



- 首页
- 所有文章
- 观点与动态
- 基础知识
- 系列教程
- 实践项目
- 工具与框架
- 工具资源
- Python小组

- 导航条 - ▾

伯乐在线 > Python - 伯乐在线 > 所有文章 > 实践项目 > 用 Python 和 OpenCV 检测和跟踪运动对象

用 Python 和 OpenCV 检测和跟踪运动对象

2015/06/13 · [实践项目](#) · [11 评论](#) · [OpenCV](#), [Python](#)

分享到：
本文由 [伯乐在线](#) - [艾凌风](#) 翻译，[黄利民](#) 校稿。未经许可，禁止转载！
英文出处：[pyimagesearch](#)。欢迎加入[翻译组](#)。



对于一个男人来讲，这些话永远都不该说。但是当我关上冰箱门的时候，我愤怒地叹息，感到厌恶，自言自语地说了这些。

你看，我花了12个小时写了这篇将要发表的文章《[PyImageSearch Gurus course](#)》。我的脑子都糊掉了，像个半熟的摊鸡蛋一样，几乎要从耳朵里流出来了。当我深夜决定结束工作的时候，我只想放松一下，看看我最爱的电影——《侏罗纪公园》。同时喝着来自 Smuttynose 的最好的 IPA 冰啤，Smuttynose 是近来我非常喜欢的一家酒厂。

但是，昨天晚上来串门的该死的 James 喝掉了我最后一罐啤酒。

好吧，据称。

我并不能证明任何我的猜测。实际上，我并没有亲眼看到他喝我的啤酒，因为我埋头于笔记本电脑中，手指在键盘上跳动，兴奋地敲击出教程和文章。但是我感觉他就是嫌疑犯。他是我唯一会喝 IPA 的（前）朋友。

所以我做了一件任何男人都会做的事。

我在橱柜顶上安装了一个树莓派，来探测看他是不是打算再次偷啤酒。



过分了？

也许吧。

但是，我很看重我的啤酒。而且如果 James 再次尝试偷我的啤酒的话，我会逮他个正着。

做一个用于家庭监控的运动检测和追踪系统，分两部分，本文是第一篇。

本文接下来的部分，将会详细介绍如何使用计算机视觉技术来建立一个用于家庭监控的基础的运动检测和追踪系统。本例对预先录制的视频和网络摄像头的实时数据流都可以工作；然而，我们将会在我们的笔记本/桌面电脑上进行开发。

在本系列的第二部分中，我会向你展示如何升级代码，使其可以在树莓派和camera board上工作，以及如何扩展家庭监控系统，来捕捉任何检测到的运动，并且上传到你的个人Dropbox中。

也许到了最后，我们可以把 James 抓个正着。

一点关于背景移除的内容

背景移除是很多计算机视觉应用的关键内容。我们通过它来计算经过收费站的汽车个数。我们通过它来计算进进出出一间商店的人的个数。

同时我们使用它来进行运动检测。

在本文开始写代码之前，让我告诉你，OpenCV 里有很多很多方法来进行运动检测、追踪和分析。有一些非常简单，而另外一些非常复杂。两个初级的方法是某种形式的基于混合高斯模型的前景和背景分割：

1. KaewTraKulPong 等人发表的《[An improved adaptive background mixture model for real-time tracking with shadow detection](#)》。这个方法可以通过 `cv2.BackgroundSubtractorMOG` 来使用。
2. Zivkovic 提出的《[Improved adaptive Gaussian mixture model for background subtraction](#)》和《[Efficient Adaptive Density Estimation per Image Pixel for the Task of Background Subtraction](#)》。可以通过 `cv2.BackgroundSubtractorMOG2` 来使用。

在新版本的 OpenCV 中，我们有基于贝叶斯（概率）的前景和背景分割，是 Godbehere 等人在 2012 年的文章中实现的，《[Visual Tracking of Human Visitors under Variable-Lighting Conditions for a Responsive Audio Art Installation](#)》，我们可以在

