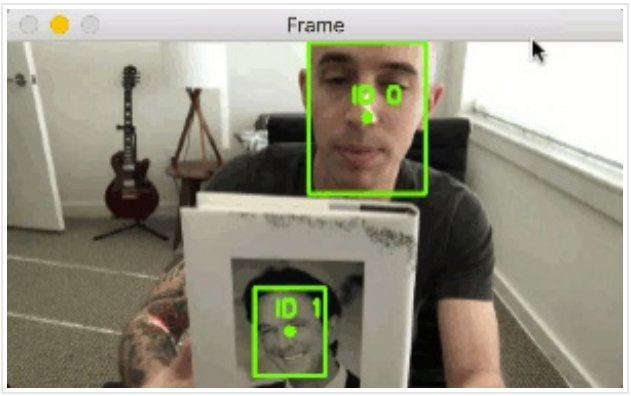
pyimagesearch
be awesome at building image search engines

Simple object tracking with OpenCV

by Adrian Rosebrock on July 23, 2018 in Object Tracking, Tutorials

Click here to download the source code to this post



Today's tutorial kicks off a new series of blog posts on object tracking, arguably one of the most requested topics here on PyImageSearch.

Object tracking is the process of:

1. Taking an initial set of object detections (such as an input set of bounding box coordinates)
2. Creating a unique ID for each of the initial detections
3. And then tracking each of the objects as they move around frames in a video, maintaining the assignment of unique IDs

Furthermore, object tracking allows us to **apply a unique ID to each tracked object**, making it possible for us to count unique objects in a video. Object tracking is paramount to building a **person counter** (which we'll do later in this series).

An ideal object tracking algorithm will:

- Only require the object detection phase once (i.e., when the object is initially detected)
- Will be extremely fast — *much* faster than running the actual object detector itself
- Be able to handle when the tracked object "disappears" or moves outside the boundaries of the video frame
- Be robust to occlusion
- Be able to pick up objects it has "lost" in between frames

This is a tall order for any computer vision or image processing algorithm and there are a variety of tricks we can play to help improve our object trackers.

But before we can build such a robust method we first need to study the fundamentals of object tracking.

In today's blog post, you will learn how to implement**centroid tracking with OpenCV**, an easy to understand, yet highly effective tracking algorithm.

In future posts in this object tracking series, I'll start going into more advanced kernel-based and correlation-based tracking algorithms.

To learn how to get started building your first object tracking with OpenCV,*just keep reading!*

Looking for the source code to this post?
Jump right to the downloads section.

Simple object tracking with OpenCV

In the remainder of this post, we'll be implementing a simple object tracking algorithm using the OpenCV library.

This object tracking algorithm is called *centroid tracking* as it relies on the Euclidean distance between (1) *existing* object centroids (i.e., objects the centroid tracker has already seen before) and (2) new object centroids between subsequent frames in a video.

We'll review the centroid algorithm in more depth in the following section. From there we'll implement a Python class to contain our centroid tracking algorithm and then create a Python script to actually run the object tracker and apply it to input videos.

Finally, we'll run our object tracker and examine the results, noting both the positives and the drawbacks of the algorithm.

The centroid tracking algorithm

The centroid tracking algorithm is a multi-step process. We will review each of the tracking steps in this section.

Step #1: Accept bounding box coordinates and compute centroids

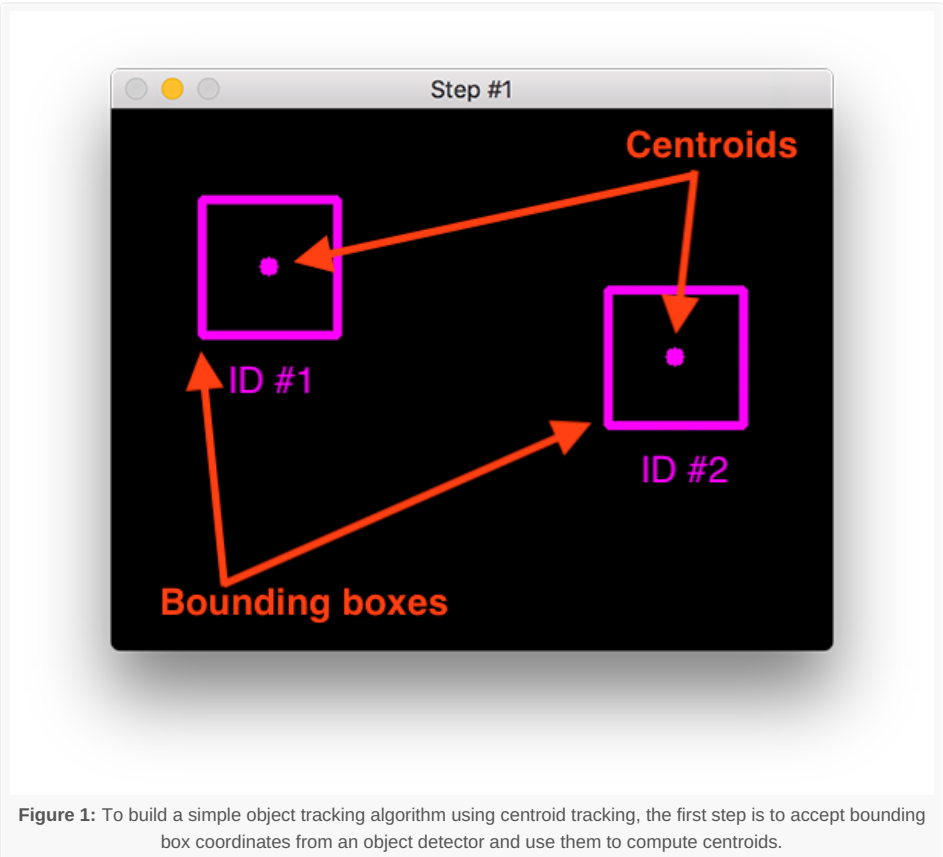


Figure 1: To build a simple object tracking algorithm using centroid tracking, the first step is to accept bounding box coordinates from an object detector and use them to compute centroids.

The centroid tracking algorithm assumes that we are passing in a set of bounding box *(x, y)*-coordinates for each detected object in ***every single frame***.

These bounding boxes can be produced by any type of object detector you would like (color thresholding + contour extraction, Haar cascades, HOG + Linear SVM, SSDs, Faster R-CNNs, etc.), provided that they are computed for every frame in the video.

Once we have the bounding box coordinates we must compute the "centroid", or more simply,*the center (x, y)-coordinates* of the bounding box. **Figure 1** above demonstrates accepting a set of bounding box coordinates and computing the centroid.

Since these are the first initial set of bounding boxes presented to our algorithm we will assign them unique IDs.

Step #2: Compute Euclidean distance between new bounding boxes and existing objects

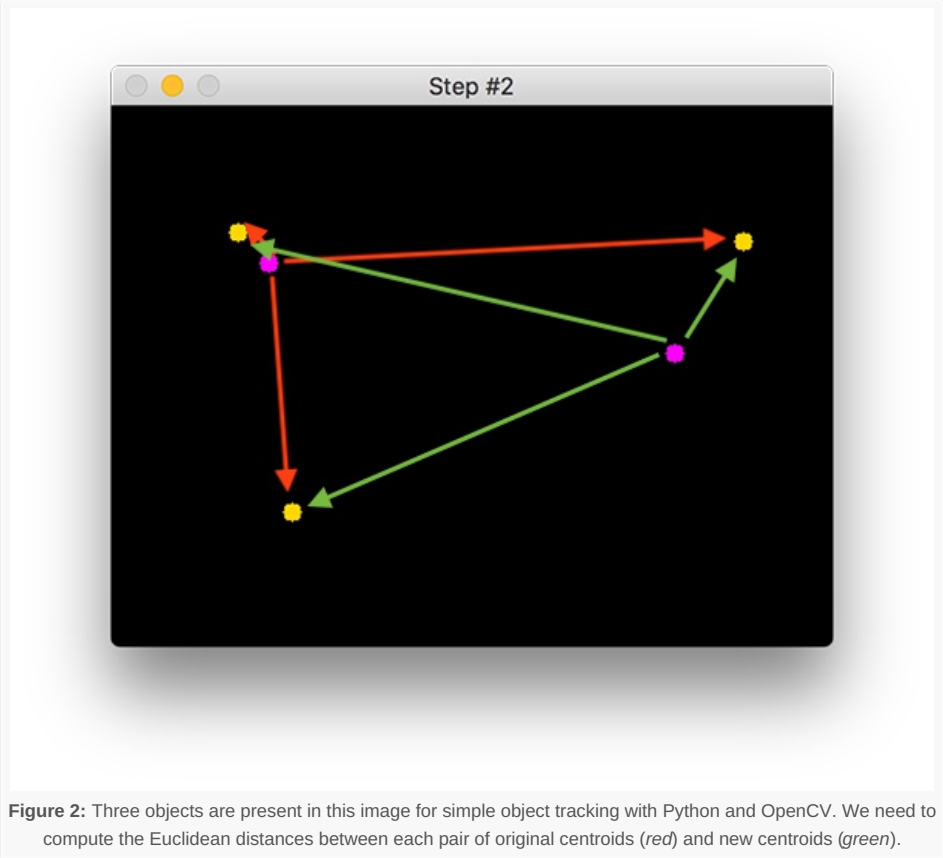


Figure 2: Three objects are present in this image for simple object tracking with Python and OpenCV. We need to compute the Euclidean distances between each pair of original centroids (*red*) and new centroids (*green*).

For every subsequent frame in our video stream we apply **Step #1** of computing object centroids; however, instead of assigning a new unique ID to each detected object (which would defeat the purpose of object tracking), we first need to determine if we can *associate* the *new* object centroids (yellow) with the *old* object centroids (purple). To accomplish this process, we compute the Euclidean distance (highlighted with green arrows) between each pair of existing object centroids and input object centroids.

From **Figure 2** you can see that we have this time detected three objects in our image. The two pairs that are close together are two existing objects.

We then compute the Euclidean distances between each pair of original centroids (yellow) and new centroids (purple). But how do we use the Euclidean distances between these points to actually match them and associate them?

The answer is in **Step #3**.

Step #3: Update *(x, y)*-coordinates of existing objects

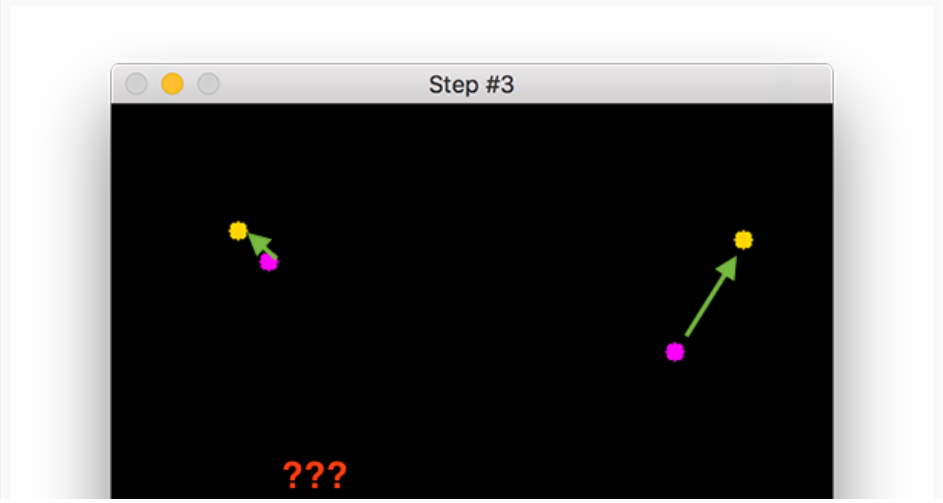


Figure 3: Our simple centroid object tracking method has associated objects with minimized object distances. What do we do about the object in the bottom left though?

The primary assumption of the centroid tracking algorithm is that a given object will potentially move in between subsequent frames, but the *distance* between the centroids for frames F_t and F_{t+1} will be *smaller* than all other distances between objects.

Therefore, if we choose to associate centroids with minimum distances between subsequent frames we can build our object tracker.

In **Figure 3** you can see how our centroid tracker algorithm chooses to associate centroids that minimize their respective Euclidean distances.

But what about the lonely point in the bottom-left?

It didn't get associated with anything — what do we do with it?

Step #4: Register new objects

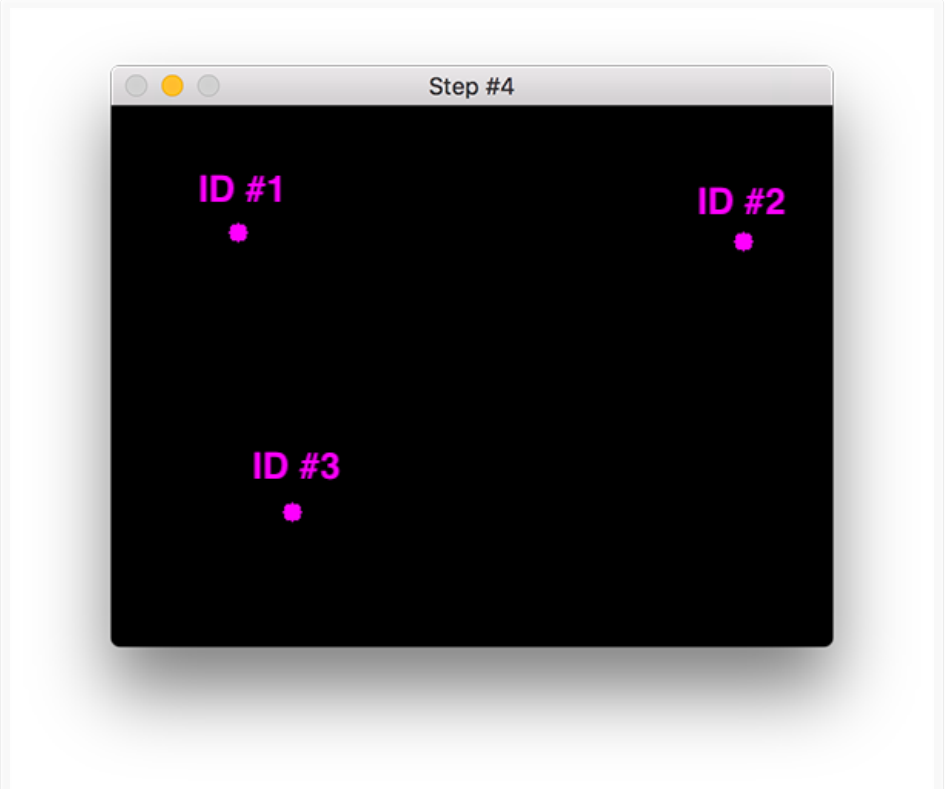


Figure 4: In our object tracking with Python and OpenCV example, we have a new object that wasn't matched with an existing object, so it is registered as object ID #3.

