats = img.statistics()
```

统计 5 是照明色彩通道的最大值。 当整个图像的照明最大值超过 20 时，以下代码将触发。 照明差异最大值应正常为零。

```
        if (stats[5] > 20):
            diff -= 1

    m = mjpeg.Mjpeg("example-%d.mjpeg" % pyb.rng())

    clock = time.clock() # Tracks FPS.
    print("You're on camera!")
    for i in range(200):
        clock.tick()
        m.add_frame(sensor.snapshot())
        print(clock.fps())

    m.close(clock.fps())
    pyb.LED(BLUE_LED_PIN).off()
    print("Restarting...")
```

5.6.6、视频录制移动示例

gif_on_movementGIF

注意：您需要一张 SD 卡才能运行此示例。

您可以使用 OpenMV Cam 来记录 gif 文件。您可以为 记录器对象提供 RGB565 帧或灰度帧。使用 像 GIMP 这样的照片编辑软件可以在将 GIF 上传到网页之前对其进行压缩和优化。

此示例演示如何使用 OpenMV Cam 进行帧差分进行 运动检测。检测到运动后，您的 OpenMV 摄像头将拍摄视频。

```
import sensor, image, time, gif, pyb, os

RED_LED_PIN = 1
BLUE_LED_PIN = 3

sensor.reset() # Initialize the camera sensor.
sensor.set_pixformat(sensor.RGB565) # or sensor.GRAYSCA
LE
sensor.set_framesize(sensor.QQVGA) # or sensor.QVGA (or
 others)
sensor.skip_frames(10) # Let new settings take affect.
sensor.set_auto_whitebal(False) # Turn off white balanc
e.
```

```
if not "temp" in os.listdir(): os.mkdir("temp") # Make
a temp directory

while(True):

    pyb.LED(RED_LED_PIN).on()
    print("About to save background image...")
    sensor.skip_frames(60) # Give the user time to get
ready.

    pyb.LED(RED_LED_PIN).off()
    sensor.snapshot().save("temp/bg.bmp")
    print("Saved background image - Now detecting motio
n!")
    pyb.LED(BLUE_LED_PIN).on()

    diff = 10 # We'll say we detected motion after 10 f
rames of motion.
    while(diff):
        img = sensor.snapshot()
        img.difference("temp/bg.bmp")
        stats = img.statistics()
```

统计 5 是照明色彩通道的最大值。 当整个图像的照明最大值超过 20 时，以下代码将触发。 照明差异最大值应正常为零。

```
        if (stats[5] > 20):
            diff -= 1

    g = gif.Gif("example-%d.gif" % pyb.rng(), loop=Tru
e)

    clock = time.clock() # Tracks FPS.
    print("You're on camera!")
    for i in range(100):
        clock.tick()
```

clock.avg () 返回帧之间的毫秒 - gif delay
g.add_frame(sensor.snapshot(), delay=int(clock.avg()/10)) # centiseconds. print(clock.fps())

```
    g.close()
    pyb.LED(BLUE_LED_PIN).off()
```

```
    print("Restarting...")
```

5.7、07-人脸识别

5.7.1、LBP 面部识别

face_recognition

参见 Timo Ahonen 的“用本地二进制模式识别面部识别”。

运行示例之前： 1）下载 AT&T 面数据库 http://www.cl.cam.ac.uk/Research/DTG/attarchive/pub/data/att_faces.zip 2）解压并将 orl_faces 目录复制到 SD 卡根目录。

```
import sensor, time, image

SUB = "s2"
NUM_SUBJECTS = 5
NUM_SUBJECTS_IMGS = 10

img = image.Image("orl_faces/%s/1.pgm"%(SUB)).mask_ellipse()
d0 = img.find_lbp((0, 0, img.width(), img.height()))
img = None

print("")
for s in range(1, NUM_SUBJECTS+1):
    dist = 0
    for i in range(2, NUM_SUBJECTS_IMGS+1):
        img = image.Image("orl_faces/s%d/%d.pgm"%(s, i)).mask_ellipse()
        d1 = img.find_lbp((0, 0, img.width(), img.height()))
        dist += image.match_descriptor(d0, d1)
    print("Average dist for subject %d: %d"%(s, dist/NUM_SUBJECTS_IMGS))
```

5.7.2、面部检测示例

face_detection

此示例说明 OpenMV Cam 的内置面部检测功能。

通过使用 Haar Cascade 特征检测器在图像上进行面部检测。一个 Haar Cascade 是一系列简单的区域对比检查。对于内置 frontalface 检测器有 25 个阶段的检查每个阶段有 数百个检查一块。哈尔瀑布运行速度快，因为后期阶段 仅在以前的阶段通过时进行评估。此外，您的 OpenMV Cam 使用 一个称为积分图像的数据结构，可以快速执行每个区域 对比检查恒定时间（特征检测的原因 灰度只是因为积分图像的空间要求）。

```
import sensor, time, image
```

复位传感器

```
sensor.reset()
```

传感器设置

```
sensor.set_contrast(1)
sensor.set_gainceiling(16)
```

HQVGA 和 GRAYSCALE 是最好的面部跟踪。

```
sensor.set_framesize(sensor.HQVGA)
sensor.set_pixformat(sensor.GRAYSCALE)
```

加载哈尔级联 默认情况下，这将使用所有阶段，较低的 satges 是更快，但不太准确。

```
face_cascade = image.HaarCascade("frontalface", stages=
25)
print(face_cascade)
```

FPS 时钟

```
clock = time.clock()

while (True):
    clock.tick()
```

捕获快照

```
    img = sensor.snapshot()
```

查找对象。 注意：较低比例因子会缩小图像并检测较小的对象。 更高的阈值导致更高的检测率，更多的假阳性。

```
    objects = img.find_features(face_cascade, threshold
=0.75, scale_factor=1.25)
```

绘制对象

```
    for r in objects:
        img.draw_rectangle(r)
```

打印 FPS。 注意：实际的 FPS 更高，流 FB 使它更慢。

```
    print(clock.fps())
```

5.7.3、面部跟踪示例

face_tracking

此示例显示使用 OpenMV Cam 的关键点功能进行跟踪 Haar Cascade 检测到一个面孔。这第一部分 script 使用前面 Haar Cascade 在图像中查找一个面。 之后，脚本使用关键点功能自动学习您的 面并跟踪它。关键点可用于自动跟踪任何内容。

```
import sensor, time, image
```

复位传感器

```
sensor.reset()
sensor.set_contrast(3)
sensor.set_gainceiling(16)
sensor.set_framesize(sensor.VGA)
sensor.set_windowing((320, 240))
sensor.set_pixformat(sensor.GRAYSCALE)
```

跳过几个帧，以允许传感器稳定下来

```
sensor.skip_frames(60)
```

加载哈尔级联 默认情况下，这将使用所有阶段，较低的 satges 是更快，但不太准确。

```
face_cascade = image.HaarCascade("frontalface", stages=
25)
print(face_cascade)
```

第一组关键点

```
kpts1 = None
```

找到一张脸!