`cv2.createBackgroundSubtractorGMG` 中找到它的实现（然而我们需要等 OpenCV 3 的到来，才能使用它的全部功能。）

所有这些方法都涉及到从前景中分离背景（它们甚至提供相应的机制来让我们辨别实际运动和阴影及关照的细微改变）！

为什么这一点特别重要？为什么我们这么在意哪个像素属于前景哪个像素属于背景？

在运动检测中，我们会做出如下的假设：

我们视频流中的背景在连续的视频帧内，多数时候应该是静止不变的，因此如果我们可以建立背景模型，我们的就可以监视到显著的变化。如果发生了显著的变化，我们就可以检测到它——通常这些变化和我们视频中的运动有关。

显然在现实世界中，我们这个假设比较容易失效。因为阴影、反色、光照条件以及环境中可能发生的其他变化，我们的背景可能会看上去变得非常不同，这会让我们的算法失效。所以为什么最成功的背景移除/前景检测系统需要固定安装的相机以及控制光照条件。

后，把该系统部署在树莓派上，因此我们最好可以坚持使用简单的方法。我们待在不断的文章中指出到这些强大的方法上，但是目前我们将保持简单和高效。

用 Python 和 OpenCV 进行基础的运动检测和追踪

好了，准备好帮助我开发一个家用监视系统来抓住那个偷啤酒的混蛋了么？打开编辑器，新建一个文件，命名为 **motion_detector.py**，然后让我们开始写代码吧。

```
Python
1 # 导入必要的软件包
2 import argparse
3 import datetime
4 import imutils
5 import time
6 import cv2
7
8 # 创建参数解析器并解析参数
9 ap = argparse.ArgumentParser()
10 ap.add_argument("-v", "--video", help="path to the video file")
11 ap.add_argument("-a", "--min-area", type=int, default=500, help="minimum area size")
12 args = vars(ap.parse_args())
13
14 # 如果video参数为None，那么我们从摄像头读取数据
15 if args.get("video", None) is None:
16     camera = cv2.VideoCapture(0)
17     time.sleep(0.25)
18
19 # 否则我们读取一个视频文件
20 else:
21     camera = cv2.VideoCapture(args["video"])
22
23 # 初始化视频流的第一帧
24 firstFrame = None
```

2-6行导入了我们必要的软件包。这些看上去都很熟悉，除了 `imutils` 这个包，它提供了一组由我编写的非常方便的函数，来让我们更简单的进行图像处理。如果你还没有安装 [imutils](#) 到你的系统，你可以通过 `pip` 来安装：`pip install imutils`

下一步，我们在**9-12行**解析了命令行参数。我们定义了两个选项。第一个，`--video`，是可选的。它会指定一个路径，指向一个预先录制好的视频文件，我们可以检测该视频中的运动。如果你不提供视频的路径，那么 `OpenCV` 会从你的摄像头中来检测运动。

我们同时还定义了 `--min-area`，它表示一个图像区域被看做实际运动的最小尺寸（以像素为单位）。正如我接下来要讲的那样，我们会发现图像中比较小的区域变化会比较显著，可能是因为噪点或是光线的变化。在实际中，这些小区域并不是实际的运动——所以我们定义一个最小的尺寸来对付和过滤掉这些假阳性（`false-positives`）结果。

15-21行获取一个我们摄像机对象的引用。在这个例子中，没有提供视频路径（**15-17行**），我们会取得一个摄像头的引用。如果提供了一个视频文件路径，那么我们会在**20-21行**建立一个指向它的指针。

最后，我们以一个变量来结束这段代码，这个变量是 `firstFrame`。能猜到 `firstFrame` 是什么吗？

假设：视频的第一帧不会包含运动，而仅仅是背景——因此我们可以使用第一帧来建立背景模型。显然我们此处建立的假设有些太大了。但是再说一次，我们的目标是要在树莓派上运行这个系统，

我们可以轻松的检测到运动并追踪他们。

Python

```
1 # 遍历视频的每一帧
2 while True:
3     # 获取当前帧并初始化occupied/unoccupied文本
4     (grabbed, frame) = camera.read()
5     text = "Unoccupied"
6
7     # 如果不能抓取到一帧，说明我们到了视频的结尾
8     if not grabbed:
9         break
10
11     # 调整该帧的大小，转换为灰阶图像并且对其进行高斯模糊
12     frame = imutils.resize(frame, width=500)
13     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
14     gray = cv2.GaussianBlur(gray, (21, 21), 0)
15
16     # 如果第一帧是None，对其进行初始化
17     if firstFrame is None:
18         firstFrame = gray
19     continue
```