In the event that there are more input detections than existing objects being tracked, we need to register the new object. "Registering" simply means that we are adding the new object to our list of tracked objects by:

1. Assigning it a new object ID
2. Storing the centroid of the bounding box coordinates for that object

We can then go back to **Step #2** and repeat the pipeline of steps for every frame in our video stream.

Figure 4 demonstrates the process of using the minimum Euclidean distances to associate existing object IDs and then registering a new object.

Step #5: Deregister old objects

Any reasonable object tracking algorithm needs to be able to handle when an object has been lost, disappeared, or left the field of view.

Exactly how you handle these situations is really dependent on where your object tracker is meant to be deployed, but for this implementation, we will deregister old objects when they cannot be matched to any existing objects for a total of *N* subsequent frames.

Object tracking project structure

To see today's project structure in your terminal, simply use the `tree` command:

```
Simple object tracking with OpenCV
$ tree --dirsfirst
.
├── pyimagesearch
│   ├── __init__.py
│   └── centroidtracker.py
├── object_tracker.py
├── deploy.prototxt
└── res10_300x300_ssd_iter_140000.caffemodel

1 directory, 5 files
```

Our `pyimagesearch` module is not pip-installable — it is included with today's ***"Downloads"*** (which you'll find at the bottom of this post). Inside you'll find the `centroidtracker.py` file which contains the `CentroidTracker` class.

The `CentroidTracker` class is an important component used in the `object_tracker.py` driver script.

The remaining `.prototxt` and `.caffemodel` files are part of the OpenCV deep learning face detector. They are necessary for today's face detection + tracking method, but you could easily use another form of detection (more on that later).

Be sure that you have NumPy, SciPy, and imutils installed before you proceed:

```
Simple object tracking with OpenCV
$ pip install numpy scipy imutils
```

...in addition to having OpenCV 3.3+ installed. If you follow one of my OpenCV install tutorials, be sure to replace the tail end of the `wget` command to grab at least OpenCV 3.3 (and update the paths in the CMake command). You'll need 3.3+ to ensure you have the DNN module.

Implementing centroid tracking with OpenCV

Before we can apply object tracking to our input video streams, we first need to implement the centroid tracking algorithm. While you're digesting this centroid tracker script, just keep in mind **Steps 1-5** above and review the steps as necessary.

As you'll see, the translation of steps to code requires quite a bit of thought, and while we perform all steps, they aren't linear due to the nature of our various data structures and code constructs.

I would suggest

1. Reading the steps above
2. Reading the code explanation for the centroid tracker
3. And finally reading the steps above once more

This process will bring everything full circle and allow you to wrap your head around the algorithm.

Once you're sure you understand the steps in the centroid tracking algorithm, open up the `centroidtracker.py` inside the `pyimagesearch` module and let's review the code:

```
Simple object tracking with OpenCV
# import the necessary packages
from scipy.spatial import distance as dist
from collections import OrderedDict
import numpy as np

class CentroidTracker():
 def __init__(self, maxDisappeared=50):
  # initialize the next unique object ID along with two ordered
  # dictionaries used to keep track of mapping a given object
  # ID to its centroid and number of consecutive frames it has
  # been marked as "disappeared", respectively
  self.nextObjectID = 0
  self.objects = OrderedDict()
  self.disappeared = OrderedDict()
```

```
15
16   # store the number of maximum consecutive frames a given
17   # object is allowed to be marked as "disappeared" until we
18   # need to deregister the object from tracking
19   self.maxDisappeared = maxDisappeared
```

On **Lines 2-4** we import our required packages and modules — `distance` , `OrderedDict` , and `numpy` .

Our `CentroidTracker` class is defined on **Line 6**. The constructor accepts a single parameter, the maximum number of consecutive frames a given object has to be lost/disappeared for until we remove it from our tracker (**Line 7**).

Our constructor builds four class variables:

- `nextObjectID` : A counter used to assign unique IDs to each object (**Line 12**). In the case that an object leaves the frame and does not come back for `maxDisappeared` frames, a new (next) object ID would be assigned.
- `objects` : A dictionary that utilizes the object ID as the key and the centroid *(x, y)*-coordinates as the value (**Line 13**).
- `disappeared` : Maintains number of consecutive frames (value) a particular object ID (key) has been marked as "lost"for (**Line 14**).
- `maxDisappeared` : The number of consecutive frames an object is allowed to be marked as "lost/disappeared" until we deregister the object.

Let's define the `register` method which is responsible for adding new objects to our tracker:

Simple object tracking with OpenCV — Python

```
21  def register(self, centroid):
22   # when registering an object we use the next available object
23   # ID to store the centroid
24   self.objects[self.nextObjectID] = centroid
25   self.disappeared[self.nextObjectID] = 0
26   self.nextObjectID += 1
```

The `register` method is defined on **Line 21**. It accepts a `centroid` and then adds it to the `objects` dictionary using the next available object ID.

The number of times an object has disappeared is initialized to `0` in the `disappeared` dictionary (**Line 25**).

Finally, we increment the `nextObjectID` so that if a new object comes into view, it will be associated with a unique ID (**Line 26**).

Similar to our `register` method, we also need a `deregister` method:

Simple object tracking with OpenCV — Python

```
28  def deregister(self, objectID):
29   # to deregister an object ID we delete the object ID from
30   # both of our respective dictionaries
31   del self.objects[objectID]
32   del self.disappeared[objectID]
```

Just like we can add new objects to our tracker, we also need the ability to remove old ones that have been lost or disappeared from our the input frames themselves.

The `deregister` method is defined on **Line 28**. It simply deletes the `objectID` in both the `objects` and `disappeared` dictionaries, respectively (**Lines 31 and 32**).

The heart of our centroid tracker implementation lives inside the `update` method:

Simple object tracking with OpenCV — Python

```
34  def update(self, rects):
35   # check to see if the list of input bounding box rectangles
36   # is empty
37   if len(rects) == 0:
38    # loop over any existing tracked objects and mark them
39    # as disappeared
40    for objectID in list(self.disappeared.keys()):
41     self.disappeared[objectID] += 1
42
43    # if we have reached a maximum number of consecutive
44    # frames where a given object has been marked as
45    # missing, deregister it
46    if self.disappeared[objectID] > self.maxDisappeared:
47     self.deregister(objectID)
48
49   # return early as there are no centroids or tracking info
50   # to update
51   return self.objects
```

The update method, defined on **Line 34**, accepts a list of bounding box rectangles, presumably from an object detector (Haar cascade, HOG + Linear SVM, SSD, Faster R-CNN, etc.). The format of the `rects` parameter is assumed to be a tuple with this

structure: `(startX, startY, endX, endY)`.

If there are no detections, we'll loop over all object IDs and increment their `disappeared` count (**Lines 37-41**). We'll also check if we have reached the maximum number of consecutive frames a given object has been marked as missing. If that is the case we need to remove it from our tracking systems (**Lines 46 and 47**). Since there is no tracking info to update, we go ahead and `return` early on **Line 51**.

Otherwise, we have quite a bit of work to do over the next seven code blocks in the `update` method:

```
Simple object tracking with OpenCV                                    Python
53    # initialize an array of input centroids for the current frame
54    inputCentroids = np.zeros((len(rects), 2), dtype="int")
55
56    # loop over the bounding box rectangles
57    for (i, (startX, startY, endX, endY)) in enumerate(rects):
58     # use the bounding box coordinates to derive the centroid
59     cX = int((startX + endX) / 2.0)
60     cY = int((startY + endY) / 2.0)
61     inputCentroids[i] = (cX, cY)
```

On **Line 54** we'll initialize a NumPy array to store the centroids for each `rect`.

Then, we loop over bounding box rectangles (**Line 57**) and compute the centroid and store it in the `inputCentroids` list (**Lines 59-61**).

If there are currently no objects we are tracking, we'll register each of the new objects:

```
Simple object tracking with OpenCV                                    Python
63    # if we are currently not tracking any objects take the input
64    # centroids and register each of them
65    if len(self.objects) == 0:
66     for i in range(0, len(inputCentroids)):
67      self.register(inputCentroids[i])
```

Otherwise, we need to update any existing object *(x, y)*-coordinates based on the centroid location that minimizes the Euclidean distance between them:

```
Simple object tracking with OpenCV                                    Python
69    # otherwise, are are currently tracking objects so we need to
70    # try to match the input centroids to existing object
71    # centroids
72    else:
73     # grab the set of object IDs and corresponding centroids
74     objectIDs = list(self.objects.keys())
75     objectCentroids = list(self.objects.values())
76
77     # compute the distance between each pair of object
78     # centroids and input centroids, respectively -- our
79     # goal will be to match an input centroid to an existing
80     # object centroid
81     D = dist.cdist(np.array(objectCentroids), inputCentroids)
82
83     # in order to perform this matching we must (1) find the
84     # smallest value in each row and then (2) sort the row
85     # indexes based on their minimum values so that the row
86     # with the smallest value is at the *front* of the index
87     # list
88     rows = D.min(axis=1).argsort()
89
90     # next, we perform a similar process on the columns by
91     # finding the smallest value in each column and then
92     # sorting using the previously computed row index list
93     cols = D.argmin(axis=1)[rows]
```

The updates to existing tracked objects take place beginning at the `else` on **Line 72**. The goal is to track the objects and to maintain correct object IDs — this process is accomplished by computing the Euclidean distances between all pairs of `objectCentroids` and `inputCentroids`, followed by associating object IDs that minimize the Euclidean distance.

Inside of the else block beginning on **Line 72**, we will:

- Grab `objectIDs` and `objectCentroid` values (**Lines 74 and 75**).
- Compute the distance between each pair of existing object centroids and new input centroids (**Line 81**). The output NumPy array shape of our distance map `D` will be `(# of object centroids, # of input centroids)`.
- To perform the matching we must (1) Find the smallest value in each row, and (2) Sort the row indexes based on the minimum values (**Line 88**). We perform a very similar process on the columns, finding the smallest value in each column, and then sorting them based on the ordered rows (**Line 93**). Our goal is to have the index values with the smallest corresponding distance at the front of the lists.

The next step is to use the distances to see if we can associate object IDs:

```
Simple object tracking with OpenCV                                    Python
 95     # in order to determine if we need to update, register,
 96     # or deregister an object we need to keep track of which
 97     # of the rows and column indexes we have already examined
 98     usedRows = set()
 99     usedCols = set()
100
101     # loop over the combination of the (row, column) index
102     # tuples
103     for (row, col) in zip(rows, cols):
104      # if we have already examined either the row or
105      # column value before, ignore it
106      # val
107      if row in usedRows or col in usedCols:
108       continue
109
110      # otherwise, grab the object ID for the current row,
111      # set its new centroid, and reset the disappeared
112      # counter
113      objectID = objectIDs[row]
114      self.objects[objectID] = inputCentroids[col]
115      self.disappeared[objectID] = 0
116
117      # indicate that we have examined each of the row and
118      # column indexes, respectively
119      usedRows.add(row)
120      usedCols.add(col)
```

Inside the code block above, we:

- Initialize two sets to determine which row and column indexes we have already used (**Lines 98 and 99**). Keep in mind that a set is similar to a list but it contains only *unique* values.
- Then we loop over the combinations of `(row, col)` index tuples (**Line 103**) in order to update our object centroids:
  - If we've already used either this row or column index, ignore it and `continue` to loop (**Lines 107 and 108**).
  - Otherwise, we have found an input centroid that:
    - 1. Has the smallest Euclidean distance to an existing centroid
    - 2. And has not been matched with any other object
    - In that case, we update the object centroid (**Lines 113-115**) and make sure to add the `row` and `col` to their respective `usedRows` and `usedCols` sets

There are likely indexes in our `usedRows` + `usedCols` sets that we have NOT examined yet:

```
Simple object tracking with OpenCV                                    Python
122     # compute both the row and column index we have NOT yet
123     # examined
124     unusedRows = set(range(0, D.shape[0])).difference(usedRows)
125     unusedCols = set(range(0, D.shape[1])).difference(usedCols)
```

So we must determine which centroid indexes we haven't examined yet and store them in two new convenient sets (`unusedRows` and `unusedCols`) on **Lines 124 and 125**.