```
while (kpts1 == None):
    img = sensor.snapshot()
    img.draw_string(0, 0, "Looking for a face...")
```

查找面孔

```
    objects = img.find_features(face_cascade, threshold
=0.5, scale=1.25)
    if objects:
```

在每个方向上将 ROI 投影 31 个像素

```
        face = (objects[0][0]-31, objects[0][1]-31,obje
cts[0][2]+31*2, objects[0][3]+31*2)
```

使用检测面部大小作为 ROI 提取关键点

```
        kpts1 = img.find_keypoints(threshold=10, scale_
factor=1.1, max_keypoints=100, roi=face)
```

在第一个面的周围绘制一个矩形

```
        img.draw_rectangle(objects[0])
```

绘制关键点

```
print(kpts1)
img.draw_keypoints(kpts1, size=24)
img = sensor.snapshot()
time.sleep(2000)
```

FPS 时钟

```
clock = time.clock()

while (True):
    clock.tick()
    img = sensor.snapshot()
```

从整个框架中提取关键点

```
    kpts2 = img.find_keypoints(threshold=10, scale_factor=1.1, max_keypoints=100, normalized=True)
    if (kpts2):
```

将第一组关键点与第二组关键点进行匹配

```
        c=image.match_descriptor(kpts1, kpts2, threshold=85)
        match = c[6] # C[6] contains the number of matches.
        if (match>5):
            img.draw_rectangle(c[2:6])
            img.draw_cross(c[0], c[1], size=10)
            print(kpts2, "matched:%d dt:%d"%(match, c[7]))
```

绘制 FPS

```
    img.draw_string(0, 0, "FPS:%.2f"%(clock.fps()))
```

5.8、08-眼球跟踪

5.8.1、虹膜检测

iris_detection

这个例子显示了 在图像中发现眼睛后如何找到眼睛注视（瞳孔检测）。该脚本使用 find_eyes 函数来确定 包含瞳孔的 roi 的中心点。它通过基本上 找到作为学生中心的眼睛中最黑暗的区域的中心来实现。

注意：此脚本首先未检测到脸部，使用长焦镜头。

```
import sensor, time, image
```

复位传感器

```
sensor.reset()
```

传感器设置

```
sensor.set_contrast(3)
sensor.set_gainceiling(16)
```

将分辨率设置为 VGA。

```
sensor.set_framesize(sensor.VGA)
```

Bin / Crop 图像为 200x100，它提供更多的细节，更少的数据要处理

```
sensor.set_windowing((220, 190, 200, 100))

sensor.set_pixformat(sensor.GRAYSCALE)
```

加载哈尔级联

默认情况下，这将使用所有阶段，较低的饱和度更快，但不太准确。

```
eyes_cascade = image.HaarCascade("eye", stages=24)
print(eyes_cascade)
```

FPS 时钟

```
clock = time.clock()

while (True):
    clock.tick()
```

捕获快照

```
    img = sensor.snapshot()
```

找眼睛！

注意：较小比例因子会缩小图像，并检测较小的物体。 更高的阈值导致更高的检测率，具有更多的假阳性。 eyes = img.find_features(eyes_cascade, threshold=0.5, scale_factor=1.5)

找虹膜

```
    for e in eyes:
        iris = img.find_eye(e)
        img.draw_rectangle(e)
        img.draw_cross(iris[0], iris[1])
```

打印 FPS。 注意：实际 FPS 更高，流 FB 使其更慢。

```
    print(clock.fps())
```

5.8.2、面部眼睛检测示例

face_eye_detection

该脚本使用内置的前端检测器找到一张脸，然后找到面部 内的眼睛。如果您想确定眼睛凝视，请参阅 iris_detection 脚本，了解如何做到这一点。

```
import sensor, time, image
```

传感器复位传感器

```
sensor.reset()
```

传感器设置

```
sensor.set_contrast(1)
sensor.set_gainceiling(16)
sensor.set_framesize(sensor.HQVGA)
sensor.set_pixformat(sensor.GRAYSCALE)
```

加载哈尔级联

```
# By default this will use all stages, lower satges is
faster but less accurate.
face_cascade = image.HaarCascade("frontalface", stages=
25)
eyes_cascade = image.HaarCascade("eye", stages=24)
print(face_cascade, eyes_cascade)
```

FPS 时钟

```
clock = time.clock()

while (True):
    clock.tick()
```

抓拍快照

```
    img = sensor.snapshot()
```

找一张脸！ 注意：较小比例因子会缩小图像，并检测较小的物体。更高的阈值导致更高的检测率，具有更多的假阳性。

```
    objects = img.find_features(face_cascade, threshold
=0.5, scale_factor=1.5)
```

绘制面部

```
    for face in objects:
        img.draw_rectangle(face)
```

现在在每张脸上找到眼睛。 注意：在这里使用更高的阈值（更多的检测）和较小的比例（找到小对象）

```
        eyes = img.find_features(eyes_cascade, threshol
d=0.5, scale_factor=1.2, roi=face)
        for e in eyes:
            img.draw_rectangle(e)
```

打印 FPS。 注意：实际 FPS 更高，流 FB 使其更慢。

```
    print(clock.fps())
```

5.9、09-特征检测

5.9.1、Canny 和 Hough 变换

lines

这个例子演示了使用 Canny 边缘检测器 和霍夫变换在图像中找到直线。

```
import sensor, image, time
sensor.reset() # Initialize the camera sensor.
sensor.set_pixformat(sensor.GRAYSCALE) # or sensor.RGB5
65
sensor.set_framesize(sensor.QQVGA) # or sensor.QVGA (or
 others)
clock = time.clock() # Tracks FPS.

while(True):
    clock.tick() # Track elapsed milliseconds between s
napshots().
    img = sensor.snapshot() # Take a picture and return
 the image.
    img.find_edges(image.EDGE_CANNY, threshold=(50, 8
0))  # Find edges
```

```
    lines = img.find_lines(threshold=50) # Find lines.
    for l in lines: img.draw_line(l, color=(127)) # Dra
w lines
    print(clock.fps()) # Note: Your OpenMV Cam runs abo
ut half as fast while
```

5.9.2、HoG

hog

定向梯度直方图（HoG）示例

这个例子演示了 HoG 可视化。

注意：由于 JPEG 工件，HoG 可视化看起来模糊。要查看 没有 JPEG 工件的图像，请将保存图像的行注释为 uSD。

```
import sensor, image, time

sensor.reset()
```

设置传感器设置

```
sensor.set_contrast(1)
sensor.set_gainceiling(8)
sensor.set_framesize(sensor.QVGA)
sensor.skip_frames(30)
sensor.set_pixformat(sensor.GRAYSCALE)

clock = time.clock() # Tracks FPS.
while (True):
    clock.tick()
    img = sensor.snapshot()
    img.find_hog()
```

取消注释保存原始 FB 到文件并退出循环 img.save（“/hog.pgm”） break

```
    print(clock.fps())
```

5.9.3、光流示例

optical_flow

您的 OpenMV Cam 可以使用光流来确定 两个图像之间的位移。这允许您的 OpenMV 摄像机跟踪您的激光 鼠标跟踪移动的运动。通过解决连续图像之间的差异， 您可以使用 OpenMV 摄像机确定本体位移！

```
import sensor, image, time

sensor.reset() # Initialize the camera sensor.
sensor.set_pixformat(sensor.GRAYSCALE) # or sensor.GRAY
SCALE
sensor.set_framesize(sensor.B64x32) # or B40x30 or B64x
64
clock = time.clock() # Tracks FPS.
```

注意：find_displacement 函数通过获取旧 图像和新图像的 2D FFT 进行工作，并使用相位相关进行比较。您的 OpenMV Cam 只有足够的内存可以在两个 64x64 的 FFT（或 128x32,32x128 或其他）上工作。

```
old = sensor.snapshot()

while(True):
    clock.tick() # Track elapsed milliseconds between s
napshots().
    img = sensor.snapshot() # Take a picture and return
 the image.