现在我们已经获取了视频文件/摄像头数据流的引用，我们可以在第一行（原文第27行）开始遍历每一帧了。

调用`camera.read()`为我们返回一个2元组。元组的第一个值是`grabbed`，表明是否成功从缓冲中读取了`frame`。元组的第二个值就是`frame`它本身。

我们同时还定义了一个叫做 `text` 的字符串，并对其进行初始化来表明我们正在监控的这个房间“没有被占领”（`Unoccupied`）。如果这个房间确实有活动，我们可以更新这个字符串。

在这个例子中，如果没有成功从视频文件中读取一帧，我们会在10-11行（原文35-36行）跳出循环。

我们可以开始处理帧数据并准备进行运动分析（**15-17行**）。我们首先会调整它的大小到500像素宽——没有必要去直接处理视频流中的大尺寸，原始图像。我们同样会把图片转换为灰阶图像，因为彩色数据对我们的运动检测算法没有影响。最后，我们会使用高斯模糊来平滑我们的图像。

认识到即使是相邻帧，也不是完全相同的这一点很重要！

由于数码相机传感器的微小变化，没有100%相同的两帧数据——一些像素肯定会有不同的强度值。也就是说，我们需要，并应用高斯平滑对一个11X11的区域的像素强度进行平均。这能帮我们滤除可能使我们运动检测算法失效的高频噪音。

正如我在上面提到的，我们需要通过某种方式对我们的图像进行背景建模。再一次的，我们会假设视频的第一帧不包含任何运动，它是一个很好的例子，表明我们的背景是如何的。如果`firstFrame`没有初始化，我们会把它保存然后继续处理视频的下一帧。（**20-22行**）

这里有一个关于示例视频第一帧的例子：

上面这一帧满足我们的假设，视频的第一帧仅仅是一个静止的背景——没有运动。

有了这个静止的背景图片，我们已经准备好实时运动检测和追踪了：

Python

```

3     thresh = cv2.threshold(frameDelta, 25, 255, cv2.THRESH_BINARY)[1]
4
5     # 扩展阈值图像填充空洞，然后找到阈值图像上的轮廓
6     thresh = cv2.dilate(thresh, None, iterations=2)
7     (cnts, _) = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
8                                   cv2.CHAIN_APPROX_SIMPLE)
9
10    # 遍历轮廓
11    for c in cnts:
12        # if the contour is too small, ignore it
13        if cv2.contourArea(c) < args["min_area"]:
14            continue
15
16        # compute the bounding box for the contour, draw it on the frame,
17        # and update the text
18        # 计算轮廓的边界框，在当前帧中画出该框
19        (x, y, w, h) = cv2.boundingRect(c)
20        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
21        text = "Occupied"

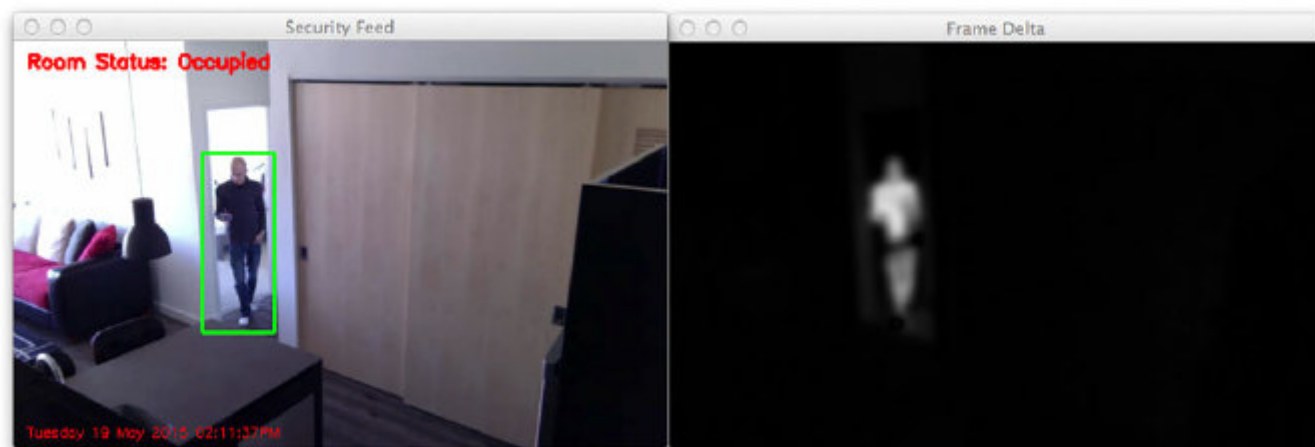
```