Our final check handles any objects that have become lost or if they've potentially disappeared:

```
Simple object tracking with OpenCV                                    Python
127     # in the event that the number of object centroids is
128     # equal or greater than the number of input centroids
129     # we need to check and see if some of these objects have
130     # potentially disappeared
131     if D.shape[0] >= D.shape[1]:
132      # loop over the unused row indexes
133      for row in unusedRows:
134       # grab the object ID for the corresponding row
135       # index and increment the disappeared counter
136       objectID = objectIDs[row]
137       self.disappeared[objectID] += 1
138
139       # check to see if the number of consecutive
140       # frames the object has been marked "disappeared"
141       # for warrants deregistering the object
142       if self.disappeared[objectID] > self.maxDisappeared:
143        self.deregister(objectID)
```

To finish up:

- If the number of object centroids is greater than or equal to the number of input centroids (**Line 131**):
  - We need to verify if any of these objects are lost or have disappeared by looping over unused row indexes if any (**Line 133**).
  - In the loop, we will:
    - 1. Increment their `disappeared` count in the dictionary (**Line 137**).

- 2. Check if the `disappeared` count exceeds the `maxDisappeared` threshold (**Line 142**), and, if so we'll deregister the object (**Line 143**).

Otherwise, the number of input centroids is greater than the number of existing object centroids, so we have new objects to register and track:

```
Simple object tracking with OpenCV                                    Python
145     # otherwise, if the number of input centroids is greater
146     # than the number of existing object centroids we need to
147     # register each new input centroid as a trackable object
148     else:
149      for col in unusedCols:
150       self.register(inputCentroids[col])
151
152   # return the set of trackable objects
153   return self.objects
```

We loop over the `unusedCols` indexes (**Line 149**) and we register each new centroid (**Line 150**). Finally, we'll return the set of trackable objects to the calling method (**Line 153**).

Understanding the centroid tracking distance relationship

Our centroid tracking implementation was quite long, and admittedly, the most confusing aspect of the algorithm is**Lines 81-93**.

If you're having trouble following along with what that code is doing you should consider opening a Python shell and performing the following experiment:

```
Simple object tracking with OpenCV                                    Python
1 >>> from scipy.spatial import distance as dist
2 >>> import numpy as np
3 >>> np.random.seed(42)
4 >>> objectCentroids = np.random.uniform(size=(2, 2))
5 >>> centroids = np.random.uniform(size=(3, 2))
6 >>> D = dist.cdist(objectCentroids, centroids)
7 >>> D
8 array([[0.82421549, 0.32755369, 0.33198071],
9        [0.72642889, 0.72506609, 0.17058938]])
```

Once you've started a Python shell in your terminal with the `python` command, import `distance` and `numpy` as shown on **Lines 1 and 2**).

Then, set a seed for reproducibility (**Line 3**) and generate 2 (random) existing `objectCentroids` (**Line 4**) and 3 `inputCentroids` (**Line 5**).

From there, compute the Euclidean distance between the pairs (**Line 6**) and display the results (**Lines 7-9**). The result is a matrix `D` of distances with two rows (# of existing object centroids) and three columns (# of new input centroids).

Just like we did earlier in the script, let's find the minimum distance in each row and sort the indexes based on this value:

```
Simple object tracking with OpenCV                                    Python
10 >>> D.min(axis=1)
11 array([0.32755369, 0.17058938])
12 >>> rows = D.min(axis=1).argsort()
13 >>> rows
14 array([1, 0])
```

First, we find the minimum value for each row, allowing us to figure out which existing object is closest to the new input centroid (**Lines 10 and 11**). By then sorting on these values (**Line 12**) we can obtain the indexes of these `rows` (**Lines 13 and 14**).

In this case, the second row (index `1`) has the smallest value and then the first row (index `0`) has the next smallest value.

Using a similar process for the columns:

```
Simple object tracking with OpenCV                                    Python
15 >>> D.argmin(axis=1)
16 array([1, 2])
17 >>> cols = D.argmin(axis=1)[rows]
18 >>> cols
19 array([2, 1])
```

…we first examine the values in the columns and find the index of the value with the smallest column (**Lines 15 and 16**).

We then sort these values using our existing `rows` (**Lines 17-19**).

Let's print the results and analyze them:

```
Simple object tracking with OpenCV                                    Python
>>> print(list(zip(rows, cols)))
[(1, 2), (0, 1)]
```

The final step is to combine them using `zip` (**Lines 20**). The resulting list is printed on **Line 21**.

Analyzing the results, we find that:

- `D[1, 2]` has the smallest Euclidean distance implying that the second existing object will be matched against the third input centroid.
- And `D[0, 1]` has the next smallest Euclidean distance which implies that the first existing object will be matched against the second input centroid.

I'd like to reiterate here that now that you've reviewed the code, you should go back and review the steps to the algorithm in the previous section. From there you'll be able to associate the code with the more linear steps outlined here.

Implementing the object tracking driver script

Now that we have implemented our `CentroidTracker` class, let's put it to work with an object tracking driver script.

The driver script is where you can use your own preferred object detector, provided that it produces a set of bounding boxes. This could be a Haar Cascade, HOG + Linear SVM, YOLO, SSD, Faster R-CNN, etc. For this example script, I'm making use of OpenCV's deep learning face detector, but feel free to make your own version of the script which implements a different detector.

Inside this script, we will:

- Work with a live `VideoStream` object to grab frames from your webcam
- Load and utilize OpenCV's deep learning face detector
- Instantiate our `CentroidTracker` and use it to track face objects in the video stream
- And display our results which includes bounding boxes and object ID annotations overlaid on the frames

When you're ready, open up `object_tracker.py` from today's ***"Downloads"*** and follow along:

```
Simple object tracking with OpenCV                                    Python
# import the necessary packages
from pyimagesearch.centroidtracker import CentroidTracker
from imutils.video import VideoStream
import numpy as np
import argparse
import imutils
import time
import cv2

# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-p", "--prototxt", required=True,
 help="path to Caffe 'deploy' prototxt file")
ap.add_argument("-m", "--model", required=True,
 help="path to Caffe pre-trained model")
ap.add_argument("-c", "--confidence", type=float, default=0.5,
 help="minimum probability to filter weak detections")
args = vars(ap.parse_args())
```

First, we specify our imports. Most notably we're using the `CentroidTracker` class that we just reviewed. We're also going to use `VideoStream` from `imutils` and OpenCV.

We have three command line arguments which are all related to our deep learning face detector:

- `--prototxt` : The path to the Caffe "deploy" prototxt.
- `--model` : The path to the pre-trained model models.
- `--confidence` : Our probability threshold to filter weak detections. I found that a default value of `0.5` is sufficient.

The prototxt and model files come from OpenCV's repository and I'm including them in the ***"Downloads"*** for your convenience.

Note: *In case you missed it at the start of this section, I'll repeat that you can use any detector you wish. As an example, we're using a deep learning face detector which produces bounding boxes. Feel free to experiment with other detectors, just be sure that you have capable hardware to keep up with the more complex ones (some may run best with a GPU, but today's face detector can easily run on a CPU).*

Next, let's perform our initializations:

```
Simple object tracking with OpenCV                                    Python
# initialize our centroid tracker and frame dimensions
```

```
# initialize our centroid tracker and frame dimensions
ct = CentroidTracker()
(H, W) = (None, None)

# load our serialized model from disk
print("[INFO] loading model...")
net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])

# initialize the video stream and allow the camera sensor to warmup
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
time.sleep(2.0)
```

In the above block, we:

- Instantiate our `CentroidTracker` , `ct` (**Line 21**). Recall from the explanation in the previous section that this object has three methods: (1) `register` , (2) `deregister` , and (3) `update` . We're only going to use the `update` method as it will register and deregister objects automatically. We also initialize `H` and `W` (our frame dimensions) to `None` (**Line 22**).
- Load our serialized deep learning face detector model from disk using OpenCV's DNN module (**Line 26**).
- Start our `VideoStream` , `vs` (**Line 30**). With `vs` handy, we'll be able to capture frames from our camera in our next `while` loop. We'll allow our camera `2.0` seconds to warm up (**Line 31**).

Now let's begin our `while` loop and start tracking face objects:

Simple object tracking with OpenCV — Python

```
# loop over the frames from the video stream
while True:
 # read the next frame from the video stream and resize it
 frame = vs.read()
 frame = imutils.resize(frame, width=400)

 # if the frame dimensions are None, grab them
 if W is None or H is None:
  (H, W) = frame.shape[:2]

 # construct a blob from the frame, pass it through the network,
 # obtain our output predictions, and initialize the list of
 # bounding box rectangles
 blob = cv2.dnn.blobFromImage(frame, 1.0, (W, H),
  (104.0, 177.0, 123.0))
 net.setInput(blob)
 detections = net.forward()
 rects = []
```

We loop over frames and `resize` them to a fixed width (while preserving aspect ratio) on **Lines 34-47**. Our frame dimensions are grabbed as needed (**Lines 40 and 41**).

Then we pass the frame through the CNN object detector to obtain predictions and object locations (**Lines 46-49**).

We initialize a list of `rects` , our bounding box rectangles on **Line 50**.

From there, let's process the detections:

Simple object tracking with OpenCV — Python

```
 # loop over the detections
 for i in range(0, detections.shape[2]):
  # filter out weak detections by ensuring the predicted
  # probability is greater than a minimum threshold
  if detections[0, 0, i, 2] > args["confidence"]:
   # compute the (x, y)-coordinates of the bounding box for
   # the object, then update the bounding box rectangles list
   box = detections[0, 0, i, 3:7] * np.array([W, H, W, H])
   rects.append(box.astype("int"))

   # draw a bounding box surrounding the object so we can
   # visualize it
   (startX, startY, endX, endY) = box.astype("int")
   cv2.rectangle(frame, (startX, startY), (endX, endY),
    (0, 255, 0), 2)
```

We loop over the detections beginning on **Line 53**. If the detection exceeds our confidence threshold, indicating a valid detection, we:

- Compute the bounding box coordinates and append them to the `rects` list (**Lines 59 and 60**)
- Draw a bounding box around the object (**Lines 64-66**)

Finally, let's call `update` on our centroid tracker object, `ct` :

Simple object tracking with OpenCV — Python

```
 # update our centroid tracker using the computed set of bounding
 # box rectangles
 objects = ct.update(rects)
```

```
71
72  # loop over the tracked objects
73  for (objectID, centroid) in objects.items():
74   # draw both the ID of the object and the centroid of the
75   # object on the output frame
76   text = "ID {}".format(objectID)
77   cv2.putText(frame, text, (centroid[0] - 10, centroid[1] - 10),
78    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
79   cv2.circle(frame, (centroid[0], centroid[1]), 4, (0, 255, 0), -1)
80
81  # show the output frame
82  cv2.imshow("Frame", frame)
83  key = cv2.waitKey(1) & 0xFF
84
85  # if the `q` key was pressed, break from the loop
86  if key == ord("q"):
87   break
88
89 # do a bit of cleanup
90 cv2.destroyAllWindows()
91 vs.stop()
```

The `ct.update` call on **Line 70** handles the heavy lifting in our simple object tracker with Python and OpenCV script.

We would be done here and ready to loop back to the top if we didn't care about visualization.

But that's no fun!

On **Lines 73-79** we display the centroid as a filled in circle and the unique object ID number text. Now we'll be able to visualize the results and check to see if our `CentroidTracker` properly keeps track of our objects by associating the correct IDs with the objects in the video stream.

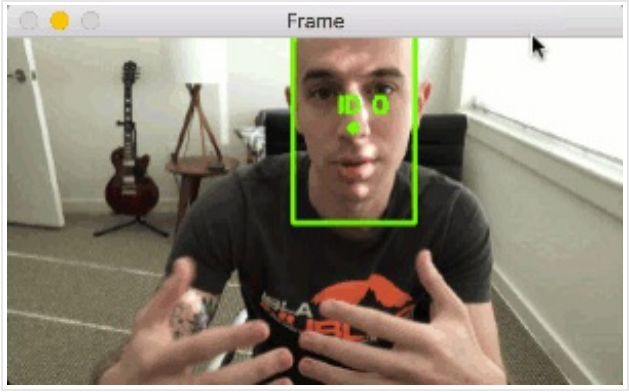
We'll display the frame on **Line 82** until the quit key ("q") has been pressed (**Lines 83-87**). If the quit key is pressed, we simply `break` and perform cleanup (**Lines 87-91**).

Centroid object tracking results

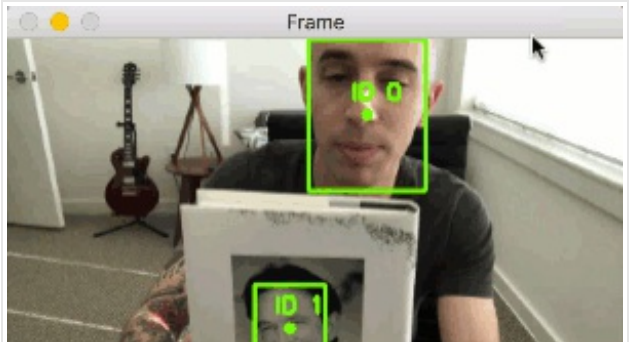
To see our centroid tracker in action using the ***"Downloads"*** section of this blog post to download the source code and OpenCV face detector. From there, open up a terminal and execute the following command:

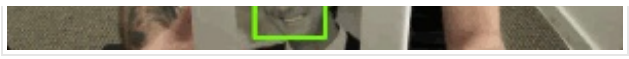
```
Simple object tracking with OpenCV                                    Shell
1 $ python object_tracker.py --prototxt deploy.prototxt \
2  --model res10_300x300_ssd_iter_140000.caffemodel
3 [INFO] loading model...
4 [INFO] starting video stream...
```

Below you can see an example of a single face (my face) being detected and tracked:



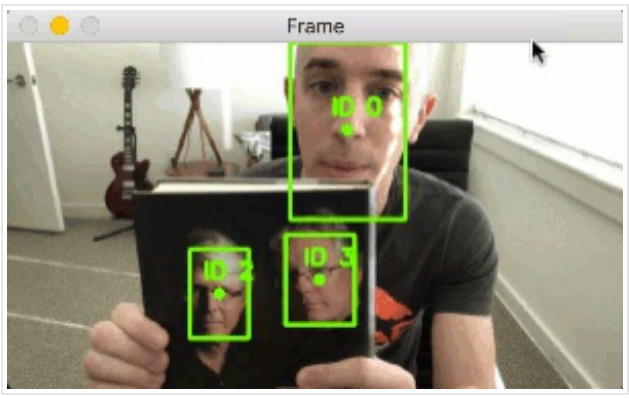
This second example includes *two* objects being correctly detected and tracked:





Notice how I even though the second face is “lost” once I move the book cover outside the view of the camera, our object tracking is able to pick the face back up again when it comes into view. If the face had existed *outside* the field of view for more than 50 frames, the object would have been deregistered.