    [delta_x, delta_y, response] = old.find_displacemen
t(img)

    old = img.copy()

    print("%0.1f X\t%0.1f Y\t%0.2f QoR\t%0.2f FPS" % \
        (delta_x, delta_y, response, clock.fps()))
```

5.9.4、关键点对象跟踪

keypoints 以关键点为例的对象跟踪。 向相机显示一个对象，然后运行该脚本。一组关键点将被提取 一次，然后在以下帧中进行跟踪。如果您想要一组新的关键点重新运行 该脚本。注意：有关参数的文档，请参阅 find_keypoints 和 match_keypoints。

```
import sensor, time, image
```

复位传感器

```
sensor.reset()
```

传感器设置

```
sensor.set_contrast(3)
sensor.set_gainceiling(16)
sensor.set_framesize(sensor.VGA)
sensor.set_windowing((320, 240))
sensor.set_pixformat(sensor.GRAYSCALE)

sensor.skip_frames(30)
sensor.set_auto_gain(False, value=100)

def draw_keypoints(img, kpts):
    print(kpts)
    img.draw_keypoints(kpts)
    img = sensor.snapshot()
    time.sleep(1000)

kpts1 = None
```

注意：从文件 kpts1 = image.load_descriptor（“/ desc.orb”）加载关键点描述符的注释 img = sensor.snapshot（） draw_keypoints（img，kpts1） clock = time.clock() while (True): clock.tick() img = sensor.snapshot() if (kpts1 == None):

注意：默认情况下，find_keypoints 返回从图像金字塔提取的多尺度关键点。

```
        kpts1 = img.find_keypoints(max_keypoints=150, t
hreshold=10, scale_factor=1.2)
        draw_keypoints(img, kpts1)
    else:
```

注意：当提取关键点以匹配第一个描述符时，我们使用 normalized = True 从第一个缩放中提取关键点，这将与第一个描述符中的一个比例匹配。

```
        kpts2 = img.find_keypoints(max_keypoints=150, t
hreshold=10, normalized=True)
        if (kpts2):
            c = image.match_descriptor(kpts1, kpts2, th
reshold=85)
```

```
            match = c[6] # C[6] contains the number of
matches.
            if (match>10):
                img.draw_rectangle(c[2:6])
                img.draw_cross(c[0], c[1], size=10)

            print(kpts2, "matched:%d dt:%d"%(match, c
[7]))
```

注意：如果要绘制关键点 img.draw_keypoints（kpts2，size = KEYPOINTS_SIZE，matched = True），请取消注释

绘制 FPS

```
    img.draw_string(0, 0, "FPS:%.2f"%(clock.fps()))
```

5.9.5、关键点描述符示例

keypoints_save 此示例显示如何将关键点描述符保存到文件。向相机显示一个对象 ，然后运行该脚本。脚本将提取并保存关键点描述符和图像。 您可以使用 keypoints_editor.py util 来删除不需要的关键点。 注意：请在运行此脚本后重新启动相机以查看新文件。

```
import sensor, time, image
```

复位传感器

```
sensor.reset()
```

传感器设置

```
sensor.set_contrast(3)
sensor.set_gainceiling(16)
sensor.set_framesize(sensor.VGA)
sensor.set_windowing((320, 240))
sensor.set_pixformat(sensor.GRAYSCALE)

sensor.skip_frames(30)
sensor.set_auto_gain(False, value=100)

FILE_NAME = "desc"
img = sensor.snapshot()
```

注意：请参阅文档了解其他参数 注意：默认情况下，
find_keypoints 返回从图像金字塔提取的多尺度关键点。

```
kpts = img.find_keypoints(max_keypoints=150, threshold=
10, scale_factor=1.2)

if (kpts == None):
    raise(Exception("Couldn't find any keypoints!"))

image.save_descriptor(kpts, "/%s.orb"%(FILE_NAME))
img.save("/%s.pgm"%(FILE_NAME))

img.draw_keypoints(kpts)
sensor.snapshot()
time.sleep(1000)
raise(Exception("Done! Please reset the camera"))
```

5.9.6、局部二进制模式（LBP）示例

lbp

此示例说明如何 在 OpenMV Cam 上使用本地二进制模式特征描述符。
LBP 描述符像 Freak 特征描述符一样工作。

警告：LBP 支持需要重做！到目前为止，这个功能需要 大量的工作
才能使之成为有用的东西。该脚本将重新显示 功能存在，但在当前
状态不足。

```
import sensor, time, image
sensor.reset()
```

复位传感器

```
sensor.reset()

# Sensor settings
sensor.set_contrast(1)
sensor.set_gainceiling(16)
sensor.set_framesize(sensor.HQVGA)
sensor.set_pixformat(sensor.GRAYSCALE)
```

加载哈尔级联

默认情况下，这将使用所有阶段，较低的饱和度更快，但不太准确。

```
face_cascade = image.HaarCascade("frontalface", stages=
25)
print(face_cascade)
```

跳过几帧以使传感器稳定下来 注意：从 IDE 中需要更多时间。

```
for i in range(0, 30):
    img = sensor.snapshot()
    img.draw_string(0, 0, "Please wait...")

d0 = None

d0 = image.load_descriptor("/desc.lbp")

clock = time.clock()

while (True):
    clock.tick()
    img = sensor.snapshot()

    objects = img.find_features(face_cascade, threshold
=0.5, scale_factor=1.25)
    if objects:
        face = objects[0]
        d1 = img.find_lbp(face)
        if (d0 == None):
            d0 = d1
        else:
            dist = image.match_descriptor(d0, d1)
            img.draw_string(0, 10, "Match %d%%"%(dist))

        img.draw_rectangle(face)
```

绘制 FPS

```
    img.draw_string(0, 0, "FPS:%.2f"%(clock.fps()))
```

5.9.7、模板匹配

template_matching

此示例显示如何使用 OpenMV Cam 的 NCC 功能进行匹配 图像的一部分的图像...极度受控的环境因素 NCC 不是全部能识别。

警告：NCC 支持需要重做！至于现在这个功能需要 很多工作要做成更好用的。这个脚本会 reamine 显示 该功能存在，但是，在其当前状态并不太完整。

```
import time, sensor, image
from image import SEARCH_EX, SEARCH_DS
```

复位传感器

```
sensor.reset()
```

设置传感器参数

```
sensor.set_contrast(1)
sensor.set_gainceiling(16)
```

与 SEARCH_EX 模板匹配的最大分辨率为 QQVGA

```
sensor.set_framesize(sensor.QQVGA)
```

＃您可以设置窗口缩小搜索图像。 sensor.set_windowing(((640-80)//2, (480-60)//2, 80, 60))

```
sensor.set_pixformat(sensor.GRAYSCALE)
```

加载模板。 模板应该是一个小（例如，32x32 像素）灰度图像。

```
template = image.Image("/template.pgm")
clock = time.clock()
```

运行模板匹配

```
while (True):
    clock.tick()
    img = sensor.snapshot()
```

find_template（template，threshold，[roi，step，search]）

ROI：感兴趣区域 tuple（x，y，w，h）。 Step：使用的循环步骤（y + =Step，x + =Step）使用更大的步骤，使其更快。 搜索既可以是 image.SEARCH_EX 用于详尽搜索，也可以使用 image.SEARCH_DS 用于菱形搜索 Note1：ROI 必须小于图像并大于模板。 Note2：在菱形搜索中，步长和 ROI 都将被忽略。

```
    r = img.find_template(template, 0.70, step=4, searc
h=SEARCH_EX) #roi=(10, 0, 60, 60))
    if r:
        img.draw_rectangle(r)

    print(clock.fps())
```

由于我们的模板大小超出 openmv 内置的 flash，需要插上 sd 卡后进行下列步骤。

先运行 helloworld.py 例程，让 frambuffer 显示出图像，然后进行截取需要识别的图片。

选择 save image selection to pc，注意从 openmv 里面直接截取保存的图片是 bmp 格式的，我们需要把它转换成 pgm 格式。
https://convertio.co/zh/bmp-pgm/

然后，我们将转换完的 pgm 模板保存到 sd 卡（根目录）。

然后我们打开模板匹配的例程，修改模板的名称为保存的模板名称。这样模板识别就实现了！

5.9.8、边缘检测

edges

这个例子演示了 Canny 边缘检测器。