现在我们已经从`firstFrame`变量对背景进行了建模，我们可以利用它来计算起始帧和视频流数据中后续新帧之间的不同。

计算两帧的不同是一个简单的减法，我们使用两方相应的像素强度差的绝对值。（第二行）

`delta = |background_model - current_frame|`

两帧差值图例如下：



注意到图片的背景是如何变为黑色的。然而，包含运动的区域（比如包含我自己走过房间动作的区域）会更亮一些。这以为这两帧差值大的地方是图片中发生移动的区域。

我们随后在**第3行**对`frameDelta`进行阈值化来显示图片中像素强度值有显著变化的区域。如果差值小于25，我丢弃该像素将其设置为黑色（例如，背景）。如果差值大于25，我们将其设定为白色（例如，前景）。阈值化的差值图片如下：



再一次，注意到图片的背景是黑色的，而前景（运动发生的位置）是白色的。有了这个阈值化的图片，只要简单的进行实施轮廓检测来找到白色区域的外轮廓线（**第7行**）

我们在第14行开始对轮廓线进行遍历，在15行滤掉小的，不相关的轮廓。如果轮廓面积比我们提供的`--min-area`值大，我们会在前景和移动区域画边框线。（**23-25行**）。我们同样会更新`text`状态字符串来表示这个房间”被占领“（`Occupied`）了

Python

```

1 # draw the text and timestamp on the frame
2 # 在当前帧上写文字以及时间戳
3 cv2.putText(frame, "Room Status: {}".format(text), (10, 20),
4             cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)
5 cv2.putText(frame, datetime.datetime.now().strftime("%A %d %B %Y %I:%M:%S%p"),
6             (10, frame.shape[0] - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.35, (0, 0, 255), 1)
7
8 显示当前帧并记录用户是否按下按键
9 cv2.imshow("Security Feed", frame)
10 cv2.imshow("Thresh", thresh)
11 cv2.imshow("Frame Delta", frameDelta)
12 key = cv2.waitKey(1) & 0xFF
13
14 # 如果q键被按下，跳出循环
15 if key == ord("q"):
16     break
17
18 # 清理摄像机资源并关闭打开的窗口
19 camera.release()
20 cv2.destroyAllWindows()

```

11-13行显示了我的工作成果，运行我们可以在视频中看到是否检测到了运动，使用帧差值和阈值图像我们可以调试我们的脚本。

注意：如果你下载了本文的源代码并打算应用到你自己的视频文件上，你可能需要改变`cv2.threshold`的值和`--min-area`参数来获得你所在光照环境下的最佳效果。

最后，**22行和23行**清理并释放了视频流的指针。

结果

显然，我要确定我们的运动监测系统可以在James那个偷酒贼再次造访的之前能够正常工作——我们将在本系列第二篇文章中谈到他。为了测试我们使用Python和OpenCV搭建的运动监测系统，我录制了两个视频文件。

example_02.mp4 使用安装在橱柜上的摄像头录制的。它监控厨房和客厅，当有人从其中走动的时候完成检测。

让我们给我们简单的探测器一次尝试的机会，打开终端并执行下面指令：

```
Python
1 | $ python motion_detector.py --video videos/example_01.mp4
```