The final example animation here demonstrates tracking *three* unique objects:



Again, despite object ID #2 being unsuccessfully detected between some frames, our object tracking algorithm is able to find it again and associate it with its original centroid.

For a more detailed demonstration of our object tracker, including commentary, be sure to refer to the video below:

Limitations and drawbacks

While our centroid tracker worked great in this example, there are two primary drawbacks of this object tracking algorithm.

The first is that it requires that object detection step to be run on every frame of the input video.

- For very fast object detectors (i.e., color thresholding and Haar cascades) having to run the detector on every input frame is likely not an issue.
- But if you are (1) using a significantly more computationally expensive object detector such as HOG + Linear SVM or deep learning-based detectors on (2) a resource-constrained device, your frame processing pipeline will slow down *tremendously* as you will be spending the entire pipeline running a very slow detector.

The second drawback is related to the underlying assumptions of the centroid tracking algorithm itself —*centroids must lie close together between subsequent frames.*

- This assumption typically holds, but keep in mind we are representing our 3D world with 2D frames —**what happens when an object overlaps with another one?**
- The answer is that **object ID switching could occur**.
- If two or more objects overlap each other to the point where their centroids intersect and instead have the minimum distance to the other respective object, the algorithm may (unknowingly) swap the object ID.
- It’s important to understand that the overlapping/occluded object problem is *not* specific to centroid tracking — it happens for many other object trackers as well, including advanced ones.
- However, the problem is more pronounced with centroid tracking as we relying strictly on the Euclidean distances between centroids and no additional metrics, heuristics, or learned patterns.

As long as you keep these assumptions and limitations in mind when using centroid tracking the algorithm will work wonderfully for you.

Summary

In today's blog post you learned how to perform simple object tracking with OpenCV using an algorithm called**centroid tracking**.

The centroid tracking algorithm works by:

1. Accepting bounding box coordinates for each object in every frame (presumably by some object detector).
2. Computing the Euclidean distance between the centroids of the *input* bounding boxes and the centroids of *existing* objects that we already have examined.
3. Updating the tracked object centroids to their new centroid locations based on the new centroid with the smallest Euclidean distance.
4. And if necessary, marking objects as either "disappeared" or deregistering them completely.

Our centroid tracker performed well in this example tutorial but has two primary downsides:

1. It requires that we run an object detector for each frame of the video — if your object detector is computationally expensive to run you would not want to utilize this method.
2. It does not handle overlapping objects well and due to the nature of the Euclidean distance between centroids, it's actually possible for our centroids to "swap IDs" which is far from ideal.

Despite its downsides, centroid tracking can be used in quite a few object tracking applications provided (1) your environment is somewhat controlled and you don't have to worry about potentially overlapping objects and (2) your object detector itself can be run in real-time.

If you enjoyed today's blog post, be sure to *download the code using the form below.* I'll be back next week with another object tracking tutorial!

Downloads:

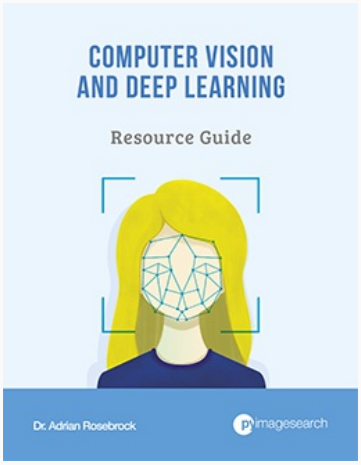
If you would like to download the code and images used in this post, please enter your email address in the form below. Not only will you get a .zip of the code, I'll also send you a **FREE 17-page Resource Guide on Computer Vision, OpenCV, and Deep Learning.** Inside you'll find my hand-picked tutorials, books, courses, and libraries to help you master CV and DL! Sound good? If so, enter your email address and I'll send you the code immediately!

Email address:

Your email address

DOWNLOAD THE CODE!

Resource Guide (it's totally free).



Enter your email address below to get my **free 17-page Computer Vision, OpenCV, and Deep Learning Resource Guide PDF**. Inside you'll find my hand-picked tutorials, books, courses, and Python libraries to help you master computer vision and deep learning!

Your email address

DOWNLOAD THE GUIDE!

centroid tracking, **object detection**, **object tracking**, **python**

‹ OpenCV Tutorial: A Guide to Learn OpenCV OpenCV Object Tracking ›

219 Responses to *Simple object tracking with OpenCV*

fdkssdks July 23, 2018 at 11:18 am #

REPLY

Thanks, Adrian for this post. I was looking forward to it

I am going to implement it soon and see how it goes.

Also, opencv released this tracker just for the info, you may already know about it.

GOTURN : Deep Learning based Object Tracking

Adrian Rosebrock July 23, 2018 at 2:21 pm #

REPLY

I saw! It's a purely deep learning-based object detector. I'm sure I will utilize it in a future post and demo how to use it

smit August 21, 2018 at 6:10 am #

REPLY

Any update on the demo by using GOTURN @Adrian

Adrian Rosebrock August 21, 2018 at 6:43 am #

REPLY

I haven't had any time to play around with GOTURN yet. When I do I'll be writing a dedicated blog post.

Terlumun Nyamor November 12, 2018 at 10:40 pm #

REPLY

hey Is the rects variable constantly growing in this code?

Adrian Rosebrock November 13, 2018 at 4:22 pm #

REPLY

Can you be a bit more specific regarding your question?

damvantai February 12, 2019 at 2:01 am #

REPLY

i tried with GOTURN in opencv 3.4, but speed is very slow in cpu of mylaptop, my precision don't good as use median flow tracker!!!

Zubair Ahmed July 23, 2018 at 12:04 pm #

REPLY

Excell…excellent post!

I read totally understood it in the first go for reasons I have shared in my email

There is a little typo 'First, we find the minimum value for each row, allowing [is]->[us] to figure out which existing object is closest to the new input centroid'

Looking forward to the whole series

Adrian Rosebrock July 23, 2018 at 2:23 pm #

REPLY

Thanks Zubair! Typo fixed

Yassine Benhajali July 23, 2018 at 1:36 pm #

REPLY

Hi Adrian, big fan of your blog, you do an awesome job.

I guess it's also possible to use your code for other than faces tracking (let say dogs or cat).
do you know to get that working.
Thanks,

Adrian Rosebrock July 23, 2018 at 2:22 pm #

REPLY

You would swap out the face detector and utilize your own object detector. This post will help you get started.

Otherwise, you will want to wait until next week where I'll have a specific example of using a different object detector (such as a dog or cat one).

Sneha Agarwal March 18, 2019 at 3:12 pm #

REPLY

Hey

Are you don with the code to detect cars?

Adrian Rosebrock March 19, 2019 at 9:56 am #

REPLY

I'll be covering it in my upcoming Computer Vision + Raspberry Pi book. Make sure you're on the PyImageSearch newsletter to be notified when it's released!

Jay Abrams July 23, 2018 at 1:38 pm #

REPLY

thank you for your time and effort Adrian!

Adrian Rosebrock July 23, 2018 at 2:20 pm #

REPLY

Thanks Jay!

satinder singh July 23, 2018 at 2:56 pm #

REPLY

Thank you for your blogs adrian, I enjoy reading them. Looking forward for the next blogs. Thank you again.

Adrian Rosebrock July 23, 2018 at 4:22 pm #

REPLY

Thank you Satinder

Osmin July 23, 2018 at 9:42 pm #

REPLY

Thank you very much…

I am getting a lot of help!!

Can I use your code in my APP ??

Adrian Rosebrock July 24, 2018 at 8:27 am #

REPLY

Yes, you can use the code in your app. I would appreciate a credit or link back to the PyImageSearch blog though

zhao July 23, 2018 at 10:40 pm #

REPLY

Thanks, Adrian for this post. I want to use your blog to track fish. How do you train fish detectors?
Thank you again.

Adrian Rosebrock July 24, 2018 at 8:27 am #

REPLY

I would suggest starting with this post on deep learning object detection fundamentals. That guide will get you going in the right direction.

Stalon July 23, 2018 at 11:43 pm #

REPLY

Adrian, thank you very much for the sharing.
I really learned a lot from this post.

I have a question about the occlusion solution.
What is the mechanism of solving occlusion problem while tracking?
Is that similar as Kalman Filter or something?

Adrian Rosebrock July 24, 2018 at 8:26 am #

REPLY

There is no true “solution” to occlusion. There are methods and heuristics that we can apply (based on the project specifications) that can help with occlusion but there is no one true “solution” to occlusion.

coffee code March 15, 2019 at 6:49 pm #

REPLY

i also need solution to occlusions. Adrian please help, youre doing great

sohail July 24, 2018 at 3:39 am #

REPLY

Great Work.

Adrian Rosebrock July 24, 2018 at 8:26 am #

REPLY

Thanks Sohail

ki2rin July 24, 2018 at 3:40 am #

REPLY

Thank you for another great tutorial, Adrian.

I have only used Haar + Adaboost in the past(around 2011 or 2012) for face detection.
What I remember is 1) it was super fast, but 2) it accepts a lot of false positives.
So, in my natural thought, the vanilla implementation of Adaboost can hardly be used for the tracking system, am I right?
Otherwise, the centroid tracking seems to be much promising for this purpose.
One thing I wonder is that how robust is the centroid tracking when the object moves fast?

Adrian Rosebrock July 24, 2018 at 8:29 am #

REPLY

Haar cascades are face detectors, they are not object trackers. You use Haar cascades to compute the bounding box of a face, then the resulting centroid, and then finally pass that info to your object tracker.

That said, you are correct that if your detector is not accurate than you cannot expect the tracker to be accurate as well. The detector must be working accurately — that is why I choose to use a more accurate deep learning-based object detector in this post.

David July 24, 2018 at 5:53 am #

REPLY

Hi Adrian!
I may have find an error in your code. If I initialize the CentroidTracker() class with a lower maxDisappeared (10 for example) I end up with this error in line 40 of centroidtracker.py : “RuntimeError: OrderedDict mutated during iteration”.
I solved it by replacing line 40 with the following code: “for objectID in list(self.disappeared.keys()):”

I am looking forward for your blog post in this series, keep going you rock!

David

Adrian Rosebrock July 24, 2018 at 8:30 am #

REPLY

Thanks for sharing, David!

Nihat Guliyev October 17, 2018 at 8:38 am #

REPLY

Thank you very much. This replacement is very vital. I am running on CPU and sometimes it gives that mentioned error. So it’s solved)

toz March 15, 2019 at 3:59 am #

REPLY

had the same error:

“RuntimeError: OrderedDict mutated during iteration”.
I solved it by replacing line 40 with the following code: “for objectID in list(self.disappeared.keys()):”

https://www.pyimagesearch.com/2018/07/23/simple-object-tracking-with-opencv/#comment-471851

@ adrian: you should update your code-sample for mutating orderdict to be error-proof also in python 3.x…

Martin C. July 13, 2019 at 4:01 pm #

REPLY

Thank you!!

Paul Zikopoulos July 24, 2018 at 7:12 am #

REPLY

No biggie, slight typo in comment of code line 86 of 1st script …

with the smallest value as at the *front* of the index

as > is

GREAT ARTICLE … between this and my amazing course I have a full time learner’s job

Adrian Rosebrock July 24, 2018 at 8:30 am #

REPLY

Thanks Paul

Thanks Paul

Stefan Karos July 24, 2018 at 10:28 am #

REPLY

Thank you Adrian for your clarity and devotion to this subject. Without saying the words, you even inspire novices like me to think about use cases.Would this approach be appropriate to use to count people walking in a street? For example I live on a ‘dead end’ (cul de sac for the masses). I would like to count the people coming and going into this ‘gated community’. Walking from left to right would be entering. Walking from right to left would be exiting. My camera can be positioned from 5 to 30 feet above the street looking down on its entire width (20 feet). So I suspect I would need to track object motion and and assign a vector value to each centroid. (Positive values for entry and negative values for exit). Summing the signs would give me the total counts. Looking at vector magnitude would also give me their speed (how much sauntering, walking, running, etc)

Such a project (especially if implementable on a raspberry Pi) has enormous utility. You could even make and sell a ‘people counter’ ! Knowing how many people enter and leave a building, a museum (or museum wing), a graduation a theme park adds significant security in addition to providing useful information to improve facilities.

Forgive me if this is not the right place to post this but I did not find your forums on this site. But perhaps that’s intentional-when would you find time to moderate forums?!

Adrian Rosebrock July 24, 2018 at 10:45 am #

REPLY

Hey Stefan — I’ll actually be covering “people counting” later in this series of blog posts. Stay tuned as I believe the implementation I’ll be sharing will help you solve your project. Outside of the actual blog posts go, you should take a look at the PyImageSearch Gurus course which includes a dedicated, private community forums.