```
import sensor, image, time

sensor.reset() # Initialize the camera sensor.
sensor.set_pixformat(sensor.GRAYSCALE) # or sensor.RGB5
65
sensor.set_framesize(sensor.QQVGA) # or sensor.QVGA (or
 others)
sensor.skip_frames(30) # Let new settings take affect.
sensor.set_gainceiling(8)

clock = time.clock() # Tracks FPS.
while(True):
    clock.tick() # Track elapsed milliseconds between s
napshots().
```

```
    img = sensor.snapshot() # Take a picture and return
 the image.
```

使用 Canny 边缘检测器

```
    img.find_edges(image.EDGE_CANNY, threshold=(50, 8
0))
```

更简单的边缘检测

```
    mg.find_edges(image.EDGE_SIMPLE, threshold=(100, 25
5))
    print(clock.fps()) # Note: Your OpenMV Cam runs abo
ut half as fast while
```

5.10、10-颜色跟踪

5.10.1、单色 RGB565 Blob 跟踪示例

single_color_rgb565_blob_tracking

该示例显示使用 OpenMV Cam 的单色 RGB565 跟踪。

```
import sensor, image, time

threshold_index = 0 # 0 for red, 1 for green, 2 for blu
e
```

颜色跟踪阈值（L Min，L Max，A Min，A Max，B Min，B Max） 以下阈值通常记录红/绿/蓝色物体。你可能希望调整他们...

```
thresholds = [(30, 100, 15, 127, 15, 127), # generic_re
d_thresholds
              (30, 100, -64, -8, -32, 32), # generic_gr
een_thresholds
              (0, 30, 0, 64, -128, 0)] # generic_blue_t
hresholds

sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QVGA)
sensor.skip_frames(30)
sensor.set_auto_gain(False) # must be turned off for co
lor tracking
```

```
sensor.set_auto_whitebal(False) # must be turned off fo
r color tracking
clock = time.clock()
```

只有比“pixel_threshold”更多的像素和比“area_threshold”更多的区域的斑点才能 通过下面的“find_blob”返回。如果更改 摄像机分辨率，请更改“pixels_threshold”和 “area_threshold” 。“merge = True”合并图像中的所有重叠的斑点。

```
while(True):
    clock.tick()
    img = sensor.snapshot()
    for blob in img.find_blobs([thresholds[threshold_in
dex]], pixels_threshold=200, area_threshold=200, merge=
True):
        img.draw_rectangle(blob.rect())
        img.draw_cross(blob.cx(), blob.cy())
    print(clock.fps())
```

5.10.2、单色代码跟踪示例

single_color_code_tracking

本示例使用 OpenMV Cam 显示单色代码跟踪。

颜色代码是由两种或多种颜色组成的斑点。下面的例子将 只跟踪有颜色的对象，它们中都有下面的颜色。

```
import sensor, image, time
```

颜色跟踪阈值（L Min，L Max，A Min，A Max，B Min，B Max） 以下阈值跟踪一般的红色/绿色物体。你可能希望调整他们...

```
thresholds = [(30, 100, 15, 127, 15, 127), # generic_re
d_thresholds -> index is 0 so code == (1 << 0)
              (30, 100, -64, -8, -32, 32)] # generic_gr
een_thresholds -> index is 1 so code == (1 << 1)
```

当“find_blob”的“merge = True”时，代码被合并在一起。

```
sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QVGA)
```

```
sensor.skip_frames(30)
sensor.set_auto_gain(False) # must be turned off for co
lor tracking
sensor.set_auto_whitebal(False) # must be turned off fo
r color tracking
clock = time.clock()
```

只有比“pixel_threshold”更多的像素和比“area_threshold”更多的区域的斑点才能 通过下面的“find_blob”返回。如果更改 摄像机分辨率，请更改“pixels_threshold”和 “area_threshold” 。必须将“merge = True”设置为合并颜色代码的重叠颜色斑点。

```
while(True):
    clock.tick()
    img = sensor.snapshot()
    for blob in img.find_blobs(thresholds, pixels_thres
hold=100, area_threshold=100, merge=True):
        if blob.code() == 3: # r/g code == (1 << 1) |
(1 << 0)
            img.draw_rectangle(blob.rect())
            img.draw_cross(blob.cx(), blob.cy())
    print(clock.fps())
```

5.10.3、单色灰度 Blob 跟踪示例

single_color_grayscale_blob_tracking

该示例使用 OpenMV Cam 显示单色灰度跟踪。

```
import sensor, image, time
```

颜色跟踪阈值（灰度最小值，灰度最大值） 下面的灰度阈值设置为仅发现非常明亮的白色区域。

```
thresholds = (245, 255)

sensor.reset()
sensor.set_pixformat(sensor.GRAYSCALE)
sensor.set_framesize(sensor.VGA)
sensor.skip_frames(30)
sensor.set_auto_gain(False) # must be turned off for co
lor tracking
```

```
sensor.set_auto_whitebal(False) # must be turned off fo
r color tracking
clock = time.clock()
```

只有比“pixel_threshold”更多的像素和比“area_threshold”更多的区域的斑点才能 通过下面的“find_blob”返回。如果更改 摄像机分辨率，请更改“pixels_threshold”和 “area_threshold” 。“merge = True”合并图像中的所有重叠的斑点。

```
while(True):
    clock.tick()
    img = sensor.snapshot()
    for blob in img.find_blobs([thresholds], pixels_thr
eshold=100, area_threshold=100, merge=True):
        img.draw_rectangle(blob.rect())
        img.draw_cross(blob.cx(), blob.cy())
    print(clock.fps())
```

5.10.4、图像直方图信息示例

image_histogram_info

该脚本计算图像的直方图并将其打印出来。

```
import sensor, image, time

sensor.reset()
sensor.set_pixformat(sensor.GRAYSCALE) # or RGB565.
sensor.set_framesize(sensor.QVGA)
sensor.skip_frames(30)
sensor.set_auto_gain(False) # must be turned off for co
lor tracking
sensor.set_auto_whitebal(False) # must be turned off fo
r color tracking
clock = time.clock()

while(True):
    clock.tick()
    img = sensor.snapshot()
    # Gets the grayscale histogram for the image into 8
 bins.
    # Bins defaults to 256 and may be between 2 and 25
6.
```

```
    print(img.get_histogram(bins=8))
    print(clock.fps())
```

您也可以通过 get_histogram（）“roi =”来获取该区域的直方图。 get_histogram（）允许您快速确定 图像中任何区域的颜色通道信息。

5.10.5、图像统计信息示例

image_statistics_info

该脚本计算图像的统计信息并打印出来。

```
import sensor, image, time

sensor.reset()
sensor.set_pixformat(sensor.GRAYSCALE) # or RGB565.
sensor.set_framesize(sensor.QVGA)
sensor.skip_frames(30)
sensor.set_auto_gain(False) # must be turned off for co
lor tracking
sensor.set_auto_whitebal(False) # must be turned off fo
r color tracking
clock = time.clock()

while(True):
    clock.tick()
    img = sensor.snapshot()
    print(img.get_statistics())
    print(clock.fps())
```

您也可以通过 get_statistics（）一个“roi =”来获取该区域的统计信息。 get_statistics（）允许您快速确定 图像中任何区域的颜色通道信息。

5.10.6、彩色 Blob 跟踪示例

multi_color_blob_tracking

此示例使用 OpenMV Cam 显示多色斑点跟踪。

```
import sensor, image, time
```

颜色跟踪阈值（L Min，L Max，A Min，A Max，B Min，B Max） 以下阈值跟踪一般的红色/绿色物体。你可能希望调整他们...

