**OpenCV People Counter:**

In this tutorial you will learn how to build a “people counter” with OpenCV and Python. Using OpenCV, we’ll count the number of people who are heading “in” or “out” of a department store in real-time. Building a person counter with OpenCV has been one of the most-requested topics here on the PyImageSearch and I’ve been meaning to do a blog post on people counting for a year.

From there I’ll provide a brief discussion on the difference between object detection and object tracking, along with how we can leverage both to create a more accurate people counter.

Afterwards, we’ll review the directory structure for the project and then implement the entire person counting project.

Finally, we’ll examine the results of applying people counting with OpenCV to actual videos.

### Required Python libraries for people counting

In order to build our people counting applications, we’ll need a number of different Python libraries, including:

* [NumPy](http://www.numpy.org/)
* [OpenCV](https://pyimagesearch.com/opencv-tutorials-resources-guides/)
* [dlib](http://dlib.net/)
* [imutils](https://github.com/jrosebr1/imutils)

### Understanding object detection vs. object tracking

There is a fundamental difference between object detection and object tracking that you must understand before we proceed with the rest of this tutorial.

When we apply object detection we are determining where in an image/frame an object is. An object detector is also typically more computationally expensive, and therefore slower, than an object tracking algorithm. Examples of object detection algorithms include Haar cascades, HOG + Linear SVM, and deep learning-based object detectors such as Faster R-CNNs, YOLO, and Single Shot Detectors (SSDs).

An object tracker, on the other hand, will accept the input (x, y)-coordinates of where an object is in an image and will:

1. Assign a unique ID to that particular object
2. Track the object as it moves around a video stream, predicting the new object location in the next frame based on various attributes of the frame (gradient, optical flow, etc.)

Examples of object tracking algorithms include MedianFlow, MOSSE, GOTURN, kernalized correlation filters, and discriminative correlation filters, to name a few.

If you’re interested in learning more about the object tracking algorithms built into OpenCV, be sure to [refer to this blog post](https://pyimagesearch.com/2018/07/30/opencv-object-tracking/).

### Combining both object detection and object tracking

Highly accurate object trackers will combine the concept of object detection and object tracking into a single algorithm, typically divided into two phases:

* **Phase 1 — Detecting:** During the detection phase we are running our computationally more expensive object tracker to (1) detect if new objects have entered our view, and (2) see if we can find objects that were “lost” during the tracking phase. For each detected object we create or update an object tracker with the new bounding box coordinates. Since our object detector is more computationally expensive we only run this phase once every N frames.
* **Phase 2 — Tracking:** When we are not in the “detecting” phase we are in the “tracking” phase. For each of our detected objects, we create an object tracker to track the object as it moves around the frame. Our object tracker should be faster and more efficient than the object detector. We’ll continue tracking until we’ve reached the N-th frame and then re-run our object detector. The entire process then repeats.

**The benefit of this hybrid approach is that we can apply highly accurate object detection methods without as much of the computational burden.** We will be implementing such a tracking system to build our people counter.