PYIMAGESEARCH

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

EMBEDDED/IOT AND COMPUTER VISION (HTTPS://WWW.PYIMAGESEARCH.COM/CATEGORY/EMBEDDED/)

IOT (HTTPS://WWW.PYIMAGESEARCH.COM/CATEGORY/IOT/)

RASPBERRY PI (HTTPS://WWW.PYIMAGESEARCH.COM/CATEGORY/RASPBERRY-PI/)

TUTORIALS (HTTPS://WWW.PYIMAGESEARCH.COM/CATEGORY/TUTORIALS/)

OpenCV – Stream video to web browser/HTML page

by Adrian Rosebrock (https://www.pyimagesearch.com/author/adrian/) *on* September 2, 2019

Ever have your car stolen?

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

Mine was stolen over the weekend. And let me tell you, *I'm pissed*.

I can't share too many details as it's an active criminal investigation, but here's what I can tell you:

My wife and I moved to Philadelphia, PA from Norwalk, CT about six months ago. I have a car, which I don't drive often, but still keep just in case of emergencies.

Parking is hard to find in our neighborhood, so I was in need of a parking garage.

I heard about a garage, signed up, and started parking my car there.

Fast forward to this past Sunday.

My wife and I arrive at the parking garage to grab my car. We were about to head down to Maryland to visit my parents and have some blue crab (Maryland is famous for its crabs).

I walked to my car and took off the cover.

I was immediately confused — this isn't my car.

Where the #$&@ is my car?

After a few short minutes I realized the reality — my car was stolen.

Over the past week, my work on my upcoming ***Raspberry Pi for Computer Vision*** **(https://www.pyimagesearch.com/raspberry-pi-for-computer-vision/)** book was interrupted — I've been working with the owner of the the parking garage, the Philadelphia Police Department, and the GPS tracking service on my car to figure out what happened.

I can't publicly go into any details until it's resolved, but let me tell you, there's a whole mess of paperwork, police reports, attorney letters, and insurance claims that I'm wading neck-deep through.

I'm *hoping* that this issue gets resolved in the next month — I hate distractions, especially

I've managed to use my frustrations to inspire a new **security-related computer vision blog post.**

Click here to download the source code to this post
(//app.monstercampaigns.com/c/tortsem7gkvvuxc4cyfj)

In this post, we'll learn how to **stream video to a web browser using Flask and OpenCV.**

You will be able to deploy the system on a Raspberry Pi in **less than 5 minutes:**

- Simply install the required packages/software and start the script.
- Then open your computer/smartphone browser to navigate to the URL/IP address to watch the video feed (and ensure nothing of yours is stolen).

There's nothing like a little video evidence to catch thieves.

While I continue to do paperwork with the police, insurance, etc, you can begin to arm yourself with Raspberry Pi cameras to catch bad guys wherever you live and work.

To learn how to use OpenCV and Flask to stream video to a web browser HTML page, *just keep reading!*

Looking for the source code to this post?

JUMP RIGHT TOO THE DOWNLOADS SECTION →

OpenCV – Stream video to web browser/HTML page

programming language.

We'll learn the fundamentals of motion detection so that we can apply it to our project. We'll proceed to implement motion detection by means of a background subtractor.

Click here to download the source code to this post (http://app.monstercampaigns.com/c/tortsen7qkvyexc4cyfi)

From there, we will combine Flask with OpenCV, enabling us to:

1. Access frames from RPi camera module or USB webcam.
2. Process the frames and apply an arbitrary algorithm (here we'll be using background subtraction/motion detection, but you could apply image classification, object detection, etc.).
3. Stream the results to a web page/web browser.

Additionally, the code we'll be covering will be able to support multiple clients (i.e., more than one person/web browser/tab accessing the stream at once), something the vast majority of examples you will find online cannot handle.

Putting all these pieces together results in a home surveillance system capable of performing motion detection and then streaming the video result to your web browser.

Let's get started!

The Flask web framework

(https://www.pyimagesearch.com/wp-

(https://flask.palletsprojects.com)).

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

In this section we'll briefly discuss the Flask web framework and how to install it on your system.

Flask (https://palletsprojects.com/p/flask/) is a popular micro web framework written in the Python programming language.

Along with **Django (https://www.djangoproject.com/)**, Flask is one of the most common web frameworks you'll see when building web applications using Python.

However, unlike Django, Flask is *very* lightweight, making it super easy to build basic web applications.

As we'll see in this section, we'll only need a small amount of code to facilitate live video streaming with Flask — the rest of the code either involves (1) OpenCV and accessing our video stream or (2) ensuring our code is thread safe and can handle multiple clients.

If you ever need to install Flask on a machine, it's as simple as the following command:

```
OpenCV - Stream video to web browser/HTML page
$ pip install flask
```

While you're at it, go ahead and install NumPy, OpenCV, and imutils:

```
OpenCV - Stream video to web browser/HTML page
$ pip install numpy
$ pip install opencv-contrib-python
$ pip install imutils
```

Note: *If you'd like the full-install of OpenCV including "non-free" (patented) algorithms, be sure to* ***compile OpenCV from source (https://www.pyimagesearch.com/opencv-tutorials-resources-guides/)****.*

Project structure

Before we move on, let's take a look at our directory structure for the project:

```
3. |   ├── pyimagesearch
4. |   |   ├── motion_detection
5. |   |   |   ├── __init__.py
6. |   |   |   └── singlemotiondetector.py
7. |   |   └── __init__.py
8. |   ├── templates
9. |   |   └── index.html
10. |  └── webstreaming.py
11. |
12. |  3 directories, 5 files
```

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

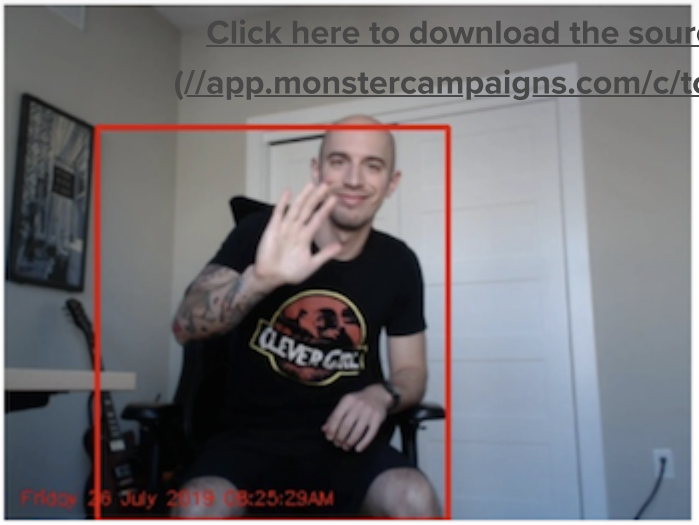
To perform background subtraction and motion detection we'll be implementing a class named `SingleMotionDetector` — this class will live inside the `singlemotiondetector.py` file found in the `motion_detection` submodule of `pyimagesearch`.

The `webstreaming.py` file will use OpenCV to access our web camera, perform motion detection via `SingleMotionDetector`, and then serve the output frames to our web browser via the Flask web framework.

In order for our web browser to have something to display, we need to populate the contents of `index.html` with HTML used to serve the video feed. We'll only need to insert some basic HTML markup — Flask will handle actually sending the video stream to our browser for us.

Implementing a basic motion detector

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)



(https://www.pyimagesearch.com/wp-content/uploads/2019/09/opencv_stream_video_flask_browser_video_surveillance_result_03.png)

Figure 2: Video surveillance with Raspberry Pi, OpenCV, Flask and web streaming. By use of background subtraction for motion detection, we have detected motion where I am moving in my chair.

Our motion detector algorithm will detect motion by form of ***background subtraction***.

Most background subtraction algorithms work by:

1. Accumulating the weighted average of the previous *N* frames
2. Taking the *current frame* and subtracting it from the weighted average of frames
3. Thresholding the output of the subtraction to highlight the regions with substantial differences in pixel values ("white" for foreground and "black" for background)
4. Applying basic image processing techniques such as erosions and dilations to remove

Our motion detection implementation will live inside the `SingleMotionDetector` class which can be found in the source code.

Click here to download the source code to this post (http://app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

We call this a "single motion detector" as the algorithm itself is only interested in finding the *single, largest* region of motion.

We can easily extend this method to handle multiple regions of motion as well.

Let's go ahead and **implement the motion detector.**

Open up the `singlemotiondetector.py` file and insert the following code:

```
OpenCV - Stream video to web browser/HTML page
 1. # import the necessary packages
 2. import numpy as np
 3. import imutils
 4. import cv2
 5.
 6. class SingleMotionDetector:
 7.    def __init__(self, accumWeight=0.5):
 8.        # store the accumulated weight factor
 9.        self.accumWeight = accumWeight
10.
11.        # initialize the background model
12.        self.bg = None
```

Lines 2-4 handle our required imports.