```
thresholds = [(30, 100, 15, 127, 15, 127), # generic_red_thresholds
              (30, 100, -64, -8, -32, 32), # generic_green_thresholds
              (0, 15, 0, 40, -80, -20)] # generic_blue_thresholds
```

您可以传达上述 16 个阈值。然而， 在颜色阈值开始重叠之前，并不真正可以将具有 16 个阈值的任何场景分段。

```
sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QVGA)
sensor.skip_frames(30)
sensor.set_auto_gain(False) # must be turned off for color tracking
sensor.set_auto_whitebal(False) # must be turned off for color tracking
clock = time.clock()
```

只有比“pixel_threshold”更多的像素和比“area_threshold”更多的区域的斑点才能 通过下面的“find_blob”返回。如果更改 摄像机分辨率，请更改“pixels_threshold”和 “area_threshold” 。不要设置“merge = True”，因为这将合并我们不希望的 blob。

```
while(True):
    clock.tick()
    img = sensor.snapshot()
    for blob in img.find_blobs(thresholds, pixels_threshold=200, area_threshold=200):
        img.draw_rectangle(blob.rect())
        img.draw_cross(blob.cx(), blob.cy())
    print(clock.fps())
```

5.10.7、彩色代码跟踪示例

multi_color_code_tracking

该示例使用 OpenMV Cam 显示多色代码跟踪。

颜色代码是由两种或多种颜色组成的斑点。下面的例子将 只跟踪下面有两个或更多颜色的彩色对象。

```
import sensor, image, time
```

颜色跟踪阈值（L Min，L Max，A Min，A Max，B Min，B Max） 以下阈值跟踪一般的红色/绿色物体。你可能希望调整他们...

```
thresholds = [(30, 100, 15, 127, 15, 127), # generic_red_thresholds -> index is 0 so code == (1 << 0)
              (30, 100, -64, -8, -32, 32), # generic_green_thresholds -> index is 1 so code == (1 << 1)
              (0, 15, 0, 40, -80, -20)] # generic_blue_thresholds -> index is 2 so code == (1 << 2)
```

当“find_blob”的“merge = True”时，代码被合并在一起。

```
sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QVGA)
sensor.skip_frames(30)
sensor.set_auto_gain(False) # must be turned off for color tracking
sensor.set_auto_whitebal(False) # must be turned off for color tracking
clock = time.clock()
```

只有比“pixel_threshold”更多的像素和比“area_threshold”更多的区域的斑点才能 通过下面的“find_blob”返回。如果更改 摄像机分辨率，请更改“pixels_threshold”和 “area_threshold” 。必须将“merge = True”设置为合并颜色代码的重叠颜色斑点。

```
while(True):
    clock.tick()
    img = sensor.snapshot()
    for blob in img.find_blobs(thresholds, pixels_threshold=100, area_threshold=100, merge=True):
        if blob.code() == 3: # r/g code
            img.draw_rectangle(blob.rect())
            img.draw_cross(blob.cx(), blob.cy())
            img.draw_string(blob.x() + 2, blob.y() + 2, "r/g")
        if blob.code() == 5: # r/b code
            img.draw_rectangle(blob.rect())
            img.draw_cross(blob.cx(), blob.cy())
```

```
            img.draw_string(blob.x() + 2, blob.y() + 2,
"r/b")
        if blob.code() == 6: # g/b code
            img.draw_rectangle(blob.rect())
            img.draw_cross(blob.cx(), blob.cy())
            img.draw_string(blob.x() + 2, blob.y() + 2,
"g/b")
        if blob.code() == 7: # r/g/b code
            img.draw_rectangle(blob.rect())
            img.draw_cross(blob.cx(), blob.cy())
            img.draw_string(blob.x() + 2, blob.y() + 2,
"r/g/b")
    print(clock.fps())
```

5.10.8、红外信标 RGB565 跟踪示例

ir_beacon_rgb565_tracking

此示例显示使用 OpenMV Cam 的 IR 信号 RGB565 跟踪。

```
import sensor, image, time

thresholds = (100, 100, 0, 0, 0, 0) # thresholds for br
ight white light from IR.

sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.VGA)
sensor.set_windowing((240, 240)) # 240x240 center pixel
s of VGA
sensor.skip_frames(30)
sensor.set_auto_gain(False) # must be turned off for co
lor tracking
sensor.set_auto_whitebal(False) # must be turned off fo
r color tracking
clock = time.clock()
```

只有比“pixel_threshold”更多的像素和比“area_threshold”更多的区域的斑点才能 通过下面的“find_blob”返回。如果更改 摄像机分辨率，请更改“pixels_threshold”和 “area_threshold” 。“merge = True”合并图像中的所有重叠的斑点。

```
while(True):
```

```
    clock.tick()
    img = sensor.snapshot()
    for blob in img.find_blobs([thresholds], pixels_thr
eshold=200, area_threshold=200, merge=True):
        ratio = blob.w() / blob.h()
        if (ratio >= 0.5) and (ratio <= 1.5): # filter
out non-squarish blobs
            img.draw_rectangle(blob.rect())
            img.draw_cross(blob.cx(), blob.cy())
    print(clock.fps())
```

5.10.9、红外信标灰度跟踪示例

ir_beacon_grayscale_tracking

该示例显示使用 OpenMV Cam 的 IR 信标灰度跟踪。

```
import sensor, image, time

thresholds = (255, 255) # thresholds for bright white l
ight from IR.

sensor.reset()
sensor.set_pixformat(sensor.GRAYSCALE)
sensor.set_framesize(sensor.VGA)
sensor.set_windowing((240, 240)) # 240x240 center pixel
s of VGA
sensor.skip_frames(30)
sensor.set_auto_gain(False) # must be turned off for co
lor tracking
sensor.set_auto_whitebal(False) # must be turned off fo
r color tracking
clock = time.clock()
```

只有比“pixel_threshold”更多的像素和更多的“find_blob”返回的斑点。如果更改 摄像机分辨率，请更改“pixels_threshold”和“area_threshold” 。“merge = True”合并图像中的所有重叠的斑点。

```
while(True):
    clock.tick()
    img = sensor.snapshot()
    for blob in img.find_blobs([thresholds], pixels_thr
eshold=200, area_threshold=200, merge=True):
```

```
        ratio = blob.w() / blob.h()
        if (ratio >= 0.5) and (ratio <= 1.5): # filter
out non-squarish blobs
            img.draw_rectangle(blob.rect())
            img.draw_cross(blob.cx(), blob.cy())
    print(clock.fps())
```

5.10.10、自动 RGB565 颜色跟踪示例

automatic_rgb565_color_tracking

本示例显示使用 OpenMV Cam 的单色自动 RGB565 颜色跟踪。

```
import sensor, image, time
print("Letting auto algorithms run. Don't put anything
in front of the camera!")

sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QVGA)
sensor.skip_frames(60)
sensor.set_auto_gain(False) # must be turned off for co
lor tracking
sensor.set_auto_whitebal(False) # must be turned off fo
r color tracking
clock = time.clock()
```

捕获图像中心的颜色阈值。

```
r = [(320//2)-(50//2), (240//2)-(50//2), 50, 50] # 50x5
0 center of QVGA.

print("Auto algorithms done. Hold the object you want t
o track in front of the camera in the box.")
print("MAKE SURE THE COLOR OF THE OBJECT YOU WANT TO TR
ACK IS FULLY ENCLOSED BY THE BOX!")
for i in range(60):
    img = sensor.snapshot()
    img.draw_rectangle(r)

print("Learning thresholds...")
threshold = [50, 50, 0, 0, 0, 0] # Middle L, A, B value
s.
for i in range(60):
```