Yash July 27, 2018 at 12:49 am #

REPLY

Please have a look at optical flow (Dense) and you will find a better and robust solution of your problem.
Though if using moving/shaky camera to record then don’t forgot to stabilize the video using template matching before applying optical flow as flow could give an wrong information in case of relative motion due to the motion of the camera.
Trust me result would be mind blowing.

Prajesh Sanghvi July 25, 2018 at 2:35 am #

REPLY

Hey Adrian, thank you so much for all this info. I am a beginner with python, and i keep on getting the error: object_tracker.py: error: argument -p/–prototxt is required

even after assigning the path in the “help” part of the argument section
i.e:

ap = argparse.ArgumentParser()
ap.add_argument(“-p”, “–prototxt”, required=True,
help=”/home/pi/CamProject/FaceTrack/simple-object-tracking/deploy.prototxt”)

can you please help me out?
also i cant figure out which of the files is for the MODEL section.

Thank you

Adrian Rosebrock July 25, 2018 at 7:55 am #

REPLY

If you are new to command line arguments that’s okay, but you’ll want to read this tutorial on command line arguments first .

Farooq July 25, 2018 at 10:53 am #

REPLY

Thanks for the awesome post Adrian.
I think you have a typo somewhere in step two;
“… Euclidean distances between each pair of original centroids (purple) and new centroids (purple)…” I think it should be new centroid(yellow)

centroid(yellow)

Adrian Rosebrock July 27, 2018 at 5:35 am #

REPLY

You are correct, thank you for reporting this typo — it's now been fixed.

Navaneeth Krishnan July 25, 2018 at 11:28 am #

REPLY

adrian i am a big fan of yours. can you make a post on "openface" a library for face recognition. i have seen your face recognition post and i have tried but some times the face name was wrongly mentioned. i we combine face recognition and tracking then the occlusion problem will be solved

Adrian Rosebrock July 27, 2018 at 5:33 am #

REPLY

Hey Navaneeth, I will consider doing a blog post dedicated to OpenFace but I cannot guarantee if/when that may be. As far as your project goes, if the face is being incorrectly identified you should spend more time playing with the parameters to the "face_recognition" library, in particular the distance parameters, to help reduce incorrect classifications. You may also want to look into fine-tuning the network as well.

Prajesh Sanghvi July 25, 2018 at 11:29 am #

REPLY

Thank you so much!!

Ian Carr-de Avelon July 26, 2018 at 4:53 am #

REPLY

What about using the hardware to reduce the CPU processing?
If you can only do a full search and identify in a fraction of the frames, the faces or objects could be tracked if you have MPEG encoding of all the video. MPEG only sends a fraction of the frames and then vectors saying how bits should be moved around to recreate the following frames. Calculating those vectors is highly efficiently coded or even available in hardware. Eg. you can pay extra for a code number to uncripple the hardware encoders in the PI.
This is an idea I've had for a while and occasionally searched for useful libraries. I've just come across this:
http://picamera.readthedocs.io/en/release-1.12/recipes2.html#recording-motion-vector-data
which gives Python code to get the vectors from a PIcam.
Yours
Ian

Adrian Rosebrock July 27, 2018 at 5:32 am #

REPLY

You could certainly develop heuristics and assumptions as well to help reduce computational burden. But keep in mind that those assumptions may only be relevant to a specific project and may not be able to be used across all projects.

Secondly, deep learning-based object detectors included an "objectness" component that quickly determines which areas of the image warrant additional computation, including applying the object detector. They are still slower (but more accurate) than your traditional Haar cascades but they yield higher accuracy as well. As hardware progresses in the future these models will run faster.

Cristian Benglenok July 26, 2018 at 10:11 am #

REPLY

very good tutorial, we have implemented an object counter using deep learning to identify the object and using OpenCV an algorithm to perform the count. It's pretty slow, we'll try this method to improve the speed.

Thanks

Adrian Rosebrock July 27, 2018 at 5:30 am #

REPLY

Adrian Rosebrock July 27, 2018 at 5:30 am #

Hey Christian — congrats on implementing a successful project! As far as deep learning goes, have you tried pushing the inference to the GPU? That would speedup the pipeline.

amir taherkhani July 27, 2018 at 1:22 am #

REPLY

hi adrian i am iranian user on your website i can not buy your books but your free tutorials is very usefull for me Thank you very much for sharing …

Adrian Rosebrock July 27, 2018 at 5:28 am #

REPLY

I am happy that you are enjoying the free blog posts Amir. It's wonderful to have you as part are of the PyImageSearch family Keep learning from them!

Rob July 27, 2018 at 8:19 am #

REPLY

HI Adrian, your blog is totally awesome!

Prior to stumbling upon your site, using an RPI and a windows pc I was able to create a robotic camera which streams wirelessy (almost no latency) and is able to be aimed via servos and the arrow controls on the laptop. Moving forward, I want to add object tracking as it applies to human entities. I recently received this blog and I thought that it would be a good jump off point for the upgrades. WIth that said, I am curious as to whether you have blogged about implementing the face detector (or a general human detector) and where it can be found?

Thus far, the RPI is a streaming slave and the majority of the heavy lifting is done on the laptop. On that note, the computational resources are limited… so I am really hoping there is an efficient algorithm/implementation that already exists.

I would appreciate any advice you can provide.

Thank you.

Adrian Rosebrock July 31, 2018 at 12:04 pm #

REPLY

I actually have blog posts dedicated to both topics. See this post on face detection along with this post on general object detection, including person detection.

Sai Teja July 28, 2018 at 5:05 pm #

REPLY

Hi Adrian,

Thanks for the amazing post. You are awesome man. I am eagerly waiting and would request you to do a tutorial on object tracking using advanced methods as you mentioned in the post that you will be doing(kernel-based and correlation-based tracking algorithms)

Adrian Rosebrock July 31, 2018 at 9:57 am #

REPLY

You can find the first of the more advanced methods in this post. Stay tuned for more object tracking posts coming soon!

Zayyana July 29, 2018 at 9:37 am #

REPLY

So much thanks mr. I usually use

"…
for(x,y,w,h) in object:
(x_centre,y_centre)=(x+w/2,y+h/2)
…"
because i confused when use numpy to get the centre. computing (x,y,w,h) much easier to understand

Z Abrams July 31, 2018 at 10:50 am #

REPLY

Hi,
As usual, a great post (though the simplicity of your next post from July 30th really blew me away – once I got all the Python/Opencv Versions to work [updated to 3.4.2 + updated contrib package]
However, one thing I couldn't figure out was what the indices of the "detections" are: e.g. detections[0, 0, i, 3:7]. I get that i is the iterator, and 3:7 apparently give the bounding box coordinates, but what data type is this, and what other things are there?
I couldn't figure it out online – not even sure what data structure I'm looking at (net? forward?). Unfortunately the OpenCV documentation can sometimes be confusing [it's actually usually a bit easier for C++].

Also, on another note, I actually do everything in Windows (10), and usually, it's even easier to do than on Linux (Ubuntu). Windows packages are usually pretty easy to install – no Cmake or whatnot, though there ARE some tricks you must learn. So perhaps you can update your "how to install python/opencv/etc" tutorials. [The one exception is dlib – which is a pain, but it works via my Anaconda version of python 3.6 and its built in virtual environment system]. [Also, this statement is true regarding your Deep Learning book, where I'm using a GPU as well, and got it up and running very quickly – faster than Ubuntu]

Adrian Rosebrock July 31, 2018 at 10:57 am #

REPLY

1. The "detections" variable is a bit hard to fully understand. I don't have a fully-documented example on it (I'll consider it for the future) but for the time being be sure to refer to my source code on examples of how to extract the bounding box coordinates, labels, and corresponding probabilities.

2. I don't officially support Windows here on the PyImageSearch blog. You are more than welcome to use whatever OS you feel more comfortable with. In general, I've found that most readers struggle significantly more with Windows than Ubuntu or macOS for computer vision or deep learning.

alibigdeli August 7, 2018 at 2:50 am #

REPLY

i simply love your passion and mind blowing ideas
from old times i remember there was a windows application called flutter and it was a simple detection of hand poses to play or stop music or going left or right. that idea was bugging me for so long cause i was a kid but now i am aware of what was happening back there.
my struggle is to write a code to detect and track hand in a frame video which is from my web cam and then use it as a mouse. like moving around and clicking.
the problem is i know the whole story like i have been coding it with a haar and kalman filter to follow it and simply move the mouse too but clicking and other stuffs are the hard part.
after seeing this post it inspires me with other ideas again but still need time to learn. can you point me in right direction or even a post for this matter cause i know lots of people are after this project to simply use it for ai products but note commercial simply at home

Adrian Rosebrock August 7, 2018 at 6:29 am #

REPLY

Just so I understand correctly — you are able to track the hand via Haar cascades and Kalman filters but you need a suggestion for the GUI interaction library? I don't have a ton of experience in that area but I really like pyautogui.

alibigdeli August 7, 2018 at 5:54 pm #

REPLY

no actually its far deeper
i am using pynput for that purpose
the problem is tracking hand with the pose hand like simply making a fist to click and following the fist to move the mouse too
there are several ways to do it but my code was to get the resolution of the screen and adapt the point i am showing ,to follow like a point in the middle of the rectangle as (x,y) of the web camera feed and get the simulated point on another resolution, so that i could be covering the wider screens dynamically.
1- the main problem is double hands detection (which i am working on it but not successful yet)
2- another problem is false detections rate is high cause of haarcascade usage
lots of these had been used with ir cameras like kinect depth and so on…
but i cannot afford them all i have is my webcam and normal knowledge of image proccessing
3- another problem is when hand is gone the pointer still follows the dot prediction which is annoying for me (as a matter of fact

seeing your post gave me an idea of reseting point after not detection of hand in specific time)
4- covering edges of the screen is almost not possible i know i have to come up with adding extra numbers to resolution to cover it
5- i have no idea if this is the right path i am going or not
like using haar should be changed with something else or even kalman filter should be combined with others too (this is the real question!)

Adrian Rosebrock August 9, 2018 at 3:04 pm #

REPLY

Haar cascades are fairly prone to false positives. HOG + Linear SVM detectors are more accurate as are deep learning-based object detectors. I would swap out your Haar cascades for one of those and see how far that gets you.

Sanjay August 10, 2018 at 4:53 am #

REPLY

Hello Adrian,

Thank you for this awesome project.I wanted to ask you like there are moments when people usually are engaged in mobile and hence there head is tilted down most of the time.I have used same face detector which you have used.The problem is that with little head tilted straight down it is not able to detect the face.Can you please suggest me how to overcome this problem?

P.S:- Have been following you since long like for me you are god of CV.Thank you.

Adrian Rosebrock August 10, 2018 at 6:04 am #

REPLY

Hey Sanjay, thank you for the kind words, I appreciate it. As far as the head titling, I assume by the “same model I’m using” you are referring to the deep learning face detector? If so, the model should be able to handle slight head tilts, but if the head is tilted too much it won’t work. You may want to consider training a face detector on head tilted images.

Secondly, the dlib library includes another deep learning-based face detector. You may want to give it a try and see if it gives you better detection accuracy. An example of how to use it can be found in this post.

Dhruvin .p. patel August 13, 2018 at 2:12 am #

REPLY

Hey How to Implement on Keras base tensorflow lib.

Adrian Rosebrock August 15, 2018 at 8:54 am #

REPLY

Hm, you don’t need either Keras or TensorFlow for this example. What specific functionality do you want Keras/TensorFlow for?

Sanjay August 13, 2018 at 2:26 am #

REPLY

Hello Adrian,

Currently I am running tracking on CPU using opencv dnn face detector.It misses slightly bounding box in some frame.My Question is If i shift to GPU does it make detection in every frame better? Also Is there any relation with distance the object is standing from the camera? Also does result changes if the object is standing on left or right side of the frame?

Adrian Rosebrock August 15, 2018 at 8:54 am #

REPLY

Running inference on the GPU will make the predictions faster, but not more accurate. Right or left side of the frame won’t matter but typically objects closer to the camera are easier to detect.

Gery August 13, 2018 at 11:00 am #

REPLY

Nice article, enjoyed reading it as usual. I used the centroid tracking algorithm with YOLO face detector and it works fine. Thank you Adrian.

Adrian Rosebrock August 15, 2018 at 8:48 am #

REPLY

Awesome, I'm glad to hear it Gery!

Serhat December 4, 2018 at 4:36 am #

REPLY

Hi, Can you tell me how did you implement it to YOLO. Thanks.

Adrian Rosebrock December 4, 2018 at 9:37 am #

REPLY

Take a look at my YOLO object detection tutorial to get started.

Sanjay August 16, 2018 at 3:07 am #

REPLY

Hello Adrian,

Can we define detection distance?For example it detects people at particular distance only or object that are more closer to camera?

Adrian Rosebrock August 17, 2018 at 7:28 am #

REPLY

Absolutely, but you would need the modify the code to perform that action. A general rule is that smaller objects are farther away and larger objects are closer to the camera. Combined with this code you can determine in which direction they are moving as well.

Sanjay August 21, 2018 at 1:16 am #

REPLY

Thanks a lot adrian. You are awesome <3

MANAS SIKRI January 19, 2019 at 7:38 am #

REPLY

You can calculate the area of the rectangle using the coordinates. The faces that are close to the camera will have larger area than those faces which are far away.

absolom August 16, 2018 at 4:33 am #

REPLY

thanks, good explanation

Sanjay August 20, 2018 at 6:01 am #

REPLY

Hello Adrian,

Thank you for your replies above. I want to share concern regarding object detection.When i try to detect from the recorded video in afternoon it is not able to detect people because of sunlight issue i think so and it is able to detect people at night. How do i resolve this issue?