All of these are fairly standard, including NumPy for numerical processing, `imutils` for our convenience functions, and `cv2` for our OpenCV bindings.

We then define our `SingleMotionDetector` class on **Line 6**. The class accepts an optional argument, `accumWeight`, which is the factor used to our accumulated weighted average.

The larger `accumWeight` is, the *less* the background (`bg`) will be factored in when accumulating the weighted average.

Conversely, the *smaller* `accumWeight` is, the *more* the background `bg` will be considered

recommend this as a starting point value (you can then adjust it based on your own experiments).

Next, let's define the `update` method which will accept an input frame and compute the weighted average:

```
OpenCV - Stream video to web browser/HTML page
14.     def update(self, image):
15.         # if the background model is None, initialize it
16.         if self.bg is None:
17.             self.bg = image.copy().astype("float")
18.             return
19.
20.         # update the background model by accumulating the weighted
21.         # average
22.         cv2.accumulateWeighted(image, self.bg, self.accumWeight)
```

In the case that our `bg` frame is `None` (implying that `update` has never been called), we simply store the `bg` frame (**Lines 15-18**).

Otherwise, we compute the weighted average between the input `frame`, the existing background `bg`, and our corresponding `accumWeight` factor.

Given our background `bg` we can now apply motion detection via the `detect` method:

```
OpenCV - Stream video to web browser/HTML page
24.     def detect(self, image, tVal=25):
25.         # compute the absolute difference between the background model
26.         # and the image passed in, then threshold the delta image
27.         delta = cv2.absdiff(self.bg.astype("uint8"), image)
28.         thresh = cv2.threshold(delta, tVal, 255, cv2.THRESH_BINARY)[1]
29.
30.         # perform a series of erosions and dilations to remove small
31.         # blobs
32.         thresh = cv2.erode(thresh, None, iterations=2)
33.         thresh = cv2.dilate(thresh, None, iterations=2)
```

The `detect` method requires a single parameter along with an optional one:

- `image`: The input frame/image that motion detection will be applied to.
- `tVal`: The threshold value used to mark a particular pixel as "motion" or not.

Any pixel locations that have a difference `>= tVal` are set to 255 (white; foreground), otherwise they are set to 0 (black; background) (**Line 28**).

A series of erosions and dilations are performed to remove noise and small, localized areas of motion that would otherwise be considered false-positives (likely due to reflections or rapid changes in light).

The next step is to apply contour detection to extract any motion regions:

```
OpenCV - Stream video to web browser/HTML page
35.         # find contours in the thresholded image and initialize the
36.         # minimum and maximum bounding box regions for motion
37.         cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
38.             cv2.CHAIN_APPROX_SIMPLE)
39.         cnts = imutils.grab_contours(cnts)
40.         (minX, minY) = (np.inf, np.inf)
41.         (maxX, maxY) = (-np.inf, -np.inf)
```

Lines 37-39 perform contour detection on our `thresh` image.

We then initialize two sets of bookkeeping variables to keep track of the location where any motion is contained (**Lines 40 and 41**). These variables will form the “bounding box” which will tell us the location of where the motion is taking place.

The final step is to populate these variables (provided motion exists in the frame, of course):

```
OpenCV - Stream video to web browser/HTML page
43.         # if no contours were found, return None
44.         if len(cnts) == 0:
45.             return None
46.
47.         # otherwise, loop over the contours
48.         for c in cnts:
49.             # compute the bounding box of the contour and use it to
50.             # update the minimum and maximum bounding box regions
51.             (x, y, w, h) = cv2.boundingRect(c)
52.             (minX, minY) = (min(minX, x), min(minY, y))
53.             (maxX, maxY) = (max(maxX, x + w), max(maxY, y + h))
54.
55.         # otherwise, return a tuple of the thresholded image along
56.         # with bounding box
57.         return (thresh, (minX, minY, maxX, maxY))
```

If that's the case, then there was no motion found in the frame and we can safely ignore it.

Otherwise, motion *does* exist in the frame so we need to start looping over the contours (**Line 48**).

For each contour we compute the bounding box and then update our bookkeeping variables (**Lines 47-53**), finding the minimum and maximum *(x, y)*-coordinates that all motion has taken place it.

Finally, we return the bounding box location to the calling function.

Combining OpenCV with Flask

(https://www.pyimagesearch.com/wp-content/uploads/2019/09/opencv_stream_video_flask_browser_flask_and_opencv.png)

(running on a Raspberry Pi) to a web browser.

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

Open up the `webstreaming.py` file in your project structure and insert the following code:

```
OpenCV - Stream video to web browser/HTML page
 1. # import the necessary packages
 2. from pyimagesearch.motion_detection import SingleMotionDetector
 3. from imutils.video import VideoStream
 4. from flask import Response
 5. from flask import Flask
 6. from flask import render_template
 7. import threading
 8. import argparse
 9. import datetime
10. import imutils
11. import time
12. import cv2
```

Lines 2-12 handle our required imports:

- **Line 2** imports our `SingleMotionDetector` class which we implemented above.
- The `VideoStream` class (**Line 3**) will enable us to access our Raspberry Pi camera module or USB webcam.
- **Lines 4-6** handle importing our required Flask packages — we'll be using these packages to render our `index.html` template and serve it up to clients.
- **Line 7** imports the `threading` library to ensure we can support concurrency (i.e., multiple clients, web browsers, and tabs at the same time).

Let's move on to performing a few initializations:

```
OpenCV - Stream video to web browser/HTML page
14. # initialize the output frame and a lock used to ensure thread-safe
15. # exchanges of the output frames (useful when multiple browsers/tabs
16. # are viewing the stream)
17. outputFrame = None
18. lock = threading.Lock()
19.
20. # initialize a flask object
21. app = Flask(__name__)
```

```
26. | vs = VideoStream(src=0).start()
27. | time.sleep(2.0)
```


First, we initialize our `outputFrame` on **Line 17** — this will be the frame (post-motion detection) that will be served to the clients.

We then create a `lock` on **Line 18** which will be used to ensure thread-safe behavior when updating the `ouputFrame` (i.e., ensuring that one thread isn't trying to read the frame as it is being updated).

Line 21 initialize our Flask `app` itself while **Lines 25-27** access our video stream:

- If you are using a USB webcam, you can leave the code *as is*.
- However, if you are using a RPi camera module you should ***uncomment*** **Line 25** and ***comment out*** **Line 26.**

The next function, `index`, will render our `index.html` template and serve up the output video stream:

```
    OpenCV - Stream video to web browser/HTML page
29. | @app.route("/")
30. | def index():
31. |     # return the rendered template
32. |     return render_template("index.html")
```

This function is quite simplistic — all it's doing is calling the Flask `render_template` on our HTML file.

We'll be reviewing the `index.html` file in the next section so we'll hold off on a further discussion on the file contents until then.

Our next function is responsible for:

1. Looping over frames from our video stream
2. Applying motion detection
3. Drawing any results on the `outputFrame`

Let's take a look at this function now:

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

```
OpenCV - Stream video to web browser/HTML page
34. def detect_motion(frameCount):
35.     # grab global references to the video stream, output frame, and
36.     # lock variables
37.     global vs, outputFrame, lock
38.
39.     # initialize the motion detector and the total number of frames
40.     # read thus far
41.     md = SingleMotionDetector(accumWeight=0.1)
42.     total = 0
```

Our `detection_motion` function accepts a single argument, `frameCount`, which is the minimum number of required frames to build our background `bg` in the `SingleMotionDetector` class:

- If we don't have at least `frameCount` frames, we'll continue to compute the accumulated weighted average.
- Once `frameCount` is reached, we'll start performing background subtraction.

Line 37 grabs global references to three variables:

- `vs`: Our instantiated `VideoStream` object
- `outputFrame`: The output frame that will be served to clients
- `lock`: The thread lock that we must obtain before updating `outputFrame`

Line 41 initializes our `SingleMotionDetector` class with a value of `accumWeight=0.1`, implying that the `bg` value will be weighted higher when computing the weighted average.

Line 42 then initializes the `total` number of frames read thus far — we'll need to ensure a sufficient number of frames have been read to build our background model.