```
    img = sensor.snapshot()
    hist = img.get_histogram(roi=r)
    lo = hist.get_percentile(0.01) # Get the CDF of the
 histogram at the 1% range (ADJUST AS NECESSARY)!
    hi = hist.get_percentile(0.99) # Get the CDF of the
 histogram at the 99% range (ADJUST AS NECESSARY)!
```

百分位数值平均值。

```
    threshold[0] = (threshold[0] + lo.l_value()) // 2
    threshold[1] = (threshold[1] + hi.l_value()) // 2
    threshold[2] = (threshold[2] + lo.a_value()) // 2
    threshold[3] = (threshold[3] + hi.a_value()) // 2
    threshold[4] = (threshold[4] + lo.b_value()) // 2
    threshold[5] = (threshold[5] + hi.b_value()) // 2
    for blob in img.find_blobs([threshold], pixels_thre
shold=100, area_threshold=100, merge=True, margin=10):
        img.draw_rectangle(blob.rect())
        img.draw_cross(blob.cx(), blob.cy())
        img.draw_rectangle(r)

print("Thresholds learned...")
print("Tracking colors...")

while(True):
    clock.tick()
    img = sensor.snapshot()
    for blob in img.find_blobs([threshold], pixels_thre
shold=100, area_threshold=100, merge=True, margin=10):
        img.draw_rectangle(blob.rect())
        img.draw_cross(blob.cx(), blob.cy())
    print(clock.fps())
```

5.10.11、自动灰度色彩跟踪示例

automatic_grayscale_color_tracking

此示例显示使用 OpenMV Cam 的单色自动灰度色彩跟踪。

```
import sensor, image, time
print("Letting auto algorithms run. Don't put anything
in front of the camera!")

sensor.reset()
```

```
sensor.set_pixformat(sensor.GRAYSCALE)
sensor.set_framesize(sensor.QVGA)
sensor.skip_frames(60)
sensor.set_auto_gain(False) # must be turned off for co
lor tracking
sensor.set_auto_whitebal(False) # must be turned off fo
r color tracking
clock = time.clock()
```

捕获图像中心的颜色阈值。

```
r = [(320//2)-(50//2), (240//2)-(50//2), 50, 50] # 50x5
0 center of QVGA.

print("Auto algorithms done. Hold the object you want t
o track in front of the camera in the box.")
print("MAKE SURE THE COLOR OF THE OBJECT YOU WANT TO TR
ACK IS FULLY ENCLOSED BY THE BOX!")
for i in range(60):
    img = sensor.snapshot()
    img.draw_rectangle(r)

print("Learning thresholds...")
threshold = [128, 128] # Middle grayscale values.
for i in range(60):
    img = sensor.snapshot()
    hist = img.get_histogram(roi=r)
    lo = hist.get_percentile(0.01) # Get the CDF of the
 histogram at the 1% range (ADJUST AS NECESSARY)!
    hi = hist.get_percentile(0.99) # Get the CDF of the
 histogram at the 99% range (ADJUST AS NECESSARY)!
```

百分位数值平均值。

```
    threshold[0] = (threshold[0] + lo.value()) // 2
    threshold[1] = (threshold[1] + hi.value()) // 2
    for blob in img.find_blobs([threshold], pixels_thre
shold=100, area_threshold=100, merge=True, margin=10):
        img.draw_rectangle(blob.rect())
        img.draw_cross(blob.cx(), blob.cy())
        img.draw_rectangle(r)

print("Thresholds learned...")
print("Tracking colors...")
```

```
while(True):
    clock.tick()
    img = sensor.snapshot()
    for blob in img.find_blobs([threshold], pixels_thre
shold=100, area_threshold=100, merge=True, margin=10):
        img.draw_rectangle(blob.rect())
        img.draw_cross(blob.cx(), blob.cy())
    print(clock.fps())
```

5.10.12、黑色灰度线以下示例

black_grayscale_line_following

在机器人之后做一条生产线需要付出很多努力。此示例脚本 显示如何在机器人后面的行中执行机器视觉部分。您 可以使用此脚本的输出来驱动差速驱动器机器人 遵循一行。该脚本只生成一个转弯值，告诉 您的机器人向左或向右。

要使此脚本正常工作，您应该以 45 度左右的角度将相机对准线。请确保只有线路在 相机的视野内。

```
import sensor, image, time, math
```

跟踪一条黑线。使用[（128，255）]来追踪白线。

```
GRAYSCALE_THRESHOLD = [(0, 64)]
```

每个 roi 是（x，y，w，h）。线检测算法将尝试找到 每个 roi 中最大斑点的质心。 然后将重心的 x 位置用不同的权重进行平均，其中最重的权重被分配 给靠近图像底部的 roi，而不等于下一个 roi 等等。

```
ROIS = [ # [ROI, weight]
        (0, 100, 160, 20, 0.7), # You'll need to tweak
the weights for you app
        (0, 050, 160, 20, 0.3), # depending on how your
 robot is setup.
        (0, 000, 160, 20, 0.1)
       ]
```

计算权重除数（我们计算这个，所以你不必使权重添加到 1）。

```
weight_sum = 0
```

```
for r in ROIS: weight_sum += r[4] # r[4] is the roi wei
ght.
```

相机设置...

```
sensor.reset() # Initialize the camera sensor.
sensor.set_pixformat(sensor.GRAYSCALE) # use grayscale.
sensor.set_framesize(sensor.QQVGA) # use QQVGA for spee
d.
sensor.skip_frames(30) # Let new settings take affect.
sensor.set_auto_gain(False) # must be turned off for co
lor tracking
sensor.set_auto_whitebal(False) # must be turned off fo
r color tracking
clock = time.clock() # Tracks FPS.

while(True):
    clock.tick() # Track elapsed milliseconds between s
napshots().
    img = sensor.snapshot() # Take a picture and return
 the image.

    centroid_sum = 0
    for r in ROIS:
        blobs = img.find_blobs(GRAYSCALE_THRESHOLD, roi
=r[0:4], merge=True) # r[0:4] is roi tuple.
        if blobs:
```

找到最多像素的斑点的索引。

```
            most_pixels = 0
            largest_blob = 0
            for i in range(len(blobs)):
                if blobs[i].pixels() > most_pixels:
                    most_pixels = blobs[i].pixels()
                    largest_blob = i
```

在斑点周围画一个直角。

```
            img.draw_rectangle(blobs[largest_blob].rect
())
            img.draw_cross(blobs[largest_blob].cx(),
                           blobs[largest_blob].cy())
```

```
            centroid_sum += blobs[largest_blob].cx() *
r[4] # r[4] is the roi weight.

    center_pos = (centroid_sum / weight_sum) # Determin
e center of line.
```

将 center_pos 转换为偏转角。我们正在使用非线性 操作，使得响应越来越靠近我们的线 。非线性的操作很好地用于输出这样的算法 ，引起响应“触发”。

```
    deflection_angle = 0
```

80 是从 X res 的一半，60 是从 Y res 的一半。下面的 等式只是计算三角形的角度，其中三角形的 相对侧是中心位置与中心的偏差 ，相邻边是 Y res 的一半。这 将角度输出限制在-45 到 45 度左右（这不是-45 和 45）。

```
    deflection_angle = -math.atan((center_pos-80)/60)
```

将角度以弧度转换为度数。

```
    deflection_angle = math.degrees(deflection_angle)
```

现在，您有一个角度告诉您多少转动机器人，其中 将最靠近机器人的线的一部分和 远离机器人的线的部分进行更好的预测。

```
    print("Turn Angle: %f" % deflection_angle)

    print(clock.fps()) # Note: Your OpenMV Cam runs abo
ut half as fast while
```

连接到您的计算机。一旦断开连接，FPS 应该增加。

5.11、11-显示屏

5.11.1、显示屏

lcd

注意：要运行此示例，您将需要一个 LCD 屏幕用于您的 OpenMV 摄像头。 LCD 屏幕允许您随时随地查看 OpenMV 的帧缓冲区。