Adrian Rosebrock August 22, 2018 at 9:57 am #

REPLY

What type of object detector are you using?

Sanjay August 23, 2018 at 2:25 am #

REPLY

The same which is used in this article.I even tried dlib and LBP and other detector as well even they are failing badly but then i found RetinaNet and it is able to detect quite awesomely.Can you tell me how can i convert any object detection model into tracking?

Adrian Rosebrock August 24, 2018 at 8:45 am #

REPLY

Object detection and object tracking are two totally different algorithms. You perform object detection and then object tracking. This tutorial demonstrates how to perform the entire pipeline.

sanjay September 3, 2018 at 1:02 am #

Thank you, Adrian. Can you tell me how can I do object tracking on mobile? Is there any tutorial where you have explained to do object detection or tracking on android mobile?

Adrian Rosebrock September 5, 2018 at 9:00 am #

You will need to port the code to Java + Android. OpenCV provides Java/Android bindings but I'm not familiar with them. You'll need to refer to the documentation.

Keesh September 11, 2018 at 8:51 am #

REPLY

Hi Adrian,

Thanks very much for this insightful project. I'm pretty new to this and I have a couple of questions if they aren't too difficult to briefly explain:
If I wanted to track the person with the lowest ID number to print out their pixel coordinates on the screen, how would I go about that?
Also, What parts of the centroid tracker would I need to update if I wanted to implement a Z (depth) axis from an IR depth camera?

Cheers

Adrian Rosebrock September 11, 2018 at 9:01 am #

REPLY

This code assumes you're working with a standard RGB image without any depth information. Adding depth information would require quite the overhaul and is honestly too far outside the context of this tutorial — it would require it's own dedicated tutorial.

Secondly, Line 73 starts a "for" loop that contains the (x, y)-coordinates for each object. You can sort the "objects" dictionary based on the object ID and then show only the object information with the smallest ID.

Keesh September 11, 2018 at 9:24 am #

REPLY

Thanks for the quick reply Adrian. I will try what you have suggested.
I have been working on a personal project successfully implementing a depth camera IR stream which is masked to the RGB stream. I was hoping to modify your code to implement the third depth axis found at the centroid location to give greater accuracy to the Euclidean distance mapping to prevent detection ID crossovers, but by the sounds of it, its likely out of my league.

Barry September 14, 2018 at 7:08 am #

REPLY

Hi! Thanks for the wonderful tutorial. Couple of questions though:

1. The part “detection = net.forward()” – what exactly does this forward function return? I tried printing it to find out, got an array of float values. I want to know what these values correspond to.

2. While trying to test the code, I am getting an error – for i in range(0, detections.shape[2]):
IndexError: tuple index out of range
The code logically looks fine, is this a run-time error?

3. Is this code compatible only with the Caffe model? Can this code be tried with the Darknet YOLO configuration as well, or does it need some modifications?

Adrian Rosebrock September 14, 2018 at 9:19 am #

REPLY

1. The detections is a NumPy array. It returns potential bounding box coordinates and their corresponding probabilities.

2. It looks like a run-time error of some sort. Make sure you’re using the “Downloads” section of the blog post to download the source code and models. You may have accidentally introduced a bug when copying and pasting.

3. It would need some modifications. I’ll try to do a YOLO tutorial in the future.

Barry September 14, 2018 at 10:29 pm #

REPLY

Ok! Thank you so much!

Would try to modify and make it YOLO consistent and share the results.

hadi September 16, 2018 at 3:10 am #

REPLY

hi dear Adrian. thanks for nice blog post. really good.
in using this post, after detection and tracking faces, I want to save frames that include ID 0 (MEAN YOUR FACE). but all faces saved.
how can I do it? how can separate boxes from each other to do something with everyone. thanks
I used in my code like this:
if objectID==0:
do something…… but this do for every face.

Adrian Rosebrock September 17, 2018 at 2:21 pm #

REPLY

Keep in mind that we’re just performing object tracking in this blog post, we’re not performing face recognition. I would suggest you read this post on face recognition before continuing.

hadi September 23, 2018 at 1:23 am #

REPLY

dear Adrian I don’t wanna to face recognition. I mean is that how can I people who has ID0 from others. ?? how separate ID0 box from another boxes? thanks

Adrian Rosebrock October 8, 2018 at 12:59 pm #

REPLY

The for loop on Line 73 allows you to associate the object ID with the bounding box.

Thanh September 19, 2018 at 4:30 am #

REPLY

Hi Adrian

Thank you for this great post. What camera (or webcame) did you use in this post? I want to buy one to get into Computer Vision.

REPLY

Adrian Rosebrock October 8, 2018 at 1:30 pm #

It was actually the builtin webcam on my iMac.

REPLY

smit September 19, 2018 at 11:02 am #

Hi @Adrian,
Can your model work if we give object to track in first frame (give bounding box in first frame) only?

REPLY

Adrian Rosebrock October 8, 2018 at 1:26 pm #

Yes, provided that the object is visible in the remaining frames and the first frame is a good representation of what the object looks like.

REPLY

Wim Valcke September 23, 2018 at 10:57 am #

Hi Adrian,

Thanks for the post.
The model definition wants an input of (1, 3, 300, 300) as numpy shape.
Why is the image taken from the video stream rescaled to a width of 400 ?
In a previous blog post about face detection with the same network the images were scaled to 300×300 pixels.

Best regards
Wim.

REPLY

Meera Chandran September 25, 2018 at 4:52 am #

Hi Adrian, Do you have this implemented in C++???

REPLY

Adrian Rosebrock October 8, 2018 at 12:45 pm #

Sorry, no, I only provide Python code here on PyImageSearch.

REPLY

Rawa September 25, 2018 at 10:11 am #

hello Adrian
I want to do face detection (haar) and based on that I want to track this face by (KCF)
is it possible? and how?

REPLY

Adrian Rosebrock October 8, 2018 at 12:42 pm #

Yes, of course. You will use your Haar cascade to detect the bounding box coordinates of the face. From there, pass in the bounding box coordinates to the KCF tracker. From there the face will be tracked.

REPLY

Rob September 27, 2018 at 5:16 pm #

As always, awesome blog!

I modified the code slightly as to not require imutils.video import VideoStream and instead just use cv2.VideoCapture(0). It works perfectly; however, it is really laggy by way of frame rate. Is this why imutils.video was used in this instance? The process now consumes approximately 60% cpu… which is enough, but the needle isn't pegged. Unfortunately I am running a laptop, that uses an AMD gpu so this could play a factor, but considering that the library is already trained that seemingly shouldn't play into it.

Using haar-cascade, I have zero lag on my frame rate, using cv2.VideoCapture(), and my cpu usage is actually higher.

On a side note, when using the haar-cascade if you modifiy the line:

faceRects = face.detect(grayImage, scaleFactor = 1.1, minNeighbors = 5,
minSize = (30, 30))

to:

faceRects = face.detect(grayImage[::4, ::4], scaleFactor = 1.08, minNeighbors = 6,
minSize = (30, 30))

and the lines:

for (fX, fY, fW, fH) in faceRects:
cv2.rectangle(frameClone, (fX, fY), (fX + fW, fY + fH), (0, 0, 255), 2)

to:

for (fX, fY, fW, fH) in faceRects:
cv2.rectangle(frameClone, (fX*4, fY*4), (fX*4 + fW*4, fY*4 + fH*4), (0, 0, 255), 2)

The haar version of face detect will consume about 2/3 less cpu, and detect face much better. Previously it was at ~75% cpu and it now runs at approximately 20% cpu.

Adrian Rosebrock October 8, 2018 at 12:28 pm #

REPLY

The reason we use the imutils "VideoStream" class rather than OpenCV's "cv2.VideoCapture" function is because VideoStream is threaded and is much faster. A separate thread is used to increase the throughput rate of the pipeline. You can read more about the speedup in this post.

Naj November 10, 2018 at 10:38 am #

REPLY

Hey Rob,

Can you help me implement this using Haar cascade?

Monwar Jahan September 29, 2018 at 4:55 am #

REPLY

Hi Adrian,
Thanks for the article

Can you please tell me how can I detect something in real time with a sample object.

Like, I want to detect a pen from a collection of pens that matches my sample pen.

Thanks agasin

Adrian Rosebrock October 8, 2018 at 12:18 pm #

REPLY

So, given an image that contains N pens you want to find *one* specific pen? How is that particular pen unique? Is it color? Shape?

Barry October 15, 2018 at 5:46 am #

REPLY

Hello!
I've used this code for a project of mine (would be giving credits in the acknowledgement).

I am running into an issue – my project deals with detecting objects and tracking them within a certain ROI, beyond which it should stop

tracking. Now when the object goes out of the ROI box, the detection as well as the tracking stops but the ID number stays on screen. Now, I understood the point that the entry for the particular ID number in the Ordered Dictionary will be there for atmost n number of frames defined in the variable maxDisappeared. Currently, it is set to 10. But in spite of crossing the 10 frame limit, the entry stays in the DIctionary and doesn't get deregistered till a new object gets added in the dictionary. Only then the previous ID gets deleted.

I assumed this was a minor issue, but now if I introduce a new object even after 70-80 frames, the last ID stays in the Dictionary.

The ID assignment code is completely same as mentioned here, I did not make any changes to it.

Do help.

Thanks

Adrian Rosebrock October 16, 2018 at 8:33 am #

REPLY

Hey Barry — just to clarify which dictionary object are you referring to? We use quite a few dictionaries in the code so I want to make sure we're on the same page. Could you point to a specific line(s) number?

Ram October 23, 2018 at 2:55 am #

REPLY

hey adrian, i have a problem statement of counting vehicles moved left/right/u turn etc etc…can you suggest me how can i start for that

Adrian Rosebrock October 29, 2018 at 2:14 pm #

REPLY

My OpenCV People Counter tutorial can easily be extended to vehicle counting by changing the class label that is detected.

Sam October 24, 2018 at 7:25 am #

REPLY

Thank you very much for the post Adrian!

Are you also writing a post on object tracking without detection in every frame? What I mean to say is, in the method above, you are detecting the object(s) to be tracked in every frame. As you might know, there are methods where a discriminative classifier is used to calculate the confidence scores for each patch around a location to predict the object's location in the current frame and also to update the classifier for the next frame. It would be nice to have a tutorial on this approach as well.

Adrian Rosebrock October 29, 2018 at 2:03 pm #

REPLY

I actually wrote a blog post that covers the hybrid approach of intermittent object detection mixed with tracking. You can find it here.

Abhishek October 29, 2018 at 2:18 am #

REPLY

Hey Adrian ! Nice work! Can we use this code for videos that are already available instead of a real time video?

Also how can i modify this code for vehicle tracking ??

Thanks in advance . Cheers!

Adrian Rosebrock October 29, 2018 at 1:14 pm #

REPLY

Yes, you can use this code for existing videos. Just use the cv2.VideoCapture function.

Ram October 31, 2018 at 3:12 am #

REPLY

Hey adrian thanks for the reply,

ccentroid tracking algo is not working well for my use case as id are getting switched often and i am losing track of a vehicle to get the path of the vehicle…can you suggest me how can i improve that

Naj November 10, 2018 at 10:37 am #

REPLY

Hey Adrian, what a excellent work you have put here.

I am super new in programming. Currently I m using Haar cascade to detect cars. You mentioned that it can be implement using Haar cascade. Even after I have read Implementing the object tracking driver script also I can't figure out what to change. Is there any solution?

Abdullah khan November 23, 2018 at 5:35 pm #

REPLY

Hi,

adrian how we can save the bounding box coordinates.?

Adrian Rosebrock November 25, 2018 at 9:08 am #

REPLY

You mean save them to disk? I would suggest you look into basic Python file I/O before continuing. If you want to use a database you should refer to the documentation for whatever database software you are using.

Abdullah khan November 25, 2018 at 5:46 am #

REPLY

Hi,

Humble request, if you can write a similar blog for OBject tracking for YOLO (Darknet).As this blog uses Cafffe framework only.

Best regards,
Abdullah

Adrian Rosebrock November 25, 2018 at 8:51 am #

REPLY

I actually already have a tutorial on YOLO. You can learn how to combine deep learning object detectors with object tracking in this tutorial.

Jonathan Mejia November 29, 2018 at 11:29 pm #

REPLY

This is my first time working with image processing, I have downloaded the open CV from the example and have done the pip install the get the numpy imutil and scipy, however i would get a error in the terminal from the downloaded code which would not recognise the module name of the imulit, scipy. Any suggestion or instructions on what to do with the downloaded files after from the email and where should I place them??

ImportError: No module named scipy.spatial

Adrian Rosebrock November 30, 2018 at 8:46 am #

REPLY

Hey Jonathan, how did you install OpenCV, NumPy, and SciPy? Did you use the install tutorials I use here on the PyImageSearch blog? Or did you use a different set of instructions?

sir kam January 6, 2019 at 6:33 pm #

REPLY

hey adrian….how install pyimagesearch library I cannot find it in the available package is there any hints(

Adrian Rosebrock January 8, 2019 at 6:57 am #

REPLY

Use the “Downloads” section of this tutorial to download the source code, including the “pyimagesearch” module.

Tessa van der Heiden December 2, 2018 at 7:30 am #

REPLY

Hi Adrian, Thanks for the tutuorial. Do you know what happens, when one objects leaves and one arrives between frames? Example 3 objects, one appears left, one stays in the middle (id=1), one leaves right (id=2). Will the middle object get id=2 and the one that appears id=1?

danish December 3, 2018 at 7:09 am #

REPLY

How to calculate time duration for particular objects which are being track?