From there, we'll be able to perform background subtraction.

```
44.     # loop over frames from the video stream
45.     while True:
46.         # read the next frame from the video stream, resize it,
47.         # convert the frame to grayscale and blur it
48.         frame = vs.read()
49.         frame = imutils.resize(frame, width=400)
50.         gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
51.         gray = cv2.GaussianBlur(gray, (7, 7), 0)
52.
53.         # grab the current timestamp and draw it on the frame
54.         timestamp = datetime.datetime.now()
55.         cv2.putText(frame, timestamp.strftime(
56.             "%A %d %B %Y %I:%M:%S%p"), (10, frame.shape[0] - 10),
57.             cv2.FONT_HERSHEY_SIMPLEX, 0.35, (0, 0, 255), 1)
```


Line 48 reads the next `frame` from our camera while **Lines 49-51** perform preprocessing, including:

- Resizing to have a width of 400px (the *smaller* our input frame is, the *less data* there is, and thus the *faster* our algorithms will run).
- Converting to grayscale.
- Gaussian blurring (to reduce noise).

We then grab the current timestamp and draw it on the `frame` (**Lines 54-57**).

With one final check, we can perform motion detection:

```
    OpenCV - Stream video to web browser/HTML page
59.         # if the total number of frames has reached a sufficient
60.         # number to construct a reasonable background model, then
61.         # continue to process the frame
62.         if total > frameCount:
63.             # detect motion in the image
64.             motion = md.detect(gray)
65.
66.             # check to see if motion was found in the frame
67.             if motion is not None:
68.                 # unpack the tuple and draw the box surrounding the
69.                 # "motion area" on the output frame
70.                 (thresh, (minX, minY, maxX, maxY)) = motion
71.                 cv2.rectangle(frame, (minX, minY), (maxX, maxY),
72.                     (0, 0, 255), 2)
```

```
77.             total += 1
78.
79.             # acquire the lock, set the output frame, and release the
80.             # lock
81.             with lock:
82.                 outputFrame = frame.copy()
```


On **Line 62** we ensure that we have read at least `frameCount` frames to build our background subtraction model.

If so, we apply the `.detect` motion of our motion detector, which returns a single variable, `motion`.

If `motion` is `None`, then we know no motion has taken place in the current `frame`. Otherwise, if `motion` is *not* `None` (**Line 67**), then we need to draw the bounding box coordinates of the motion region on the `frame`.

Line 76 updates our motion detection background model while **Line 77** increments the `total` number of frames read from the camera thus far.

Finally, **Line 81** acquires the `lock` required to support thread concurrency while **Line 82** sets the `outputFrame`.

We need to acquire the lock to ensure the `outputFrame` variable is not accidentally being *read* by a client while we are trying to *update* it.

Our next function, `generate`, is a Python generator used to encode our `outputFrame` as JPEG data — let's take a look at it now:

```
OpenCV - Stream video to web browser/HTML page
84.  def generate():
85.     # grab global references to the output frame and lock variables
86.     global outputFrame, lock
87.
88.     # loop over frames from the output stream
89.     while True:
90.         # wait until the lock is acquired
91.         with lock:
92.             # check if the output frame is available, otherwise skip
93.             # the iteration of the loop
94.             if outputFrame is None:
95.                 continue
```

```
            # ensure the frame was successfully encoded
            if not flag:
                continue

        # yield the output frame in the byte format
        yield(b'--frame\r\n' b'Content-Type: image/jpeg\r\n\r\n' +
            bytearray(encodedImage) + b'\r\n')
```


Line 86 grabs global references to our `outputFrame` and `lock`, similar to the `detect_motion` function.

Then `generate` starts an infinite loop on **Line 89** that will continue until we kill the script.

Inside the loop, we:

- First acquire the `lock` (**Line 91**).
- Ensure the `outputFrame` is not empty (**Line 94**), which may happen if a frame is dropped from the camera sensor.
- Encode the `frame` as a JPEG image on **Line 98** — JPEG compression is performed here to reduce load on the network and ensure faster transmission of frames.
- Check to see if the success `flag` has failed (**Lines 101 and 102**), implying that the JPEG compression failed and we should ignore the frame.
- Finally, serve the encoded JPEG frame as a byte array that can be consumed by a web browser.

That was quite a lot of work in a short amount of code, so definitely make sure you review this function a few times to ensure you understand how it works.

The next function, `video_feed` calls our `generate` function:

```
OpenCV - Stream video to web browser/HTML page
@app.route("/video_feed")
def video_feed():
    # return the response generated along with the specific media
    # type (mime type)
    return Response(generate(),
        mimetype = "multipart/x-mixed-replace; boundary=frame")
```

The `app.route` signature tells Flask that this function is a URL endpoint and that data is being served from `http://your_ip_address/video_feed`.

The output of `video_feed` is the live motion detection output, encoded as a byte array via the `generate` function. Your web browser is smart enough to take this byte array and display it in your browser as a live feed.

Our final code block handles parsing command line arguments and launching the Flask app:

```
OpenCV - Stream video to web browser/HTML page
115. | # check to see if this is the main thread of execution
116. | if __name__ == '__main__':
117. |     # construct the argument parser and parse command line arguments
118. |     ap = argparse.ArgumentParser()
119. |     ap.add_argument("-i", "--ip", type=str, required=True,
120. |         help="ip address of the device")
121. |     ap.add_argument("-o", "--port", type=int, required=True,
122. |         help="ephemeral port number of the server (1024 to 65535)")
123. |     ap.add_argument("-f", "--frame-count", type=int, default=32,
124. |         help="# of frames used to construct the background model")
125. |     args = vars(ap.parse_args())
126. |
127. |     # start a thread that will perform motion detection
128. |     t = threading.Thread(target=detect_motion, args=(
129. |         args["frame_count"],))
130. |     t.daemon = True
131. |     t.start()
132. |
133. |     # start the flask app
134. |     app.run(host=args["ip"], port=args["port"], debug=True,
135. |         threaded=True, use_reloader=False)
136. |
137. | # release the video stream pointer
138. | vs.stop()
```

Lines 118-125 handle parsing our command line arguments.

We need three arguments here, including:

- `--ip` : The IP address of the system you are launching the `webstream.py` file from.

- `--port` : The port number that the Flask app will run on (you'll typically supply a value of

model before motion detection is performed. By default, we use `32` frames to build the background model.

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

Lines 128-131 launch a thread that will be used to perform motion detection.

Using a thread ensures the `detect_motion` function can safely run in the background — it will be constantly running and updating our `outputFrame` so we can serve any motion detection results to our clients.

Finally, **Lines 134 and 135** launches the Flask app itself.

The HTML page structure

As we saw in `webstreaming.py`, we are rendering an HTML template named `index.html`.

The template itself is populated by the Flask web framework and then served to the web browser.

Your web browser then takes the generated HTML and renders it to your screen.

Let's inspect the contents of our `index.html` file:

```
OpenCV - Stream video to web browser/HTML page
1. <html>
2.   <head>
3.     <title>Pi Video Surveillance</title>
4.   </head>
5.   <body>
6.     <h1>Pi Video Surveillance</h1>
7.     <img src="{{ url_for('video_feed') }}">
8.   </body>
9. </html>
```

As we can see, this is super basic web page; however, pay close attention to **Line 7** — notice how we are instructing Flask to dynamically render the URL of our `video_feed` route.

Since the `video_feed` function is responsible for serving up frames from our webcam, the

video stream.

Click here to download the source code to this post

Putting the pieces together

(https://pyimagesearch.lpages.com/c/tortsem7qkvyuxc4cyfi)

Now that we've coded up our project, let's put it to the test.