```
import sensor, image, lcd

```

```
sensor.reset() # Initialize the camera sensor.
sensor.set_pixformat(sensor.RGB565) # or sensor.GRAYSCA
LE
sensor.set_framesize(sensor.QQVGA2) # Special 128x160 f
ramesize for LCD Shield.
lcd.init() # Initialize the lcd screen.

while(True):
    lcd.display(sensor.snapshot()) # Take a picture and
 display the image.
```

5.12、15-测试

5.12.1、FPS 测试脚本

fps

```
import sensor, image, time

sensor.reset() # Initialize the camera sensor.
sensor.set_framesize(sensor.QQVGA) # or sensor.QQVGA (o
r others)
sensor.set_pixformat(sensor.RGB565) # or sensor.GRAYSCA
LE
sensor.set_colorbar(True)   # Enable colorbars output

clock = time.clock() # Tracks FPS.
for i in range(0, 600):
    clock.tick() # Track elapsed milliseconds between s
napshots().
    sensor.snapshot() # Capture snapshot.

print("FPS:", clock.fps())
```

5.12.2、自检

selftest

此示例显示您的 OpenMV 在出厂之前进行测试。每个 OpenMV Cam 都应该通过这个测试。

```
import sensor, time, pyb

def test_int_adc():
```

```
    adc  = pyb.ADCAll(12)
    # Test VBAT
    vbat = adc.read_core_vbat()
    vbat_diff = abs(vbat-3.3)
    if (vbat_diff > 0.1):
        raise Exception('INTERNAL ADC TEST FAILED VBAT
=%fv'%vbat)
```

测试 VREF

```
    vref = adc.read_core_vref()
    vref_diff = abs(vref-1.2)
    if (vref_diff > 0.1):
        raise Exception('INTERNAL ADC TEST FAILED VREF
=%fv'%vref)
    adc = None
    print('INTERNAL ADC TEST PASSED...')

def test_color_bars():
    sensor.reset()
```

设置传感器设置

```
    sensor.set_brightness(0)
    sensor.set_saturation(3)
    sensor.set_gainceiling(8)
    sensor.set_contrast(2)
```

设置传感器像素格式

```
    sensor.set_framesize(sensor.QVGA)
    sensor.set_pixformat(sensor.RGB565)
```

启用色标测试模式

```
    sensor.set_colorbar(True)
```

跳过几帧以使传感器稳定下来

```
    for i in range(0, 100):
        image = sensor.snapshot()
```

色条阈值

```
    t = [lambda r, g, b: r < 70  and g < 70  and b < 7
0,   # Black
         lambda r, g, b: r < 70  and g < 70  and b > 20
0,  # Blue
         lambda r, g, b: r > 200 and g < 70  and b < 7
0,   # Red
         lambda r, g, b: r > 200 and g < 70  and b > 20
0,  # Purple
         lambda r, g, b: r < 70  and g > 200 and b < 7
0,   # Green
         lambda r, g, b: r < 70  and g > 200 and b > 20
0,  # Aqua
         lambda r, g, b: r > 200 and g > 200 and b < 7
0,   # Yellow
         lambda r, g, b: r > 200 and g > 200 and b > 20
0]  # White
```

OV7725 的彩条反转

```
    if (sensor.get_id() == sensor.OV7725):
        t = t[::-1]
```

320x240 的图像与 8 个彩条，每个大约 40 像素。 我们从帧缓冲区的中心开始，并 从每个颜色条的中心平均 10 个样本像素的值。

```
    for i in range(0, 8):
        avg = (0, 0, 0)
        idx = 40*i+20 #center of colorbars
        for off in range(0, 10): #avg 10 pixels
            rgb = image.get_pixel(idx+off, 120)
            avg = tuple(map(sum, zip(avg, rgb)))

        if not t[i](avg[0]/10, avg[1]/10, avg[2]/10):
            raise Exception('COLOR BARS TEST FAILED.'
            'BAR#(%d): RGB(%d,%d,%d)'%(i+1, avg[0]/10,
avg[1]/10, avg[2]/10))

    print('COLOR BARS TEST PASSED...')

if __name__ == '__main__':
    print('')
    test_int_adc()
    test_color_bars()
```

5.12.3、色条测试示例

colorbar

这个例子是在出厂前被每个 OpenMV Cam 运行的颜色条测试。OMV 传感器可以输出一个彩条图像，您可以通过阈值来检查相机总线是否正确连接。

```
import sensor, time

sensor.reset()
```

设置传感器设置

```
sensor.set_brightness(0)
sensor.set_saturation(3)
sensor.set_gainceiling(8)
sensor.set_contrast(2)
```

设置传感器像素格式

```
sensor.set_framesize(sensor.QVGA)
sensor.set_pixformat(sensor.RGB565)
```

启用色标测试模式

```
sensor.set_colorbar(True)

# Skip a few frames to allow the sensor settle down
for i in range(0, 30):
    image = sensor.snapshot()
```

色条阈值

```
t = [lambda r, g, b: r < 70  and g < 70  and b < 70,
# Black
     lambda r, g, b: r < 70  and g < 70  and b > 200,
# Blue
     lambda r, g, b: r > 200 and g < 70  and b < 70,
# Red
     lambda r, g, b: r > 200 and g < 70  and b > 200,
# Purple
     lambda r, g, b: r < 70  and g > 200 and b < 70,
# Green
```

```
    lambda r, g, b: r < 70  and g > 200 and b > 200,
# Aqua
    lambda r, g, b: r > 200 and g > 200 and b < 70,
# Yellow
    lambda r, g, b: r > 200 and g > 200 and b > 200]
# White
```

OV7725 的彩条反转

```
if (sensor.get_id() == sensor.OV7725):
    t = t[::-1]
```

320x240 的图像与 8 个彩条，每个大约 40 像素。 我们从帧缓冲区的中心开始，并 从每个颜色条的中心平均 10 个样本像素的值。

```
for i in range(0, 8):
    avg = (0, 0, 0)
    idx = 40*i+20 # center of colorbars
    for off in range(0, 10): # avg 10 pixels
        rgb = image.get_pixel(idx+off, 120)
        avg = tuple(map(sum, zip(avg, rgb)))

    if not t[i](avg[0]/10, avg[1]/10, avg[2]/10):
        raise Exception("COLOR BARS TEST FAILED. "
        "BAR#(%d): RGB(%d,%d,%d)"%(i+1, avg[0]/10, avg
[1]/10, avg[2]/10))

print("COLOR BARS TEST PASSED...")
```

5.13、16-二维码

5.13.1、AprilTags 示例

AprilTags Example

此示例显示 OpenMV Cam 检测 OpenMV Cam M7 上的四月标签的功能。M4 版本无法检测四月标签。

```
import sensor, image, time, math

sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QQVGA) # we run out of memo
ry if the resolution is much bigger...
```

```
sensor.skip_frames(30)
sensor.set_auto_gain(False)  # must turn this off to pr
event image washout...
sensor.set_auto_whitebal(False)  # must turn this off t
o prevent image washout...
clock = time.clock()
```

注意！与 find_qrcodes 不同，find_apriltags 方法不需要对图像进行镜头校正。

apriltag 代码最多支持 6 个标签系列，可以同时处理。 返回的标签对象将在标签系列内具有标签系列和标识。

