# **使用opencv-python读取多个(海康\大华)网络摄像头的视频流,解决实时读取延迟问题**

<https://blog.csdn.net/ljx1400052550/article/details/106987943>

在上一篇博客中，主要介绍了python之opencv按帧提取视频中的图片，但是，由于最近在做人脸识别的项目，用的是大华的监控摄像头，我发现大华的摄像头实时读取延迟问题特别严重，尤其是主码流,这个问题困扰了我好久，最终想到的方式就是自己实时推流，经过实践，终于解决了实时读取延迟问题。

前言

同样需要准备对应的python开发环境，具体参考上一篇python之opencv按帧提取视频中的图片，里面介绍了详细的需要的库文件。

好了，既然是自己实现实时预览推流，那就要确定使用推流方式，我这里使用的是RTSP地址和格式实现推流的，下面是我归纳的各大监控摄像头厂商的RTSP具体推流格式。

各大摄像头厂商RTSP推流格式

海康实时流

rtsp://[username]:[password]@[ip]:[port]/[codec]/[channel]/[subtype]/av\_stream

说明：

username: 用户名。例如admin。

password: 密码。例如123456。

ip: 为设备IP。例如 192.168.0.224。

port: 端口号默认为554，若为默认可不填写。

codec：有h264、MPEG-4、mpeg4这几种。

channel: 通道号，起始为1。例如通道1，则为ch1。

subtype: 码流类型，主码流为main，子码流为sub。

举个栗子：

例如，请求海康摄像机通道1的主码流，Url如下

rtsp://admin:123456@192.168.0.224:554/h264/ch1/main/av\_stream

rtsp://admin:123456@192.168.0.224:554/MPEG-4/ch1/main/av\_stream

rtsp://admin:123456@192.168.0.224:554/h264/ch33/main/av\_stream //ipc

例如,请求海康摄像机通道1的子码流，Url如下：

rtsp://admin:123456@192.168.0.224/mpeg4/ch1/sub/av\_stream

rtsp://admin:123456@192.168.0.224/h264/ch1/sub/av\_stream

【新版本】URL：

rtsp://username:password@<address>:<port>/Streaming/Channels/<id>(?parm1=value1&parm2-=value2…)

注：VLC可以支持解析URL里的用户名密码，实际发给设备的RTSP请求不支持带用户名密码。

举例：

DS-9632N-ST的模拟通道01主码流：

rtsp://admin:123456@192.168.0.224:554/Streaming/Channels/101?transportmode=unicast

DS-9016HF-ST的IP通道01主码流：

rtsp://admin:123456@192.168.0.224:554/Streaming/Channels/1701?transportmode=unicast

DS-9016HF-ST的模拟通道01子码流：

rtsp://admin:123456@192.168.0.224:554/Streaming/Channels/102?transportmode=unicast (单播)

rtsp://admin:123456@192.168.0.224:554/Streaming/Channels/102?transportmode=multicast (多播)

rtsp://admin:123456@192.168.0.224:554/Streaming/Channels/102 (?后面可省略，默认单播)

注：前面老URL，NVR（>=64路的除外）的IP通道从33开始；新URL，通道号全部按顺序从1开始。

大华

rtsp://username:password@ip:port/cam/realmonitor?channel=1&subtype=0

说明:

username: 用户名，例如admin。

password: 密码，例如admin。

ip: 为设备IP，例如192.168.0.224。

port: 端口号默认为554，若为默认可不填写。

channel: 通道号，起始为1；例如通道2，则为channel=2。

subtype: 码流类型，主码流为0（即subtype=0）；子码流为1（即subtype=1）。

举个栗子：

例如，请求某设备的通道2的子码流，Url如下

rtsp://admin:admin@192.168.0.224:554/cam/realmonitor?channel=2&subtype=1

雄迈/巨峰

默认IP地址：192.168.0.224

用户名: admin

密码空:123456

端口：TCP端口：34567 和 HTTP端口：80，onvif端口是8899

举个栗子：

RTSP地址：rtsp://192.168.0.224 :554/user=admin&password=123456&channel=1&stream=0.sdp?real\_stream