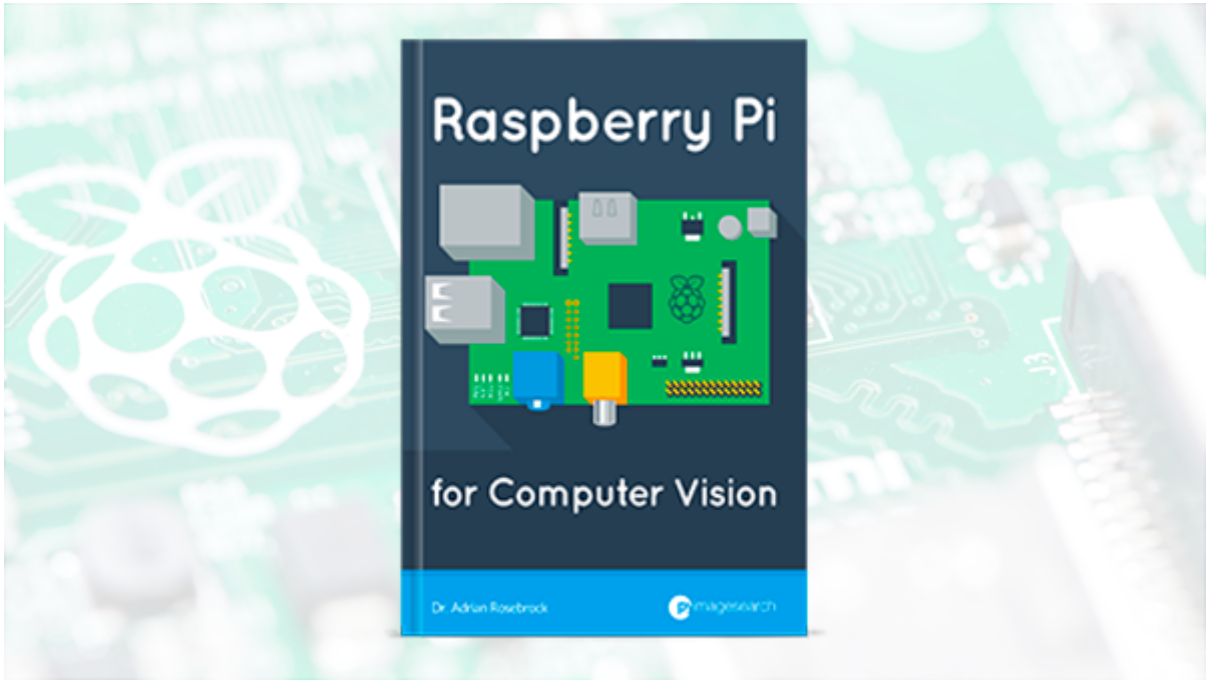
Open up a terminal and execute the following command:

```
OpenCV - Stream video to web browser/HTML page
$ python webstreaming.py --ip 0.0.0.0 --port 8000
 * Serving Flask app "webstreaming" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://0.0.0.0:8000/ (Press CTRL+C to quit)
127.0.0.1 - - [26/Aug/2019 14:43:23] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [26/Aug/2019 14:43:23] "GET /video_feed HTTP/1.1" 200 -
127.0.0.1 - - [26/Aug/2019 14:43:24] "GET /favicon.ico HTTP/1.1" 404 -
```

As you can see in the video, I opened connections to the Flask/OpenCV server from multiple

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

Join the embedded computer vision and deep learning revolution!



(https://www.pyimagesearch.com/raspberry-pi-for-computer-vision/)

I first started playing guitar twenty years ago when I was in middle school. I wasn't very good at it and I gave it up only a couple years after. Looking back, I strongly believe the reason I didn't stick with it was because I wasn't learning in a practical, hands-on manner.

Instead, my music teacher kept trying to drill theory into my head — but as an eleven year old kid, I was just trying to figure out whether I even *liked* playing guitar, let alone if I wanted to *study the theory* behind music in general.

About a year and a half ago I decided to start taking guitar lessons again. This time, I took care

The result? My finger speed is now faster than ever, my rhythm is on point, and I can annoy my wife to no end rocking *Sweet Child of Mine* on my Les Paul.

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

My point is this — whenever you are learning a new skill, whether it's computer vision, hacking with the Raspberry Pi, or even playing guitar, **one of the fastest, fool-proof methods to pick up the technique is to design (small) real-world projects around the skill and try to solve it.**

For guitar, that meant learning short riffs that not only taught me parts of actual songs but also gave me a valuable technique (such as mastering a particular pentatonic scale, for instance).

In computer vision and image processing, your goal should be to **brainstorm mini-projects** and then try to solve them. *Don't get too complicated too quickly, that's a recipe for failure.*

Instead, **grab a copy of my *Raspberry Pi for Computer Vision* (https://www.pyimagesearch.com/raspberry-pi-for-computer-vision/) book**, read it, and **use it as a launchpad for your personal projects.**

When you're done reading, go back to the chapters that inspired you the most and see how you can extend them in some manner (even *if it's just applying the **same technique** to a **different scenario***).

Solving the mini-projects you brainstorm will not only keep you interested in the subject (since you personally thought of them), but they'll teach you hands-on skills at the same time.

Today's tutorial — motion detection and streaming to a web browser — is a great starting point for such a mini-project. I hope that now that you've gone through this tutorial, you have brainstormed ideas on how you may extend this project to your own applications.

But, if you're interested in learning more...

My new book, ***Raspberry Pi for Computer Vision* (https://www.pyimagesearch.com/raspberry-pi-for-computer-vision/)**, has **over 40 projects** related to embedded computer vision + Internet of Things (IoT). You can build upon the projects in the book to solve problems around your home, business, and even for your clients. **Each of these projects have an emphasis on:**

- Learning by doing.

- Building actual, real-world projects using the Raspberry Pi

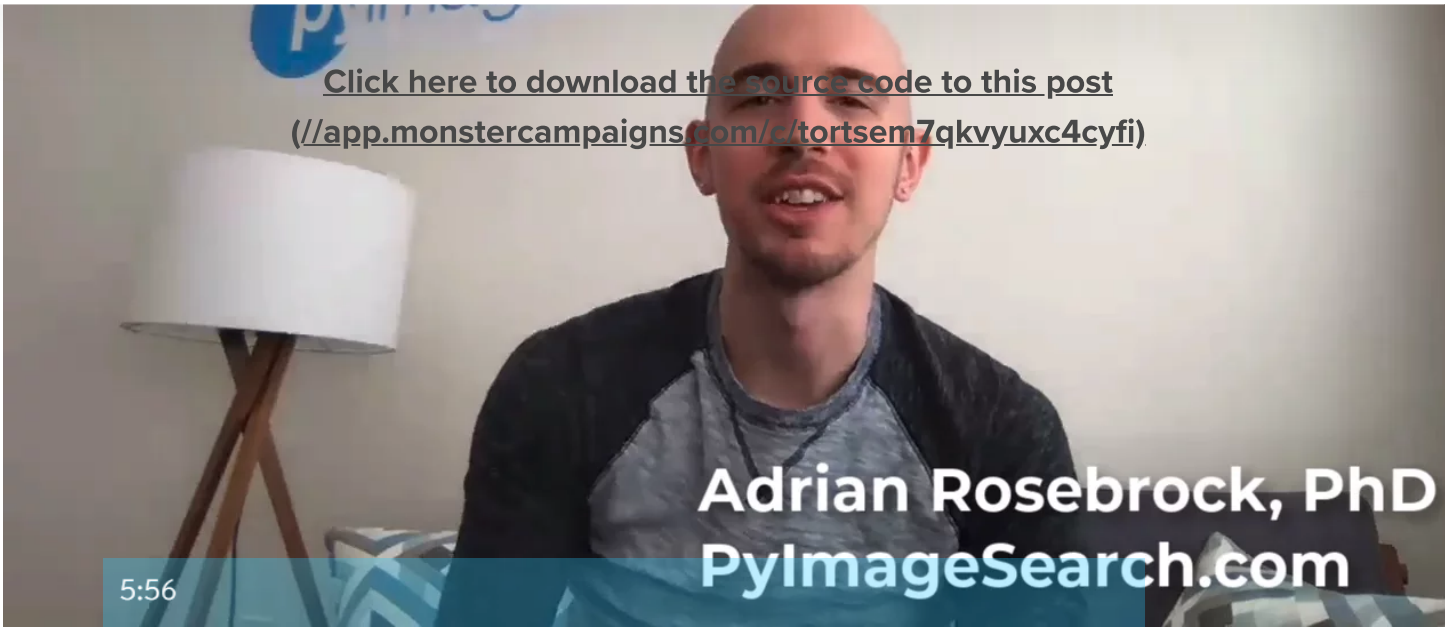
Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

A handful of the highlighted projects include:

- Daytime and nighttime wildlife monitoring
- Traffic counting and vehicle speed detection
- Deep Learning classification, object detection, and instance segmentation on resource constrained devices
- Hand gesture recognition
- Basic robot navigation
- Security applications
- Classroom attendance
- *...and many more!*

The book also covers deep learning using the **Google Coral** and **Intel Movidius NCS** coprocessors *(Hacker + Complete Bundles)*. We'll also bring in the **NVIDIA Jetson Nano** to the rescue when more deep learning horsepower is needed (*Complete Bundle*).

In case you missed the **Kickstarter (https://www.kickstarter.com/projects/adrianrosebrock/raspberry-pi-for-computer-vision-ebook)**, you may wish to watch my announcement video:



Pre-order my *Raspberry Pi for Computer Vision* book!
(https://www.pyimagesearch.com/raspberry-pi-for-computer-vision/)

Summary

In this tutorial you learned how to stream frames from a server machine to a client web browser. Using this web streaming we were able to build a basic security application to monitor a room of our house for motion.