下图是一个 gif 图，显示来自探测器的一些静止帧数据。

注意到在门被打开前没有进行运动检测——然后我们可以检测到我自己从门中走过。你可以在这里看到全部视频：

<http://www.youtube.com/embed/fi4LORwk8Fc?feature=oembed>

现在，我安装在用于监视厨房和客厅的摄像机表现如何呢？然我们一探究竟。输入下面命令：

```
Python
1 | $ python motion_detector.py --video videos/example_02.mp4
```

来自第二个视频文件的结果样本如下：



同样，这里是我们的运动检测结果的完整视频：

<http://www.youtube.com/embed/36j238XtcIE?feature=oembed>

门和离开房间。

然而，现实来讲，结果还远远谈不上完美。尽管只有一个人在屋内走动，我们却得到了多个外框——这和理想状态相差甚远。而且我可以看到，微小的光线变化，比如阴影和墙面反射，都触发了假阳性的运动检测结果。

为了解决这些问题，我们依靠OpenCV中更加强大的背景移除方法，这些方法对阴影和少量的反射进行了处理。（我将在未来的文章中谈到这些更为先进的背景移除/前景检测方法）

但是于此同时，请考虑一下我们的最终目标

这个系统，尽管是在我们的笔记本/台式机系统上开发的，却是为了要部署在树莓派上，树莓派的计算资源非常有限。因此，我们需要让我们的运动检测方法保持简单和快速。我们的运动检测系统并不完美，很不幸这是一个不利的方面，但是对于我们特定的项目，它仍然能够很好的完成工作。

最后，如果你想要利用你的摄像头的原始视频流来进行运动检测，空着`--video`选项即可。

Python

```
1 | $ python motion_detector.py
```

小结

通过本文，我们已经认识到我的朋友James是一个偷酒贼。真是个混蛋啊！

为了能抓他个人赃并获，我们决定使用Python和OpenCV建立一个运动检测和追踪系统。这个系统可以获取视频流并分析它们获取运动。考虑到我们所使用的方法，能够得到可以接受的监测结果。

最终目标是要把本系统部署在树莓派上，因此我们没有依赖OpenCV中一些比较先进的背景移除方法。相反，我们依赖一个简单，但合理高效的假设——视频的第一帧仅仅包含我们想要建模的背景，而不包括其他任何东西。

在这个假设下，我们可以实施背景移除，检测图片中的运动，在检测到运动的区域画出轮廓框。

在这个关于运动检测系列文章的**第二部分**，我们会**更新代码使其在树莓派上运行**。

我们同样会**集成Dropbox API**，允许我们监控家用监控系统并且当我们的系统检测到运动时，获取实时更新数据。

敬请期待！

打赏支持我翻译更多好文章，谢谢！

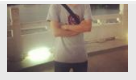
¥ [打赏译者](#)

👍 5 赞

🔖 21 收藏

💬 [11 评论](#)

关于作者：艾凌风



发，如长时间没收到请邮件催我

[个人主页](#) · [我的文章](#) · [95](#) · [🔗](#) [📧](#) [🔔](#)

相关文章

- [Python 项目可以有多大](#)
- [Python 数据处理库 pandas 进阶教程](#)
- [Python 数据处理库 pandas 入门教程](#)
- [Python 绘图库 Matplotlib 入门教程](#)
- [15 分钟用 ML 破解一个验证码系统](#) · [🔗 12](#)

可能感兴趣的话题

- [各位觉得大公司的规定适合小公司吗？](#) · [🔗 2](#)
- [Vuescroll - 一个基于Vue的虚拟滚动条](#)
- [现在报个班可以帮助自己吗？](#) · [🔗 15](#)
- [国外的程序员可以工作到退休而国内的为什么这么短命（思维认知）](#) · [🔗 4](#)
- [Java非要用繁杂的框架吗？JSP+serverlet有什么优缺点](#) · [🔗 9](#)
- [自然语言处理（NLP）如何入门？](#)

登录后评论

新用户注册

直接登录



最新评论

[默默](#)[2015/08/26](#)

文件的代码：`key = cv2.waitKey(1) & 0xFF`
提示有错误：`amp` 没有定义，应该怎么修改啊

[👍 赞](#) [回复](#) [↩](#)[Latifa](#) ([🎓 1](#) · [👤](#))
宁夏大学研究生[2016/11/17](#)