192.168.0.224 这个是被连接的设备的IP

554这个是RTSP服务的端口号，可以在设备的网络服务里面更改

user=admin这个是设备的登录用户名

password= 123456

channel=1 第一通道

stream=0.sdp？主码流

stream=1.sdp？副码流

天视通

默认IP地址：192.168.0.224

用户名admin

密码123456

端口：http端口80 数据端口8091 RTSP端口554 ONVIF端口 80

举个栗子：

RTSP地址(不需要密码)：

主码流地址:rtsp://192.168.0.224 :554/mpeg4

子码流地址:rtsp://192.168.0.224 :554/mpeg4cif

RTSP地址(需要密码)：

主码流 rtsp://admin:123456@192.168.0.224 :554/mpeg4

子码流 rtsp://admin:123456@192.168.0.224 :554/mpeg4cif

中维/尚维

默认IP地址：DHCP 默认（0.0.0.0）

用户名admin 默认

密码 空

举个栗子：

RTSP地址：rtsp://0.0.0.0:8554/live1.264（次码流）

rtsp://0.0.0.0:8554/live0.264 (主码流)

九安

RTSP地址：rtsp://IP:port（website port）/ch0\_0.264（主码流）

rtsp://IP:port（website port）/ch0\_1.264（子码流）

技威/YOOSEE

默认IP地址：DHCP 用户名admin 密码123456

RTSP地址：主码流：rtsp://IPadr:554/onvif1

次码流：rtsp://IPadr:554/onvif2

onvif端口是5000

设备发现的端口是3702

V380

默认IP地址：DHCP 用户名admin 密码空/admin

onvif端口8899

RTSP地址：主码流rtsp://ip//live/ch00\_1

子码流rtsp://ip//live/ch00\_0

宇视

默认IP地址： 192.168.0.13/DHCP 默认用户名 admin 和默认密码 123456

端口：HTTP 80/RTSP 554/HTTPS 110(443)/onvif端口 80

RTSP地址：rtsp://用户名:密码@ip:端口号/video1/2/3，分别对应主/辅/三码流；

举个栗子：

rtsp://admin:admin@192.168.8.8:554/video1，就表示主码流；

rtsp://admin:admin@192.168.8.8:554/video2，表示子码流；

rtsp://admin:admin@192.168.8.8:554/video3，表示3码流；

天地伟业

默认IP地址：192.168.1.2 用户名“Admin”、密码“1111”

onvif端口号“8080”

RTSP地址：rtsp：//192.168.1.2

巨龙/JVT

默认IP地址：192.168.1.88 默认用户名 admin 默认密码admin

RTSP地址：

主码流地址:rtsp://IP地址/av0\_0

次码流地址:rtsp://IP地址/av0\_1

onvif端口 2000

海清

RTSP地址：rtsp://用户名:密码@ip:端口号/av0\_0

D-Link

rtsp://[username]:[password]@[ip]:[port]/[channel].sdp

说明：

username：用户名。例如admin

password：密码。例如12345，如果没有网络验证可直接写成rtsp:// [ip]:[port]/[channel].sdp

ip：为设备IP。例如192.168.0.108。

port：端口号默认为554，若为默认可不填写。

channel：通道号，起始为1。例如通道2，则为live2。

举个栗子：

例如，请求某设备的通道2的码流，URL如下

rtsp://admin:12345@192.168.200.201:554/live2.sdp

Axis（安讯士）

rtsp://[username]:[password]@[ip]/axis-media/media.amp?[videocodec]&[resolution]

说明：

username：用户名。例如admin

password：密码。例如12345，如果没有网络验证可省略用户名密码部分以及@字符。

ip：为设备IP。例如192.168.0.108。

videocodec：支持MPEG、h.264等，可缺省。

resolution：分辨率，如resolution=1920x1080，若采用默认分辨率，可缺省此参数。

举个栗子：

例如，请求某设备h264编码的1280x720的码流，URL如下：

rtsp:// 192.168.200.202/axis-media/media.amp?videocodec=h264&resolution=1280x720

好了。支持，市场上主流的监控摄像头RTSP推流就介绍完毕了，接下来就实战RTSP实时推流吧。这里一大华摄像头为栗子。

实战

在上一篇，我们知道了，开启实时预览的方式，需要开启opencv VideoCapture，细心一点你会发现，在上一篇中有这样的代码如下：

# 导入所需要的库

import cv2

import numpy as np

# 读取视频文件

videoCapture = cv2.VideoCapture("test.mp4")

# 通过摄像头的方式

# videoCapture=cv2.VideoCapture(1)

经过分析，你会发现，我们只需要把cv2.VideoCapture(“test.mp4”)这里做成实时推流的即可。

一：开启RTSP：

在前面，我们知道了大华摄像头的RTSP推流方式，那好，第一步就先实现RTSP推流吧。代码如下：

import cv2

import time

import multiprocessing as mp

def image\_put(q, name, pwd, ip, channel=1):