Background subtraction is an extremely common method utilized in computer vision. Typically, these algorithms are computationally efficient, making them suitable for resource-constrained devices, such as the Raspberry Pi.

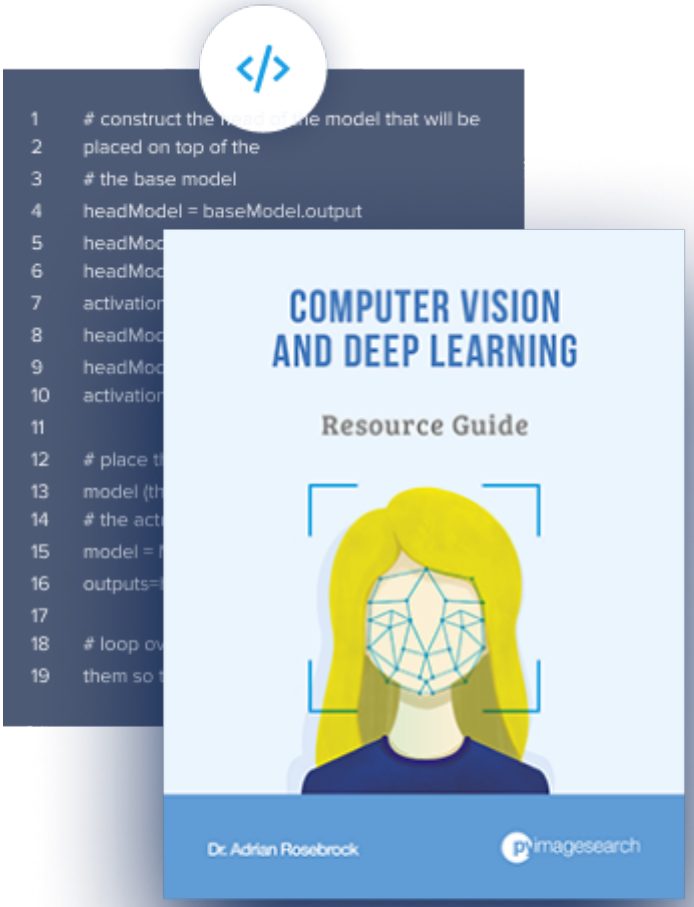
After implementing our background subtractor, we combined it with the Flask web framework, enabling us to:

1. Access frames from RPi camera module/USB webcam.
2. Apply background subtraction/motion detection to each frame.
3. Stream the results to a web page/web browser.

Whenever you need to stream frames from a device to a web browser, definitely use this code as a template/starting point.

Click here to download the source code to this post (https://app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

To download the source code to this post, and be notified when future posts are published here on PyImageSearch, *just enter your email address in the form below!*



Download the Source Code and FREE 17-page Resource

Guide on Computer Vision, OpenCV, and Deep Learning. Inside you'll find my hand-picked tutorials, books, courses, and libraries to help you master CV and DL!

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

About the Author

Hi there, I'm Adrian Rosebrock, PhD. All too often I see developers, students, and researchers wasting their time, studying the wrong things, and generally struggling to get started with Computer Vision, Deep Learning, and OpenCV. I created this website to show you what I believe is the best possible way to get your start.

Previous Article:

Building an Image Hashing Search Engine with VP-Trees and OpenCV

(https://www.pyimagesearch.com/2019/08/26/building-an-image-hashing-search-engine-

108 responses to: OpenCV – Stream video to web browser/HTML page

auraham
September 2, 2019 at 10:14 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-546413)

Great tutorial, as always! Although, sorry for your car.

Reply

Adrian Rosebrock
September 5, 2019 at 10:19 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-547102)

Thanks Auraham.

Reply

Tom
September 2, 2019 at 10:51 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-

TypeError: only integer scalar arrays can be converted to a scalar index

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

add ".tobytes()" in line 108:

yield (b'–frame\r\n' b'Content-Type: image/jpeg\r\n\r\n' + bytearray(encodedImage.tobytes()) + b'\r\n')

Reply

Adrian Rosebrock

September 5, 2019 at 10:19 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-547100)

I think you're using Python 2.7. The code is only compatible with Python 3.

Reply

Tom

September 6, 2019 at 6:01 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-547323)

3.6
great tutorial ofc, guru!

Reply

Adrian Rosebrock

September 12, 2019 at 11:26 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-550314)

just in case any other readers know what the problem may be.

Reply

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

Tom

September 16, 2019 at 8:46 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-551763)

There is no problem 😉 My comment highlighted an error (Type Error exception) that others may also face, and then I offered a solution (add ".tobytes()" in line 108) and how that addition will look like in final code.
Cheers,
Tom

Adrian Rosebrock

September 19, 2019 at 10:03 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-552805)

Thanks for the clarification Tom!

Rajat

September 2, 2019 at 10:55 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-546423)

Hello Adrian. Great tutorial once again !
My query is: Can I view the cam feed from a device which not connected to same network ?
Say if I am at my office can I view it from there ?

Reply

You would need to utilize port forwarding on your router.

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

Reply

Samjith

September 2, 2019 at 11:29 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-546429)

Hi Adrian ,
Are u using CCTV camera to record videos ? How will u connect it into raspberry ?

Reply

Samjith

September 2, 2019 at 11:58 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-546436)

Actually I'm using a analog cctv camera , which didn't have usb output .. How to convert this into usb output and connect to raspberry pi usb port ?

Reply

Tom

September 9, 2019 at 9:06 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-549329)

if it was a modern cctv (ip camera) then it's simply a case of using stream URL (similar to

Reply

Adrian Rosebrock

September 5, 2019 at 10:18 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-547099)

Sorry, I don't have any guides with CCTV.

Reply

Imam Ferianto (http://gdilab.com)

September 2, 2019 at 12:51 pm (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-546443)

This detection process is only run when someone acces the server via browser. The correct method I belive is run the detection on background process and pipe stream via ffmpeg to youtube or other rtmp, then display stream on html for public view.

Reply

Adrian Rosebrock

September 5, 2019 at 10:27 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-547112)

Actually, the detection method will run in the background *regardless* of whether or not your web browser is open ☺

Reply

September 2, 2019 at 12:54 pm (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-546444)

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

Hi Adrian,
I noticed that you used flask to post the web page, the official documentation states that flask is not suitable for production (I usually use gunicorn as a production webserver), do you think it is safe to use Flask directly?.

Reply

Adrian Rosebrock

September 5, 2019 at 10:17 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-547098)

I've addressed that question a few times in the comments already. Please refer to my replies.

Reply

Vincent PINTE DEREGNAUCOURT

September 2, 2019 at 1:05 pm (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-546447)

when the bundle will be ready (I see pre order : ok, but the book + code + ... are for sept, nov, january ?)
Thx

Reply

Chapters will start to release in September 2019. If you have pre-ordered a copy you will receive an email with a link to download the files.

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

Reply

Scott Adrian Braconnier

September 2, 2019 at 1:20 pm (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-546449)

Adrian,

So sorry to hear about your vehicle. “It’s a horrible feeling” to have your personal space violated like that.

Finding a way to share your store in a positive way is truly a gift and shows a lot about your character as a person.

Karma is all around us and it will prevail.

Reply

Adrian Rosebrock

September 5, 2019 at 10:17 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-547096)

Thanks Scott ☺

Reply

Brian (http://pyimagesearch.com)

September 2, 2019 at 2:09 pm (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-546452)

Awesome content!

Wouldn't it be great to have car-cams recording each other in parking lots…communicating via dynamic mesh networks…ultimately uploading relevant footage to owners of stolen cars…?! This avoids the problem of the dvr footage lost with the stolen vehicle, as nearby cars capture and save the action.

(My car was once stolen from a hospital parking lot while I visited a friend in the hospital. It was recovered a few weeks later.)

Reply

Adrian Rosebrock

September 5, 2019 at 10:16 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-547095)

That would be pretty neat!

Reply

Pero (http://www.helentronica.com)

September 2, 2019 at 4:59 pm (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-546481)

Thanks for the great post, Adrian, I tried it out immediately as I got your newsletter 😀

Reply

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

Adrian Rosebrock

September 5, 2019 at 10:16 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-547094)

It sounds like you have an old version of imutils installed. You should upgrade it via:

```
$ pip install --upgrade imutils
```

Reply

Rahmoun

September 2, 2019 at 5:00 pm (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-546483)

Thanks a lot Dear Adrian. Pretty good projects, my students and I love what you are doing: Thanks for sharing. My hat is off. BRAVO

Reply

Adrian Rosebrock

September 5, 2019 at 10:16 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-547093)

Thanks Rahmoun!