```
tag_families = 0
tag_families |= image.TAG16H5 # comment out to disable
this family
tag_families |= image.TAG25H7 # comment out to disable
this family
tag_families |= image.TAG25H9 # comment out to disable
this family
tag_families |= image.TAG36H10 # comment out to disable
 this family
tag_families |= image.TAG36H11 # comment out to disable
 this family (default family)
tag_families |= image.ARTOOLKIT # comment out to disabl
e this family
```

标签家族有什么区别？那么，例如，TAG16H5 系列实际上是 一个 4x4 的方形标签。因此，这意味着它可以在比标签 为 6x6 的标签更长的距离处看到。然而，较低的 H 值（H5 对 H11）意味着 4x4 标签的误判率比 6x6 标签高许多，甚至更高。所以，除非你 有理由使用其他标签，否则只需使用默认系列的 TAG36H11。

```
def family_name(tag):
    if(tag.family() == image.TAG16H5):
        return "TAG16H5"
    if(tag.family() == image.TAG25H7):
        return "TAG25H7"
    if(tag.family() == image.TAG25H9):
        return "TAG25H9"
    if(tag.family() == image.TAG36H10):
        return "TAG36H10"
    if(tag.family() == image.TAG36H11):
        return "TAG36H11"
```

```
    if(tag.family() == image.ARTOOLKIT):
        return "ARTOOLKIT"

while(True):
    clock.tick()
    img = sensor.snapshot()
    for tag in img.find_apriltags(families=tag_familie
s): # defaults to TAG36H11 without "families".
        img.draw_rectangle(tag.rect(), color = (255, 0,
 0))
        img.draw_cross(tag.cx(), tag.cy(), color = (0,
255, 0))
        print_args = (family_name(tag), tag.id(), (180
* tag.rotation()) / math.pi)
        print("Tag Family %s, Tag ID %d, rotation %f (d
egrees)" % print_args)
    print(clock.fps())
```

5.13.2、AprilTags 示例

find_apriltags_3d_pose

此示例显示 OpenMV Cam 检测 OpenMV Cam M7 上的四月标签的功能。M4 版本无法检测四月标签。

```
import sensor, image, time, math

sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QQVGA) # we run out of memo
ry if the resolution is much bigger...
sensor.skip_frames(30)
sensor.set_auto_gain(False)  # must turn this off to pr
event image washout...
sensor.set_auto_whitebal(False)  # must turn this off t
o prevent image washout...
clock = time.clock()
```

注意！与 find_qrcodes 不同，find_apriltags 方法不需要对图像进行镜头校正。

标签家族有什么区别？那么，例如，TAG16H5 系列实际上是 一个 4x4 的方形标签。因此，这意味着它可以在比标签 为 6x6 的标签更长的距离处看到。然而，较低的 H 值（H5 对 H11）意味着 4x4 标签的误

判率比 6x6 标签高许多，甚至更高。所以，除非你 有理由使用其他标签，否则只需使用默认系列的 TAG36H11。

AprilTags 库输出标签的姿态信息。这是 x / y / z 平移和 x / y / z 旋转。x / y / z 旋转为弧度，可以转换为度数。对于 翻译，单位是无量纲的，您必须应用转换函数。

f_x 是相机的 x 焦距。它应该等于透镜焦距，以 mm 除以 x 传感器尺寸，以 mm 乘以传感器长度的像素数。 以下数值适用于带有 2.8 mm 镜头的 OV7725 相机。

f_y 是相机的 y 焦距。它应该等于以 mm 为单位的透镜焦距 除以 y 传感器尺寸，以 mm 乘以传感器长度的像素数。 以下数值适用于带有 2.8 mm 镜头的 OV7725 相机。

c_x 是以像素为单位的图像 x 中心位置。 c_y 是以像素为单位的图像 y 中心位置。

```
f_x = (2.8 / 3.984) * 656 # find_apriltags defaults to
this if not set
f_y = (2.8 / 2.952) * 488 # find_apriltags defaults to
this if not set
c_x = 160 * 0.5 # find_apriltags defaults to this if no
t set (the image.w * 0.5)
c_y = 120 * 0.5 # find_apriltags defaults to this if no
t set (the image.h * 0.5)

def degrees(radians):
    return (180 * radians) / math.pi

while(True):
    clock.tick()
    img = sensor.snapshot()
    for tag in img.find_apriltags(fx=f_x, fy=f_y, cx=c_
x, cy=c_y): # defaults to TAG36H11
        img.draw_rectangle(tag.rect(), color = (255, 0,
 0))
        img.draw_cross(tag.cx(), tag.cy(), color = (0,
255, 0))
        print_args = (tag.x_translation(), tag.y_transl
ation(), tag.z_translation(), \
            degrees(tag.x_rotation()), degrees(tag.y_ro
tation()), degrees(tag.z_rotation()))
        # Translation units are unknown. Rotation units
 are in degrees.
```

```
        print("Tx: %f, Ty %f, Tz %f, Rx %f, Ry %f, Rz %
f" % print_args)
    print(clock.fps())
```

5.13.3、AprilTags 示例

find_apriltags_w_lens_zoom

此示例显示 OpenMV Cam 检测 OpenMV Cam M7 上的四月标签的功能。M4 版本无法检测四月标签。

```
import sensor, image, time, math

sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.VGA) # we run out of memory
 if the resolution is much bigger...
sensor.set_windowing((160, 120)) # Look at center 160x1
20 pixels of the VGA resolution.
sensor.skip_frames(30)
sensor.set_auto_gain(False)  # must turn this off to pr
event image washout...
sensor.set_auto_whitebal(False)  # must turn this off t
o prevent image washout...
clock = time.clock()
```

注意！与 find_qrcodes 不同，find_apriltags 方法不需要对图像进行镜头校正。

标签家族有什么区别？那么，例如，TAG16H5 系列实际上是 一个 4x4 的方形标签。因此，这意味着它可以在比标签 为 6x6 的标签更长的距离处看到。然而，较低的 H 值（H5 对 H11）意味着 4x4 标签的误判率比 6x6 标签高许多，甚至更高。所以，除非你 有理由使用其他标签，否则只需使用默认系列的 TAG36H11。

```
while(True):
    clock.tick()
    img = sensor.snapshot()
    for tag in img.find_apriltags(): # defaults to TAG3
6H11
        img.draw_rectangle(tag.rect(), color = (255, 0,
 0))
        img.draw_cross(tag.cx(), tag.cy(), color = (0,
255, 0))
```

```
        print_args = (tag.id(), (180 * tag.rotation())
/ math.pi)
        print("Tag Family TAG36H11, Tag ID %d, rotation
 %f (degrees)" % print_args)
    print(clock.fps())
```

5.13.4、QRCode 示例

qrcodes_with_lens_corr

此示例显示了 OpenMV Cam 检测 QR 码的能力 使用镜头校正（有关更高的性能，请参阅 qrcodes_with_lens_corr.py 脚本）。

```
import sensor, image, time

sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QQVGA)#可以是 M7 上的 QVGA
 ...
sensor.skip_frames(30)
sensor.set_auto_gain(False) #必须关闭此选项以防止图像
被清除...
clock = time.clock()

while(True):
    clock.tick()
    img = sensor.snapshot()
    img.lens_corr(1.8) #强度为 1.8 对于 2.8mm 镜头合适。
    for code in img.find_qrcodes():
        print(code)
    print(clock.fps())
```

5.13.5、QRCode 示例

qrcodes_with_lens_zoom

此示例显示了 OpenMV Cam 在 无需镜头校正的情况下检测 QR 码的功率。

```
import sensor, image, time

sensor.reset()
sensor.set_pixformat(sensor.GRAYSCALE)
sensor.set_framesize(sensor.VGA)
```

```
sensor.set_windowing((240, 240)) # look at center 240x2
40 pixels of the VGA resolution.
sensor.skip_frames(30)
sensor.set_auto_gain(False) # must turn this off to pre
vent image washout...
clock = time.clock()

while(True):
    clock.tick()
    img = sensor.snapshot()
    for code in img.find_qrcodes():
        print(code)
    print(clock.fps())
```