Rick December 3, 2018 at 10:59 pm #

REPLY

Hi Adrian,

Thanks for your wonderful blog. Has been following your blog for quite a while.

My project will be working on application of passenger counting by counting face to obtain passenger count, age and gender in a bus.

I will be using a few Movidius NCS stick to perform genderage net on the edge.

Problem I encountered is having the NCS redundantly counting the same face as different input. Hence, I try to figure out to use this blogpost’s code to solve the problem. Could it be possible to extract the face (in the rectangular box perhaps?) according to the ID in your code and proceed my application based on the individual ID box to reach my end goal?

Thank you so much.

Adrian Rosebrock December 4, 2018 at 9:42 am #

REPLY

Hey Rick — do you have any example images of the NCS redundantly counting the same face? I’d like to see what’s going on before providing a suggestion.

Rick December 4, 2018 at 9:53 am #

REPLY

Good day Adrian,

In the ncappzoo/apps/ gender_age_lbp (https://github.com/movidius/ncappzoo/tree/master/caffe/GenderNet) I made small edit on the C++ in cpp folder titled gender_age_lbp.cpp on line 643 I’ve output the variable “rectangle_text” into a csv folder for me to do the analysis on passengers’ gender and age.

Once I run the application, it will redundantly giving me inference on the same faces which is not what I want for my project.

It would be good if I can extract the face according to their ID# (so it is not redundant) in this blog as there’s a NCS python code that can do inference on image stored at local storage. (in ncappzoo/caffe/gendernet/run.py).

My question is a bit long. Sincerely appreciate for your time.

Thanks!

Adrian Rosebrock December 5, 2018 at 10:02 am #

REPLY

Hey Rick! Hats off to you for digging into the C++ code in GenderNet for Movidius. This isn’t actually something I’ve done, but it might make for a good future blog post — has anyone on the Movidius forums been helpful? Are you using both

the NCS v2 and the NCS API v2?

Ahmad December 5, 2018 at 3:26 am #

REPLY

Hi Adrian

Your work and the explanations are really awesome. As an extension to this post can you tell me how can we detect an object which is there in the frame for more than specified number of frames at the same location. For example if you stay in the same position for 30 frames then we need to put a text that says Andrian is stayed in that region since one second without moving.

How can we do that.

Adrian Rosebrock December 6, 2018 at 9:45 am #

REPLY

You can do that via object tracking. This tutorial will help you get started.

qwe December 10, 2018 at 4:23 pm #

REPLY

I just figure out that I don't need this implementation, beacuse trackers had already been wrote and you can find them in opencv_contrib/tracking, so after finishing this tutorail i feel like wasted my time, one interesting think was to getting min value form euclides comparison arrays

Adrian Rosebrock December 11, 2018 at 12:41 pm #

REPLY

If you're interested in the object trackers built into OpenCV then I would suggest following this tutorial.

qwe December 10, 2018 at 5:29 pm #

REPLY

Thanks for your work Adrian. You are awesome!

Adrian Rosebrock December 11, 2018 at 12:40 pm #

REPLY

Thank you!

TwelveW December 12, 2018 at 3:58 pm #

REPLY

Hi, Do you have any tips on improving the frame rates if I implement this on a Raspberry Pi?

Adrian Rosebrock December 13, 2018 at 9:00 am #

REPLY

Much of the computation is spent on the face detector. Try using a Haar cascade.

Rick December 17, 2018 at 2:59 am #

REPLY

Hi Adrian, its me again.

I would love to know how can you implement this object tracker like what you did on laptop webcam into Raspberry Pi Zero W. I have change line 34 into vs = VideoStream(usePiCamera=True).start() [as stated in https://www.pyimagesearch.com/2018/06/25/raspberry-pi-face-recognition/%5D to use my Raspberry Pi Camera instead of webcam. However it couldn't work directly as desired.

Since I am accessing to my RPi Zero W using SSH, I could not know what error I have into while I'm running the python code of object_tracker.py. It only shows "Illegal instruction" in the terminal.

So, I tried printing a simple sentence of every line to see which line that has error have from my finding its at line 41 (frame = imutils.resize(frame, width=400)) that fails to print my simple sentence after this line.

However I could not figure out what's the error of this line.

Could you help about it? Or did I left out something else?

Thanking you in advance!

-Rick

Adrian Rosebrock December 18, 2018 at 9:02 am #

REPLY

For whatever reason the Raspberry Pi Zero + OpenCV hates the default inter=cv2.INTER_AREA to imutils.resize.

Change it to:

frame = imutils.resize(frame, width=400, inter=cv2.INTER_NEAREST)

Rick December 18, 2018 at 11:39 pm #

REPLY

Hi Adrian,

The code works, however, is there any other possible to solution to stop the camera from streaming? As I am doing SSH to the RPi Zero and I could only stop the camera by CTRL + Z and the camera is still running (red LED still on).

As compared to webcam we have the frame window pops out and can be stopped by press 'Q'.

Appreciate for your reply.

Thank you!

-Rick

Rick December 19, 2018 at 12:45 am #

REPLY

Hi Adrian,

Sorry for asking such question. It is resolved by just pressing CTRL + C. Though some patience is needed for it to stop.

Adrian Rosebrock December 19, 2018 at 1:51 pm #

Out of curiosity, were you performing X11 forwarding via SSH?

Ivan December 30, 2018 at 12:51 am #

REPLY

Hey Adrian,
thx for the great article.
quick question for the distance calculation/ordering.
Am I correct in thinking that if I do the following:

zip(np.arange(D.shape[0]),D.argmin(axis=1))

the final result would be the same except that the centroids would not be ordered by distance.
In this case I assume the ordering is not really important.

git-scientist December 30, 2018 at 10:13 pm #

REPLY

Hi Adrian, thanks once again for your awesomely explained tutorials. I would like to ask about some implementable tips that you can give away. Since this project utilizes CPU, in videos or (in real time scenarios), fps is less than normally expected – 30 fps. And every time the program is run, fps value is different as well. Therefore, could you give your thoughts on how one can solve this issue? Is it possible to set fixed frame rate? If yes, then wouldn't it be erroneous when program can't receive fixed number of frames due to PC

possible to set fixed frame rate? If yes, then wouldn't it be erroneous when program can't receive fixed number of frames due to PC specs?

Maybe multi-threading is possibly another workaround?! What do you think?
Massive thanks for your time!

Alexis Tonog January 6, 2019 at 9:56 am #

REPLY

Hello Adrian, thanks for this tutorial, but i get an error like this:
object_tracker.py: error: the following arguments are required: -p/–prototxt, -m/–model

i'm using Windows 7 and python 3.7. what should i do?

i hope you reply soon, thank would be very helpful. thankyou

Adrian Rosebrock January 8, 2019 at 7:00 am #

REPLY

Your error is coming from not understanding how command line arguments work. Read this tutorial and you will be able to successfully execute the code.

Raman Kumar January 19, 2019 at 11:55 am #

REPLY

Hello Adrian,

AttributeError: 'module' object has no attribute 'dnn'

I'm facing this error whenever I execute

python object_tracker.py –prototxt deploy.prototxt \
> –model res10_300x300_ssd_iter_140000.caffemodel

I hope you will reply as soon as possible.Thanks for the help.

Adrian Rosebrock January 22, 2019 at 9:36 am #

REPLY

It sounds like you're using an old version of OpenCV. You need at least OpenCV 3.3 for this tutorial. Make sure you follow one of my OpenCV install guides.

Ron January 30, 2019 at 9:19 am #

REPLY

Sorry
can not understand whyyyyyyyyyyyyy
–
Can't open "deploy.prototxt" in function 'cv::dnn::ReadProtoFromTextFile'

Adrian Rosebrock February 1, 2019 at 7:10 am #

REPLY

Your path to the input "deploy.prototxt" file is incorrect. Double-check your input file path.

Panos February 5, 2019 at 6:14 am #

REPLY

Any idea on why ssd with resnet from the TensorFlow Model Zoo is so much slower than the caffe detector you are applying here ?
Is it a different model ?

REPLY

Adrian Rosebrock February 5, 2019 at 9:14 am #

Yes, it's a different model. See this tutorial for more information on the face detector.

REPLY

Matthew February 6, 2019 at 2:51 am #

REPLY

"VIDEOIO ERROR: V4L: can't open camera by index 0"

Has anyone else run into this issue? I'm just using a picamera plugged into it's corresponding port… nothing crazy but for some reason it's not reading it for VideoStream????

Adrian Rosebrock February 7, 2019 at 7:12 am #

REPLY

You're using a Raspberry Pi camera? If so, you need to update the VideoStream call:

```
vs = VideoStream(usePiCamera=True).start()
```

Rodger February 11, 2019 at 12:46 pm #

REPLY

I found a bug. In centroidtracker.py line 48 you mutate an array while iterating over it. I changed your code to track what needed to be removed then when done with the for loop I remove the correct items. After that I was able to run completely through a Friends episode.

coffee code March 15, 2019 at 4:49 pm #

REPLY

im getting the same error. how did you solve that?

Mahesh February 13, 2019 at 10:15 am #

REPLY

i wanted to do it using tensor flow with faster rccn model can you please help me how i can do the same process using tensor flow frozen inference graph which i produced for the face detection after training. i am getting pretty good result so i want to implement this to tensorflow.

Malhar February 19, 2019 at 5:17 am #

REPLY

Hi Adrian,
Your work and the explanations are really awesome. As an extension to this post can you help me out with saving the video as a .avi or .mp4 file.

TIA

Adrian Rosebrock February 20, 2019 at 12:20 pm #

REPLY

You'll want to refer to this tutorial first followed by this one.

Rick February 19, 2019 at 10:35 pm #

REPLY

Hi Adrian, Good day!

I want to extract only the faces of a person using Raspberry and Movidius.

Is there any way to only extract the ROI of the tracked image using Movidius on Raspberry Pi? I have followed "Real-time object detection on the Raspberry Pi with the Movidius NCS" and that can provide very robust FPS on Raspberry but doesn't give me the ROI I want as this blog as that final output is keep detecting same faces as multiple result.

Cheers.

Rick

Adrian Rosebrock February 20, 2019 at 12:05 pm #

REPLY

Are you asking how to perform face *detection* with the Movidius NCS? If so, I'll be covering that exact topic in my upcoming Computer Vision + Raspberry Pi book. Stay tuned!

Rick February 20, 2019 at 9:58 pm #

REPLY

Hey Adrian,

Yes face detection and taking snapshot of the bounding box (ROI) with only single output for a same person, just like how it can be done through this blog 'Simple object tracking with OpenCV'.

Unlike workspace/ncappzoo/apps/security-cam/security-cam.py, it take multiple snapshots of same person.

My goal is to do passenger counting by extracting the face only as ROI and not multiple snapshot of same person, and then feed these snapshot into gendernet and agenet for age and gender classification.

Thanks Adrian.

Adrian Rosebrock February 22, 2019 at 6:42 am #

REPLY

Hey Rick — I don't have any examples of running a face detector using the Movidius NCS on the PyImageSearch blog but I've made a note to cover it in my upcoming Computer Vision + Raspberry Pi book. Thank you for the suggestion, I'll make sure it's covered!

Intisar March 2, 2019 at 6:04 am #

REPLY

Is it possible to optimize this code by running object detection for a certain interval of frame instead of running the object detector for every frame?
By the way, thank you so much for such a helpful tutorial.

Adrian Rosebrock March 5, 2019 at 9:02 am #

REPLY

Yes, absolutely. See this tutorial as an example of doing exactly that.

Cory March 6, 2019 at 10:59 am #

REPLY

Hi, is there a maximum distance which the input and previous centroids will not associate? Or as long as there is one input and one previous centroid will it always update vs increment the disappeared counter? Thank you for the nice tutorial!

Adrian Rosebrock March 8, 2019 at 5:35 am #

REPLY

This particular implementation does not have a max distance for association but the one I'm including in my upcoming Computer Vision + Raspberry Pi book *will* have a max distance.

Hamdi March 8, 2019 at 3:24 am #

REPLY

Hi Adrian !

Your code finds 10 id in one frame but only ı am staying at front the webcam

Eloy March 22, 2019 at 4:57 pm #

REPLY

Hi Adrian! thanks a lot for your dedication.

How can I implement this object tracking but using a contour detection instead of deeplearning?. I want to track a green rectangle.

Thanks again

Adrian Rosebrock March 27, 2019 at 9:28 am #

REPLY

Whether or not you can use contour detection is dependent on your ability to segment the object from the image using contours. You can use contour tracking here though, just pass in the bounding box of each individual contour.

Nitya March 29, 2019 at 10:10 am #

REPLY

How can I use Haar cascade classifier instead of .prototxt and .caffemodel?

Adrian Rosebrock April 2, 2019 at 6:13 am #

REPLY

Have you used a Haar cascade before? If not, refer to Practical Python and OpenCV where I teach you how to use them. Once you understand how to use Haar cascades you'll be able to swap them in place of the deep learning models.

ray April 3, 2019 at 11:07 am #

REPLY

Hi Adrian

how can i used the tensorflow models in this code? hoping for your response

Adrian Rosebrock April 4, 2019 at 1:18 pm #

REPLY

TensorFlow models to do what, specifically?

famunir April 3, 2019 at 8:26 am #

REPLY

Thank you for this great tutorial!

However, I have a little issue. Once an object is marked for tracking and I start the tracking algorithm it works fine. But after few frames, if I want to pause the video and select another object for tracking in the same video, I am unable to track this newly marked object of interest. What can be the issue here? Can you please provide some explanation?

This is some issue like when tracking is started, I cannot select another object. I have to stop the algorithm altogether to track the the aforementioned object.