//使用占位符,动态的代替ip地址,用户名,密码,预览通道等参数

cap = cv2.VideoCapture("rtsp://%s:%s@%s//Streaming/Channels/%d" % (name, pwd, ip, channel))

if cap.isOpened():

print('HIKVISION')

else:

cap = cv2.VideoCapture("rtsp://%s:%s@%s/cam/realmonitor?channel=%d&subtype=0" % (name, pwd, ip, channel))

print('DaHua')

while True:

q.put(cap.read()[1])

q.get() if q.qsize() > 1 else time.sleep(0.01)

def image\_get(q, window\_name):

cv2.namedWindow(window\_name, flags=cv2.WINDOW\_FREERATIO)

while True:

frame = q.get()

cv2.imshow(window\_name, frame)

cv2.waitKey(1)

def run\_multi\_camera():

# user\_name, user\_pwd = "admin", "password"

user\_name, user\_pwd = "admin", "admin123456"

camera\_ip\_l = [

"192.168.35.121", # ipv4

"[fe80::3aaf:29ff:fed3:d260]", # ipv6

# 把你的摄像头的地址放到这里，如果是ipv6，那么需要加一个中括号。

]

二：多线程队列解决实时阅览延迟问题：

上面，我们知道了，如何实现实时预览，下面就解决一下核心问题，实时读取延迟问题,代码如下：

import multiprocessing as mp

...

img\_queues = [mp.Queue(maxsize=2) for \_ in camera\_ip\_l] # queue

...

q.put(frame) if is\_opened else None # 线程A不仅将图片放入队列

q.get() if q.qsize() > 1 else time.sleep(0.01) # 线程A还负责移除队列中的旧图

...

好了，完成了，这俩步，就可以解决实时读取延迟问题了，最后附上完整代码。

完整代码：

import cv2

import time

import multiprocessing as mp

def image\_put(q, name, pwd, ip, channel=1):

cap = cv2.VideoCapture("rtsp://%s:%s@%s//Streaming/Channels/%d" % (name, pwd, ip, channel))

if cap.isOpened():

print('HIKVISION')

else:

cap = cv2.VideoCapture("rtsp://%s:%s@%s/cam/realmonitor?channel=%d&subtype=0" % (name, pwd, ip, channel))

print('DaHua')

while True:

q.put(cap.read()[1])

q.get() if q.qsize() > 1 else time.sleep(0.01)

def image\_get(q, window\_name):

cv2.namedWindow(window\_name, flags=cv2.WINDOW\_FREERATIO)

while True:

frame = q.get()

cv2.imshow(window\_name, frame)

cv2.waitKey(1)

def run\_multi\_camera():

# user\_name, user\_pwd = "admin", "password"

user\_name, user\_pwd = "admin", "admin123456"

camera\_ip\_l = [

"192.168.35.121", # ipv4

"[fe80::3aaf:29ff:fed3:d260]", # ipv6

# 把你的摄像头的地址放到这里，如果是ipv6，那么需要加一个中括号。

]

mp.set\_start\_method(method='spawn') # init

queues = [mp.Queue(maxsize=4) for \_ in camera\_ip\_l]

processes = []

for queue, camera\_ip in zip(queues, camera\_ip\_l):

processes.append(mp.Process(target=image\_put, args=(queue, user\_name, user\_pwd, camera\_ip)))

processes.append(mp.Process(target=image\_get, args=(queue, camera\_ip)))

for process in processes:

process.daemon = True

process.start()

for process in processes:

process.join()

if \_\_name\_\_ == '\_\_main\_\_':

run\_multi\_camera()

当然还有更简单的实现方式，下面看看如何利用OpenCV官网给出的视频流读取吧

简单版-OpenCV官网给出的视频流读取示例

经过简单修改，如下：

def run\_opencv\_camera():

video\_stream\_path = 0 # local camera (e.g. the front camera of laptop)

cap = cv2.VideoCapture(video\_stream\_path)

while cap.isOpened():

is\_opened, frame = cap.read()

cv2.imshow('frame', frame)

cv2.waitKey(1)

cap.release()