Reply

Soïta **Click here to download the source code to this post**

September 2, 2019 at 6:32 pm (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-546496) **(//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)**

Hi Adrian,

Another great post here !!

How about a production deployment with Flask ? This is a development one.

Not so secure.....

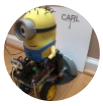
Reply

Adrian Rosebrock

September 5, 2019 at 10:16 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-547092)

This a computer vision blog. I teach CV algorithms and techniques, not web development ones. There are many books and courses on Flask, feel free to refer to them to add any other bells and whistles.

Reply



Alan McDonley

September 3, 2019 at 1:20 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-546562)

@Adrian So sorry to hear about the theft of your car, and your father – yikes.

Actually, it worked my non-aspirated RPi 3B with PiCam v1.3 to 70% load and 75 degC. I added a time.sleep(0.1) to the end of detect_motion() which dropped the load to 33% and keeps the temp at at a "cool" 60 degC.

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

Reply

Adrian Rosebrock

September 5, 2019 at 10:15 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-547091)

Thanks Alan. And congrats on getting the script to run on your RPi!

Reply

Falahgs (https://iraqprogrammer.wordpress.com/)

September 3, 2019 at 2:35 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-546579)

I am very sad about stealing your car…..you are a great person
I am a follower of all your publications..All wonderful
I hope to will you find your car as soon as possible
I wish you the best luck…
this is a great post

thanks so much for knowledge sharing

Reply

Thank you for the kind words, Fatahgs.

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

Reply

Mae

September 3, 2019 at 6:27 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-546613)

Shucks, I'm sorry to hear about your car.

Great article by the way! Easy to read and understand.

Question: How would we add user authentication to this?

Cheers!

Reply

Adrian Rosebrock

September 5, 2019 at 10:15 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-547088)

That's really up to you. Flask is a popular web framework. There are many books and courses on Flask, feel free to refer to them — those are additional bells and whistles outside of the concept taught in this post.

Reply

September 3, 2019 at 7:00 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-546621)

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

Thanks, really interesting. Did you think in recording the video to visualize it after?

Reply

Adrian Rosebrock

September 5, 2019 at 10:14 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-547087)

Yes, you can use my **KeyClipWriter. (https://www.pyimagesearch.com/2016/02/29/saving-key-event-video-clips-with-opencv/)**

Reply

Pedro

September 6, 2019 at 5:26 pm (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-547704)

Thanks!

Reply

Danial

September 4, 2019 at 1:41 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-546739)

It is another great post and I learned a lot from this.

Click here to download the source code to this post (///app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

Please tell me how to access this html page from other networks.

Reply

Adrian Rosebrock

September 5, 2019 at 10:14 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-547085)

You would need to update your router settings to perform port forwarding.

Reply

Tomal

September 4, 2019 at 7:50 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-546814)

Great work!
Really love it.

I just have a little question. Is it possible to continue run the background thread for motion detection without running (opening) any browser window. But I can see the result any time (or whenever required) in the browser.

Appreciate your feedback.

Reply

This method runs motion detection in the background thread. It *DOES NOT* require you to have a web browser open. If you want to save the result for later viewing you may want to try my Key Clip Writer:

https://www.pyimagesearch.com/2016/02/29/saving-key-event-video-clips-with-opencv/ (https://www.pyimagesearch.com/2016/02/29/saving-key-event-video-clips-with-opencv/)

Reply

Peter

September 4, 2019 at 8:14 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-546817)

Hallo. thanks for your help, its always been a life saver each time. I know you do so much but i do need a little help. Been trying to detect objects using a remote camera while streaming on the web but i havent been quit successfull. is there anything anything i should realy do to better my suituation.

Reply

Adrian Rosebrock

September 5, 2019 at 10:13 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-547083)

Have you taken a look at **Raspberry Pi for Computer Vision?**

Reply

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

Tomal

September 5, 2019 at 2:16 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-546967)

I tried it today but whenever I open the stream in browser window it just lockup my PC and I have to force power off to get it back. Seems it eats my whole resource.

By the way, it is an i3 PC with Nvidia 670/2GB and OpenCV 4.0.

Regards

Reply

Adrian Rosebrock

September 5, 2019 at 10:12 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-547082)

What operating system are you using?

Reply

Tomal

September 5, 2019 at 1:17 pm (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-547183)

Thanks for your response.

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

Tomas

September 5, 2019 at 1:19 pm (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-547185)

I forgot tell you one thing, I tried with IP cam. Not the USB one.

Reply

Adrian Rosebrock

September 12, 2019 at 11:26 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-550313)

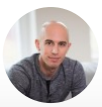
That could be it. Try using a USB webcam and see if that resolves the issue.

Carlos Urteaga

September 6, 2019 at 2:16 pm (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-547606)

Amazing page and work congrats, Sorry for your car.
I was wondering if you know how can you publish with the same flask a list of picture/description list in the right side, like array history board with previous movements

Reply



Adrian Rosebrock

September 12, 2019 at 11:25 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-550312)

As for your question, I suggest you look into some basic web development, specifically HTML, JavaScript, and CSS. That's really more of a web dev/GUI-related question than it is CV.

Otherwise, you might like this **this tutorial on saving key events. (https://www.pyimagesearch.com/2016/02/29/saving-key-event-video-clips-with-opencv/)**

Reply

Mark

September 6, 2019 at 7:29 pm (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-547781)

Hi, is it possible for a website to embed the live video feed? And how do I do that? Thanks

Reply

Adrian Rosebrock

September 12, 2019 at 11:23 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-550308)

This code already shows you how to embed the live video stream so I'm not really sure what you're asking.

Reply

Hi Adrian,

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

thank you very much for such preccious contribution and I wish that you find your car asap!!

Your code works seamlessly and I learned a lot on the way. I am trying to add the feature to turn on and off the camera controlled from index.html (client side). Could you please give a hint or show the way how it should be done?

I have tried removing “<img src" element by clicking a button but it only removes the image from client. I would like to control the stream by starting/stopping it.

Best regards,
Alihan

Reply

Adrian Rosebrock

September 12, 2019 at 11:24 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-550309)

I would suggest you look into basic web development. Learn some basic HTML and JavaScript and you’ll be able to make quick work of the project.

Reply

Muhammad Najib

September 8, 2019 at 5:41 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-548382)

Hi Adrian,

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

Adrian Rosebrock

September 12, 2019 at 11:23 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-550307)

Thanks Muhammad!

Reply

Carlos Córdoba Ruiz

September 8, 2019 at 7:57 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-548424)

Hi Adrian, nice post, i am thinking about how to consume from another computer (maybe using VLC) the streaming generated from opencv, i make some examples consuming the video from opencv video capture and the URL of flask, but didnt work, u have any idea how?
Regards

Reply

Adrian Rosebrock

September 12, 2019 at 11:22 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-550306)

Sorry, I don't have any code or tutorials for taking the output of an OpenCV script and streaming it to VLC.

Reply

Adam

September 8, 2019 at 4:06 pm (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-548744)

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

Hey Adrian, as usual – great post and sorry for your loss (been there a few years ago!!!)

Can you possibly give me a hint if you had multiple cameras in action and wanted to stream all of them into a browser?

I have followed your tutorial using message queue. I am using Flask/Socket on a central server to stream base64-encoded frames from 3 R-Pi's to clients in the browser (which update image source for each frame), but I have a feeling it's not optimal. It works like a butter from localhost (central server), but mobiles and tablets are getting a massive lag and socket.io is not catching up!

It's no worries if you have no interest in this, I understand it's an OpenCV blog. An awesome OpenCV blog 😉

Reply

Adrian Rosebrock

September 12, 2019 at 11:22 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-550305)

Are all the cameras on a single RPi? Keep in mind that you really can't use more than two cameras on a RPi, it will be far too slow.

Otherwise, I would suggest you:

1. Access each individual camera in a single Python script

That should reduce latency. Otherwise it sounds like you're trying to stream two separate sets of frames out which will certainly slow down the system.

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

Reply

Ujjawal Singh

September 11, 2019 at 2:36 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-549829)

Great post Adrian sir.
I want to take a POST request as an integer in video_feed() and then passing that integer as an argument in a utility function (generator() function in your case). Then showing the result as you did.

Reply

Adrian Rosebrock

September 12, 2019 at 11:20 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-550304)

I'm not sure I fully understand. You're saying a video is uploaded via a POST request?

Reply

Ozer

September 17, 2019 at 3:20 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-552067)

I downloaded the source code, I run the webstreaming.py and I hit **http://127.0.0.1:5000/** (http://127.0.0.1:5000/) , however on the web page I can only see Pi Video Surveillance, I could not see the camera.

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

What I am missing ?