Thang Nguyen April 29, 2019 at 5:30 pm #

REPLY

Thank you for your very helpful article.
I would like to ask about your algorithm. In step 2, if the object on the top left [ID: 1] moves out of boundary, will your algorithm consider that the new object at the bottom is the object ID: 1? Because the distance from old 1 to new is shorter than the distance from old 1 to new 2.

Colin May 2, 2019 at 4:43 pm #

REPLY

This is brilliant, and so simple when explained like this. So how do you output the result of the tracking video on the screen to an MP4 file or similar? That would be great

Adrian Rosebrock May 8, 2019 at 1:40 pm #

REPLY

You can use the cv2.VideoWriter function.

LGenzelis May 6, 2019 at 5:04 pm #

REPLY

I've been spending the last few days reading one tutorial after the other (all yours, I mean). I'm absolutely hooked. Thanks a lot for your marvelous work Adrian!

Adrian Rosebrock May 8, 2019 at 1:02 pm #

REPLY

Thanks, I'm so happy that you're enjoying them!

BryanValdez May 13, 2019 at 2:39 pm #

REPLY

First of all, great tutorials! and blog, well my question is, if there the ID number can be restarted?, like if the object (face) with the ID0 disappear from the frame and then appear again using the ID0… that can be done?

Adrian Rosebrock May 15, 2019 at 2:51 pm #

REPLY

That would be called face re-identification. It's an entirely different project. I don't have any tutorials on it though.

Ayush Goyal May 29, 2019 at 6:36 am #

REPLY

Is there any guide for the re-identification? I am in need for this.
And thanks a lot for your great work.

Adrian Rosebrock May 30, 2019 at 9:06 am #

REPLY

Sorry, I don't have any tutorials for person re-identification at this time.

Ahsan Raza May 14, 2019 at 1:45 am #

REPLY

Kindly can you elaborate how to track a single object and should not look for another object until the first object is out of frame.

Ajinkya May 14, 2019 at 2:31 am #

REPLY

Hi Adrian,

I really appreciate your post, I am stuck in a small loop I hope you can help me out:

My problem is: I am creating object detector for detecting credit cards from images. The catch here is that the cards have plethora of variety eg. Different colours, different patterns, few also have hologram on them, also the background in which these cards are placed varies to great extent (eg: some cards are placed on notebooks, some on office tabletops, while some on car seats). The only thing that remains constant among these cards is their shape. Is it possible to use a custom trained object detector and use your code to track these cards in images.

Adrian Rosebrock May 15, 2019 at 2:42 pm #

REPLY

I would recommend you train your own custom object detector. See Deep Learning for Computer Vision with Python where I provide detailed instructions + code on training your own custom object detectors.

Ahsan May 16, 2019 at 4:26 pm #

REPLY

Hi Adrian,
How to track a single object and should not look for another object until the first object is out of frame.

Adrian Rosebrock May 23, 2019 at 10:18 am #

REPLY

You would need to modify the code a bit, but basically you would detect the first object and then set a boolean indicating that an object is currently “active”. Monitor that frame and then when it reaches the boundaries of the frame, which you can check my examining the (x, y)-coordinates of the frame, you indicate that it’s “inactive” and attempt to detect a new object.

Ahsan May 23, 2019 at 3:31 pm #

REPLY

Thanks a lot, I will let you know if I complete it successfully.

Ahsan May 29, 2019 at 2:29 pm #

REPLY

Hello Adrian,

First of all thanks for your blogs and all this support and efforts for young researchers and developers seeking knowledge of Image Processing and Computer Vision. Thumbs Up!!!

I want to detect a single face and track it throughout the frames. If the face exist I want to stop seeking for another detection. Basically I want single face detection and tracking. If that face or ID “0” doesn’t exist it should then look for another face.

I tried to make changes in the code but I am unable to do it. Kindly help me out.

Adrian Rosebrock May 30, 2019 at 9:03 am #

REPLY

Hey Ahsan — I’d be happy to help but I first suggest you work through the PyImageSearch Gurus course to help you learn the basics first. From there I’d be happy to help you make this project a success.

Ahmetfy May 20, 2019 at 8:10 pm #

REPLY

Can I get your help on something?
I’m working on the object tracking system. I have a problem.
I succeeded in naming objects, but my project gives different names to the same object. For example, objects0, person1 in frame0, person1, person0 in frame1.

Adrian Rosebrock May 23, 2019 at 10:30 am #

REPLY

I’m not sure I understand your question. How are you differentiating between what is a person vs. and object? Are you using some sort of object detector?

Aman June 4, 2019 at 10:30 am #

REPLY

I want detect cars using HOG+SVM technique and I want to track the cars based on what you are learning us here. I implement it on raspberry pi 3. Do you it will be very slow? Do you suggest me to never try it? My question is related to what you state in drawbacks.

REPLY

Adrian Rosebrock June 6, 2019 at 6:51 am #

You basically would like to build a traffic counter then? If so, I'm covering that exact project inside Raspberry Pi for Computer Vision.

REPLY

Aman June 8, 2019 at 7:08 am #

ya, I need to build traffic counter. I have already done it by using HOG+SVM and some way of tracking algorithm. It works well on my pc but it is very slow on raspberry pi. I will look forward for your project project inside Raspberry Pi for Computer Vision. But until then suggest me the way to do it. may I try it using haar cascade? thanks for your reply!

REPLY

Suraj June 6, 2019 at 1:28 am #

The box variable has 4 values. In the comments if the code you say they are the X,Y coordinates of the rectangle. But Can you tell me exactly what are those 4 values represent ?

REPLY

Adrian Rosebrock June 6, 2019 at 6:39 am #

The four variables represent the starting and ending coordinates of the bounding box (i.e., the bounding box that encloses the face).

REPLY

Suraj June 6, 2019 at 7:37 pm #

So the values are X,Y coordinates of the top left point of the rectangle and the X,Y coordinates of the bottom right point respectively. Right ?

REPLY

Adrian Rosebrock June 12, 2019 at 2:04 pm #

Yes, that is correct.

REPLY

BryanValdez June 10, 2019 at 1:00 pm #

How can i run this code in a Raspberry PI 3?,
will be very slow?,
can i run the code for 2 cameras? due to the CPU consumption…

REPLY

Adrian Rosebrock June 12, 2019 at 1:36 pm #

This code can run on a RPi.

REPLY

Waylon Flinn July 12, 2019 at 3:02 pm #

Thanks for this. It helps to get a feel for what goes into an object tracking system.

One point. The centroid tracking algorithm is slightly improved if you reverse the columns and rows in your calculations. Here's what I mean (changes for lines 88-93):

cols = D.min(axis=0).argsort()

cols = D.min(axis=0).argsort()
rows = D.argmin(axis=0)[cols]

This assigns each *input* centroid to it's closest *tracked* centroid, instead of vice-versa. This avoids failing to assign a tracked centroid when two tracked centroids have the same input centroid as their closest pairing.

This method also requires a few tweaks to the code that follows.

REPLY

Adrian Rosebrock July 25, 2019 at 10:10 am #

Thanks for sharing, Waylon!

REPLY

Nuriel July 16, 2019 at 6:01 pm #

Very nice insight to first experiences with tracking!

I am just asking myself how can I evaluate the performance of this tracker? and in general how can I evaluate performance of tracking algorithms?

for object detection it is something like iou for bounding boxes but here..? what should we take?

REPLY

Newbie July 20, 2019 at 2:47 am #

I have the coordinates of the rectangle created and i want to define a counter where only those centroids that are inside that rectangle are counted. But i am not able to figure out how should i do that.

How can i check if centroid is inside the rectangle defined by me?
Please help.

REPLY

Joseph August 27, 2019 at 10:50 am #

Hi Adrian,

I wanted to know if it's possible to implement this on a set of images of a scene (in my case droplets on a plate) taken over intervals of time. I would like to have the same IDs for the droplets across the image set (also considering the some droplets evaporate with time)

REPLY

Joseph August 30, 2019 at 10:21 am #

Ciao Adrian,

I tried implementing this on a sequence of images in a for loop. The contour IDs don't seem to refresh for every image in the sequence rather the IDs add up for every image in the sequence.
I reckoned the centroid tracker script might need a change but I'm not sure where. Can you please help me with the same?

Thank You.

Joseph

REPLY

Adrian Rosebrock September 5, 2019 at 10:35 am #

Hey Joseph, I'm happy to provide the code and tutorials for free. If you need my personal help you should join the PyImageSearch Gurus course so I can help you out.

REPLY

Rahul September 7, 2019 at 10:09 am #

Hey Adrian , the tracker worked well normally. I have a query though, how do i retrieve the bounding box co-ordinates for each object ID? Currently we are getting the object Id with the centroid co-ordinates , is there a way to get the rectangle co-ordinates of the object?

Adrian Rosebrock September 12, 2019 at 12:00 pm #

REPLY

You can associate the centroid coordinates together with the bounding box. Or you could modify the code to return a set of indexes into your list of bounding boxes.

Gaurav September 16, 2019 at 3:17 am #

REPLY

I am using Background subtraction for object detection and for updating ids of contour i use centroidtracker.py but it not working properly ,in every frame it provide new ids to the contour can you please help me out

Adrian Rosebrock September 19, 2019 at 10:04 am #

REPLY

Hey Gaurav — please see this entry in my FAQ.

jan October 5, 2019 at 10:49 pm #

REPLY

Thanks for this Adrian!!! I need to found the tracking time for each person, can you please help me out!

Adrian Rosebrock October 10, 2019 at 10:23 am #

REPLY

You can use either the “time” or “datetime” Python library.

JC October 9, 2019 at 9:58 am #

REPLY

Hi,
I don’t understand why you are checking if “the number of object centroids is EQUAL or greater than the number of input centroids” and not just if “the number of object centroids is greater than the number of input centroids”.

Someone can explain that to me ?
Thanks !

Before you leave a comment...

Hey, Adrian here, author of the PyImageSearch blog. I'd love to hear from you, but before you submit a comment, **please follow these guidelines:**

- **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.
- **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.

Leave a Reply

Name (required)

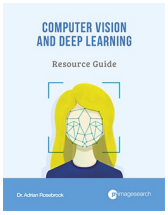
Email (will not be published) (required)

Website

SUBMIT COMMENT

Search...

Resource Guide (it's totally free).



Get your **FREE 17 page Computer Vision, OpenCV, and Deep Learning Resource Guide PDF.** Inside you'll find my hand-picked tutorials, books, courses, and libraries to help you master CV and DL.

Download for Free!

Raspberry Pi for Computer Vision

You can teach your **Raspberry Pi** to "see" using **Computer Vision**, **Deep Learning**, and **OpenCV**. **Let me show you how.**

CLICK HERE TO LEARN MORE

Deep Learning for Computer Vision with Python Book — OUT NOW!

You're interested in deep learning and computer vision, *but you don't know how to get started.* Let me help. **My new book will teach you all you need to know about deep learning.**

CLICK HERE TO MASTER DEEP LEARNING

You can detect faces in images & video.

Are you interested in **detecting faces in images & video?** But **tired of Googling for tutorials** that *never work*? Then let me help! I guarantee that my new book will turn you into a **face detection ninja** by the end of this weekend. Click here to give it a shot yourself.

CLICK HERE TO MASTER FACE DETECTION

PyImageSearch Gurus: NOW ENROLLING!

The PyImageSearch Gurus course is *now enrolling!* Inside the course you'll learn how to perform:

- Automatic License Plate Recognition (ANPR)
- Deep Learning
- Face Recognition
- *and much more!*

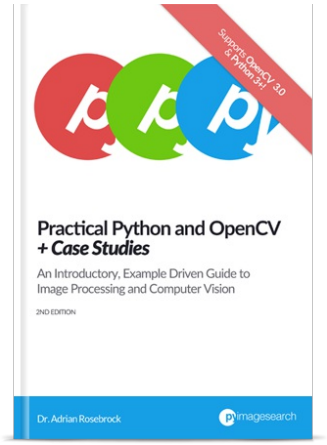
Click the button below to learn more about the course, take a tour, and get 10 (FREE) sample lessons .

TAKE A TOUR & GET 10 (FREE) LESSONS

Hello! I'm Adrian Rosebrock.

I'm Ph.D and entrepreneur who has spent his entire adult life studying Computer Vision and Deep Learning. I'm here to help you master CV, DL, and OpenCV. Learn More

Learn computer vision in a single weekend.



Want to learn computer vision & OpenCV? I can teach you in a **single weekend**. I know. It sounds crazy, but it's no joke. My new book is your **guaranteed, quick-start guide** to becoming an OpenCV Ninja. So why not give it a try? Click here to become a computer vision ninja.

CLICK HERE TO BECOME AN OPENCV NINJA

Subscribe via RSS

Never miss a post! Subscribe to the PyImageSearch RSS Feed and keep up to date with my image search engine tutorials, tips, and tricks

POPULAR

Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi
SEPTEMBER 4, 2017

Install guide: Raspberry Pi 3 + Raspbian Jessie + OpenCV 3
APRIL 18, 2016

Face recognition with OpenCV, Python, and deep learning
JUNE 18, 2018

Home surveillance and motion detection with the Raspberry Pi, Python, OpenCV, and Dropbox
JUNE 1, 2015

Install OpenCV and Python on your Raspberry Pi 2 and B+
FEBRUARY 23, 2015

Real-time object detection with deep learning and OpenCV
SEPTEMBER 18, 2017

Ubuntu 16.04: How to install OpenCV
OCTOBER 24, 2016

Find me on **Twitter**, **Facebook**, and **LinkedIn**.
Privacy Policy
© 2019 PyImageSearch. All Rights Reserved.