Python

```
1 amp = 0xFF
2 key = cv2.waitKey(1) & amp
```

[👍 赞](#) [回复](#) [↩](#)[wizarddhist](#)[2017/04/02](#)

真诚地问一下，这个代码用Windows平台怎么显示出最后的视频？

我运行的时候什么也不显示

真诚求帮助，谢谢了！

👍 赞 回复 ↩



Hy

2016/03/02

怎么查看运行后的结果呢

👍 赞 回复 ↩



zlytob

2016/10/02

key = cv2.waitKey(1) & 0xFF
NameError: name 'amp' is not defined

这个怎么解决呢？

👍 赞 回复 ↩



Latifa (🎓 1 · 🐼)

宁夏大学研究生

2016/11/17

Python

```
1 amp = 0xFF
2 key = cv2.waitKey(1) & amp
```

👍 赞 回复 ↩



narcissus361 (🎓 1)

2017/02/28

输入命令以后报错：for c in cnts:

语法错误，怎么修改

👍 赞 回复 ↩



narcissus361 (🎓 1)

2017/03/01

已解决，是前面一个语句没有写完

👍 赞 回复 ↩



太布小凡

02/07

你好，怎么得到那些视频，我这里看不了视频。



霜烛 觞焰 (6)

02/19

请问可以的到您的联系方式嘛？

👍 赞 回复 ↩



i

02/28

```
C:\Users\L\Desktop\detector>python motion_detector.py
Traceback (most recent call last):
File "motion_detector.py", line 52, in <module>
cv2.CHAIN_APPROX_SIMPLE)
ValueError: too many values to unpack (expected 2)
```

请问这个错误是什么原因啊

👍 1 赞 回复 ↩

Python小组话题

我有新话题 💬



[python2.7如何输出文件中的汉字](#)
心心。 发起 • 35 回复



[小弟机械行业3年了，自学了python半年，想...](#)
阅微 发起 • 19 回复




[2年Java, 想转 python](#)
大概会吧 发起 • 18 回复

[零基础自学Python感觉很难，不像大...](#)
keepcalm 发起 • 96 回复



[Python学习, 有哪些方向可以选择](#)
小丑的哭笑 发起 • 33 回复

[python 真的能在人工智能领域 一骑...](#)

 泽恒 发起 • 10 回复



- [本周热门Python文章](#)
- [本月热门](#)
- [热门标签](#)



[Python工具资源](#)

[更多资源 >>](#)



[Tryton : 一个通用商务框架](#) [杂项](#)



[NLTK : 一个先进的用来处理自然语言数据的Python程序。](#) [自然语言处理](#) · [3](#)



[PyMC : 马尔科夫链蒙特卡洛采样工具](#) [科学计算与分析](#)



[statsmodels : 统计建模和计量经济学](#) [科学计算与分析](#)



[Pylearn2 : 一个基于Theano的机器学习库](#)

机器学习 · [Q1](#)

关于 Python 频道

Python频道分享 Python 开发技术、相关的行业动态。

快速链接

[网站使用指南](#) »

[加入我们](#) »

[问题反馈与求助](#) »

[网站积分规则](#) »

[网站声望规则](#) »

关注我们

新浪微博：[@Python开发者](#)

RSS：[订阅地址](#)

推荐微信号



Python开发者



算法爱好者



大数据与机器学习文摘

合作联系

Email：[bd@Jobbole.com](mailto:bd@jobbole.com)

QQ：2302462408（加好友请注明来意）

更多频道

[小组](#) – 好的话题、有启发的回复、值得信赖的圈子

[头条](#) – 分享和发现有价值的内容与观点

[相亲](#) – 为IT单身男女服务的征婚传播平台

[资源](#) – 优秀的工具资源导航

[翻译](#) – 翻译传播优秀的外文文章

[文章](#) – 国内外的精选文章

[设计](#) – UI,网页，交互和用户体验

[iOS](#) – 专注iOS技术分享

[安卓](#) – 专注Android技术分享

[前端](#) – JavaScript, HTML5, CSS

[Java](#) – 专注Java技术分享

[Python](#) – 专注Python技术分享