Thank you!!!

Reply

Adrian Rosebrock

September 19, 2019 at 10:01 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-552802)

What type of camera are you using? Make sure OpenCV can access your webcam first before trying to run this script.

Reply

Alain

September 18, 2019 at 9:47 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-552464)

Hello Adrian, thank you for your very interesting tuto.

All is OK to display cam on my PC or smartphone.

I try to display on the Hololens: I get the title “Pi video surveillance”, but no Video is

Regards

Alain

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

Reply

Adrian Rosebrock

September 19, 2019 at 9:57 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-552797)

Sorry, I'm not sure. I haven't used the Hololens.

Reply

Daren Scot Wilson (http://www.darenscotwilson.com/)

September 20, 2019 at 2:32 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-553039)

My webcam stream server is running, but the video is not showing. What are the smart ways to go about diagnosing the trouble?

Reply

Ankit

September 21, 2019 at 3:56 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-553266)

(Along with link to watch a live video)

Reply

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

Adrian Rosebrock

September 25, 2019 at 10:45 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-554270)

Hey Ankit — take a look at **Raspberry Pi for Computer Vision (https://www.pyimagesearch.com/raspberry-pi-for-computer-vision/)** where that exact project is covered.

Reply

Samuel

September 21, 2019 at 11:23 pm (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-553393)

Hi I want to know the html file where video is streaming at that I need to display image which is also changing run time. How to do it?

Reply

Tayyab

September 25, 2019 at 1:07 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-554161)

can you make a tutorial how to deploy this application on Apache server.

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

Alonzo

October 14, 2019 at 2:46 pm (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-564238)

Hi Adrian,
Thanks for your inspiring work!
I have my camera working on other projects and tuts , but here I get the white screen with no video (just the headline text)
Tried to look it up with no success , Im pretty new in this and any help will be appreciated

Im on a 3b+supported logitech camera that works on other Pyimagesearch tuts

Got this error(s) :

…
File “/usr/lib/python2.7/dist-packages/werkzeug/serving.py”, line 176, in write
assert isinstance(data, bytes), ‘applications must write bytes’
AssertionError: applications must write bytes

Peace

Reply

Alonzo

October 14, 2019 at 3:01 pm (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-564240)

Solved !
It was using the python 2.7
I used “python3 python webstreaming.py –ip 0.0.0.0 –port 8000”

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

Adrian Rosebrock

October 17, 2019 at 7:09 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-564781)

Congrats on resolving the issue, Alonzo. And yes, you will need Python 3 for this project.

Reply

Patrick Alperstein

October 23, 2019 at 5:46 pm (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-566631)

Hi Adrian,

Thank you so much for sharing this amazing tutorial. We have already implemented your facial recognition project from **https://www.pyimagesearch.com/2018/06/25/raspberry-pi-face-recognition/ (https://www.pyimagesearch.com/2018/06/25/raspberry-pi-face-recognition/)**. Our next step is being able to live stream the facial recognition video on a web-browser (HTML) page. I would really appreciate it if you could give us some guidance on this project.

Thanks again,

Reply

Adrian Rosebrock

October 25, 2019 at 10:09 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-567093)

(https://www.pyimagesearch.com/raspberry-pi-for-computer-vision/). That book will help you build your project, including face recognition on the RPi and streaming to a web browser.

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

Reply

Luis

October 25, 2019 at 7:59 pm (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-567232)

Thank you for this great tutorial. I want to know if it is possible to change the parameters of the video recognizing while it is streaming. I'm trying to build an interface for calibrating object detection.

Thank you.

Reply

Ying

October 30, 2019 at 11:41 pm (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-568551)

hello，Adrian
Thanks for your amazing guide. I used this method to deal with my real video. I used YOLO V3 to do the object detection in GPU, but when i run two carema the CPU usage is so high, almost 500%, when i add time.sleep, it can reduce to 150%, it still high, do you have any suggestions to slove this issue? thank you very much

Reply

Dub

November 10, 2019 at 4:18 pm (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-571734)

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

I like the concept and the write-up / imbedded code comments are great. Fantastic tutorial and quite a bit to absorb. Lots to learn about CV and it now appears maybe some flask as well.

Command line spins up (after wrestling with quite a few library issues on stock Raspbian.) No errors on start-up and app churns along using 40-50% CPU on an RPi4. No worries. Upon the first connection from the browser things get squirley the process reports the initial connection and GET against video_feed. In the browser, I get a single image frame then the Pi hangs and crashes about 2-3 minutes later. Any background processes hang (was running top and vmstat to see if anything jumped out)

Framework seems sound though and I'll play around a bit with it. Interested in using this for some triage on automated image capture system.

Good information, well put together. Really appreciate the effort! Thanks!

Reply

Adrian Rosebrock

November 14, 2019 at 9:09 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-573205)

Thanks Dub. How large are the input frames you're trying to send over the wire? Are they large? Keep in mind that the larger the frame is, the more data that needs to be transmitted, hence the slower it will be.

Reply

haider

November 12, 2019 at 11:57 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-572442)

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

I want same functionality in django. I am trying but I haven't found any solution. can you help me in this

Reply

Adrian Rosebrock

November 14, 2019 at 9:09 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-573204)

Sorry, I do not have the same good on Django. Good luck!

Reply

Imran B (http://Splendio.com)

November 13, 2019 at 7:31 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-572720)

Hi can you help me with licence plate detection and recognition in live stream

Reply

Adrian Rosebrock

November 14, 2019 at 9:08 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-573203)

automatic license plate recognition. I suggest you start there.

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

Reply

Marcel

November 19, 2019 at 8:28 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-575175)

Hey Adrian, thanks a lot for your tutorial!
Would you happen to know if its possible to stream the output frames as H.264 instead of as image?

Reply

Shahar

November 27, 2019 at 7:28 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-585345)

Hi Adrian,

Thanks for this great tutorial!
Very much appreciate sharing your work and knowledge.

BR,
Shahar

Reply

Thanks Shahar! 😉 **Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)**

Reply

Chris

November 27, 2019 at 3:30 pm (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-585604)

Really outstanding article. It was exactly what I needed. Thanks!

Reply

Adrian Rosebrock

December 5, 2019 at 10:20 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-589852)

Thanks Chris, I'm glad you found it helpful!

Reply

philnc

December 3, 2019 at 12:36 pm (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-588701)

Now I just have to figure out how to make this work with an rtsp stream!

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

Adrian Rosebrock

December 5, 2019 at 10:19 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-589851)

Thanks, I'm glad you enjoyed it!

Reply

Emmanuel KOUPOH

December 10, 2019 at 6:20 pm (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-591197)

hi, please can you show me do get video of webcam in django app .

Reply

Adrian Rosebrock

December 12, 2019 at 9:49 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-591577)

Sorry, I don't have any Django code available. I would suggest referring to the Django documentation. Good luck!

Reply

Hi Adrian, **Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)**
Fantastic job with cv educative posts! they are ever-increasing in quality.

I tried your implementation (out of the box) in a mac, python 3.6 Opencv 4.1.2

The code runs successfully but when opening in a browser: 0.0.0.0:8000 suddenly everything crashes giving the following error: Bus error: 10

The http webpage displays the title and no image.

Any idea of what is happening?

Reply

Adrian Rosebrock

December 18, 2019 at 9:28 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-598307)

Oh no, I'm sorry to hear about the error. Can you try inserting some "print" statements or use "pdb" to determine exactly which line of code is causing the error?

Reply

Tian Zhiwei

December 31, 2019 at 3:16 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-621332)

Dear Rosebrock

process ,the droplet stream can not visible,so I am confusing and don't know how to implement it.i read a lot of your wonderful blogs and got to know you are professional on machine vision and learning,may I have your advice or some examples on my problem,if video and image are needed I will send them to you,thank you very much for your time.

Reply

carlos casado casero (http://...)

January 29, 2020 at 9:19 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-686970)

I have the same problem, but I can't obtain a solution... any help me?

Reply

alch

January 7, 2020 at 11:36 pm (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-634343)

Hi Adrian, I was wondering how to recognize faces during video stream and how to store datas with their time and dates ?
Can you do tutorial about this topic?

Reply

Adrian Rosebrock

January 16, 2020 at 10:46 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-656610)

Reply

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

Farzana Anjum

January 23, 2020 at 1:03 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-671047)

Hey Adrian,

Thank you for this great tutorial!

Its working me on system and i want to access this same live stream on Android mobile.

Can i access the same live stream on Android ?

Reply

Adrian Rosebrock

January 23, 2020 at 9:13 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-671581)

I can confirm it works on iOS but I have not tried in Android. I suggest you try and see.