当 video\_stream\_path = 0 的时候，电脑会开启默认摄像头，比如笔记本电脑的前置摄像头 。

当我们需要读取网络摄像头的时候，我们可以对 cap = cv2.VideoCapture(括号里面的东西进行修改)，填写上我们想要读取的视频流，它可以是：

List item数字0，代表计算机的默认摄像头（例如上面提及的笔记本前置摄像头）

video.avi 视频文件的路径，支持其他格式的视频文件

rtsp路径（不同品牌的路径一般是不同的，如下面举出的海康与大华）

user, pwd, ip, channel = "admin", "admin123456", "192.168.35.121", 1

video\_stream\_path = 0 # local camera (e.g. the front camera of laptop)

video\_stream\_path = 'video.avi' # the path of video file

video\_stream\_path = "rtsp://%s:%s@%s/h265/ch%s/main/av\_stream" % (user, pwd, ip, channel) # HIKIVISION old version 2015

video\_stream\_path = "rtsp://%s:%s@%s//Streaming/Channels/%d" % (user, pwd, ip, channel) # HIKIVISION new version 2017

video\_stream\_path = "rtsp://%s:%s@%s/cam/realmonitor?channel=%d&subtype=0" % (user, pwd, ip, channel) # dahua

cap = cv2.VideoCapture(video\_stream\_path)

具体参考：OpenCV官网给出的视频流读取示例代码

好了，到此，我们就解决实时读取延迟问题，但是，目前我们只是监控一路，如何监控多路，解决实时读取延迟问题了,其实很简单，因为每一路是独立，互不干涉，下面就实战多个摄像头。

实时预览多路摄像头

有了单路的思路，你会发现，多路只要使用多线程队列，就能解决延迟卡顿问题，读取多个摄像头。

def image\_put(q, user, pwd, ip, channel=1):

cap = cv2.VideoCapture("rtsp://%s:%s@%s//Streaming/Channels/%d" % (user, pwd, ip, channel))

if cap.isOpened():

print('HIKVISION')

else:

cap = cv2.VideoCapture("rtsp://%s:%s@%s/cam/realmonitor?channel=%d&subtype=0" % (user, pwd, ip, channel))

print('DaHua')

while True:

q.put(cap.read()[1])

q.get() if q.qsize() > 1 else time.sleep(0.01)

def image\_get(q, window\_name):

cv2.namedWindow(window\_name, flags=cv2.WINDOW\_FREERATIO)

while True:

frame = q.get()

cv2.imshow(window\_name, frame)

cv2.waitKey(1)

def run\_single\_camera():

user\_name, user\_pwd, camera\_ip = "admin", "admin123456", "192.168.35.121"

mp.set\_start\_method(method='spawn') # init

queue = mp.Queue(maxsize=2)

processes = [mp.Process(target=image\_put, args=(queue, user\_name, user\_pwd, camera\_ip)),

mp.Process(target=image\_get, args=(queue, camera\_ip))]

[process.start() for process in processes]

[process.join() for process in processes]

def run\_multi\_camera():

# user\_name, user\_pwd = "admin", "password"

user\_name, user\_pwd = "admin", "admin123456"

camera\_ip\_l = [

"172.20.114.26", # ipv4

"[fe80::3aaf:29ff:fed3:d260]", # ipv6

]

mp.set\_start\_method(method='spawn') # init

queues = [mp.Queue(maxsize=4) for \_ in camera\_ip\_l]

processes = []

for queue, camera\_ip in zip(queues, camera\_ip\_l):

processes.append(mp.Process(target=image\_put, args=(queue, user\_name, user\_pwd, camera\_ip)))

processes.append(mp.Process(target=image\_get, args=(queue, camera\_ip)))

for process in processes:

process.daemon = True

process.start()

for process in processes:

process.join()

if \_\_name\_\_ == '\_\_main\_\_':

# run\_single\_camera()

run\_multi\_camera()

pass

关键部分解释：

我使用Python3自带的多线程模块，创建一个队列，线程A从通过rtsp协议从视频流中读取出每一帧，并放入队列中，线程B从队列中将图片取出，处理后进行显示。线程A如果发现队列里有两张图片（证明线程B的读取速度跟不上线程A），那么线程A主动将队列里面的旧图片删掉，换上新图片。通过多线程的方法：

线程A的读取速度始终不收线程B的影响，防止网络摄像头的缓存区爆满

线程A更新了队列中的图片，使线程B始终读取到最新的画面，降低了延迟

import multiprocessing as mp

...

img\_queues = [mp.Queue(maxsize=2) for \_ in camera\_ip\_l] # queue

...

q.put(frame) if is\_opened else None # 线程A不仅将图片放入队列

q.get() if q.qsize() > 1 else time.sleep(0.01) # 线程A还负责移除队列中的旧图

...

好了，支持，多路和单路实时预览效果，便实现了，下面看一下具体的实现效果吧：