Reply

Kevin

January 27, 2020 at 5:11 pm (https://www.pyimagesearch.com/2019/09/02/opencv-stream

Hi Adrian!

Thanks for your work with OpenCV, you've really helped a beginner in myself get started with applications of OpenCV.

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

I was running the source code in this tutorial and I came across an "index 0" error – the code was not detecting my Pi camera module running.

I've enabled the camera in my Pi's configuration settings, and have ran some command line statements to enable the camera module. Did you have this issue before or could you provide some resources for further assistance? Many thanks for your help. -Kevin

Reply

Adrian Rosebrock

January 30, 2020 at 8:47 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-689013)

You're asking a RPi camera module? If so, uncomment Line 25 and comment out Line 26.

Reply

Anurag

February 7, 2020 at 2:00 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-709357)

Thank you for this tutorial .
Love ,Anurag

Reply

Adrian Rosebrock

February 13, 2020 at 11:14 am (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-722628)

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

You are welcome!

Reply

yanti (http://-)

February 11, 2020 at 11:37 pm (https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/#comment-721860)

Hello, Adrian

Thank you for sending me OpenCV stream source code.
Your code works well on Chrome but not working on Microsoft Edge (I see no video, only blank screen).
Flask using multipart/x-mixed-replace not working on Microsoft Edge.
I got the video showing on Microsoft Edge by not using Flask for streaming but using socketserver and http (this works on Chrome too).

Reply

Before you leave a comment...

Hey, Adrian here, author of the PyImageSearch blog. I'd love to hear from you, but

- **If you have a question, read the comments first.** You should also search this page (i.e., ctrl + f) for keywords related to your question. It's likely that I have already addressed your question in the comments.

Click here to download the source code to this post (https://app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

- **If you are copying and pasting code/terminal output, please don't.** Reviewing another programmers' code is a very time consuming and tedious task, and due to the volume of emails and contact requests I receive, I simply cannot do it.

- **Be respectful of the space.** I put a lot of my own personal time into creating these free weekly tutorials. On average, each tutorial takes me 15-20 hours to put together. I love offering these guides to you and I take pride in the content I create. Therefore, I will not approve comments that include large code blocks/terminal output as it destroys the formatting of the page. Kindly be respectful of this space.

- **Be patient**. I receive 200+ comments and emails per day. Due to spam, and my desire to personally answer as many questions as I can, I hand moderate all new comments (typically once per week). I try to answer as many questions as I can, but I'm only one person. Please don't be offended if I cannot get to your question

- **Do you need priority support? Consider purchasing one of my books and courses**. I place customer questions and emails in a separate, special priority queue and answer them first. **If you are a customer of mine you will receive a guaranteed response from me.** If there's any time left over, I focus on the community at large and attempt to answer as many of those questions as I possibly can.

Thank you for keeping these guidelines in mind before submitting your comment.

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

Similar articles

DLIB FACE APPLICATIONS FACIAL LANDMARKS

(Faster) Facial landmark detector with dlib

April 2, 2018

(https://www.pyimagesearch.com/2018/04/02/faster-facial-landmark-detector-with-dlib/)

IMAGE SEARCH ENGINE BASICS

Building an Image Search Engine: Defining Your Image Descriptor (Step 1 of 4)

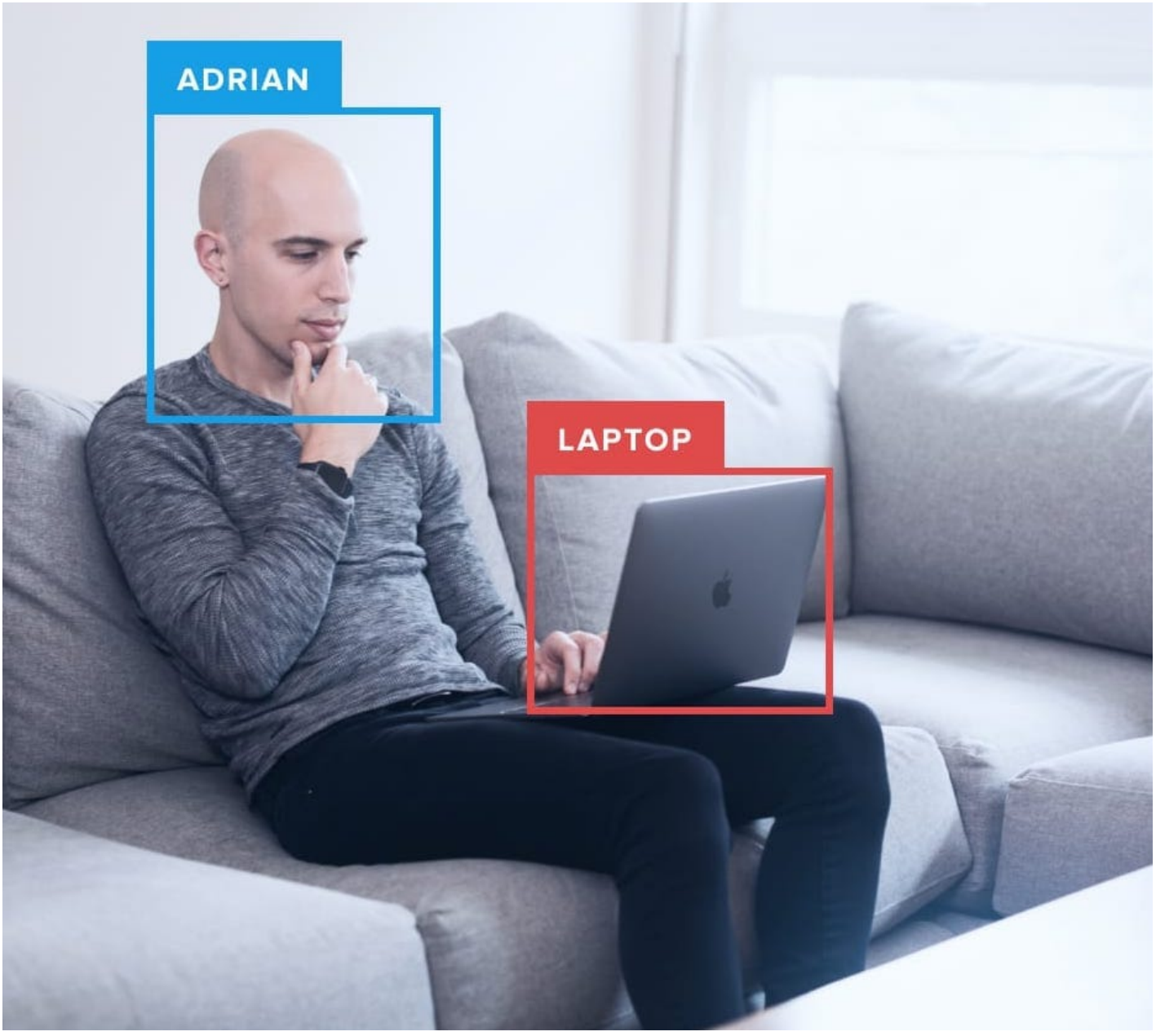
February 3, 2014
(https://www.pyimagesearch.com/2014/02/03/building-an-image-search-engine-defining-your-image-descriptor-step-1-of-4/)

LIBRARIES TUTORIALS

Increasing webcam FPS with Python and OpenCV

December 21, 2015

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)



You can learn Computer Vision, Deep Learning, and

(https://www.pyimagesearch.com/free-opencv-computer-vision-deep-learning-crash-course/)

Practical Python and OpenCV (https://www.pyimagesearch.com/practical-python-opencv/)

Deep Learning for Computer Vision with Python (https://www.pyimagesearch.com/deep-learning-computer-vision-python-book/)

PyImageSearch Gurus Course (https://www.pyimagesearch.com/pyimagesearch-gurus/)

Raspberry Pi for Computer Vision (https://www.pyimagesearch.com/raspberry-pi-for-computer-vision/)

OpenCV Install Guides (https://www.pyimagesearch.com/opencv-tutorials-resources-guides/)

Click here to download the source code to this post (https://app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

About (https://www.pyimagesearch.com/about/)

FAQ (https://www.pyimagesearch.com/faqs/)

Blog (https://www.pyimagesearch.com/topics/)

Contact (https://www.pyimagesearch.com/contact/)

Privacy Policy (https://www.pyimagesearch.com/privacy-policy/)

(https://www.facebook.com/pyimagesearch) (https://twitter.com/PyImageSearch) (http://www.linkedin.com/pub/adrian-rosebrock/2a/873/59b) (https://www.youtube.com/channel/UCoQK7OVclVy-nV4m-SMCk_Q/videos)