Observations on OpenCV Trackers

Since OpenCV is the premier computer vision tool used for image processing across fields the tracker applications provided by this library are commonly used on a variety of applications. We believe that it will be useful to the practitioner to get some insight on the characteristics of OpenCV traker operation. Additionally, the information below allows to summarize and emphasize different scenarios faced on all tracking tasks whether or not OpenCV tracking technology is used. It is assumed that the tracker will work in conjunction with a deep learning object detector.

Once the tracker is initialized with one of the bounding boxes that correspond to the vehicle of interest, it is probable that a situation will occur which will make the tracker jump outside the vehicle (it could jump to another vehicle or to something in the background, which sometimes dramatically changes the tracker's bounding box size). This situation occurs due to tracker defects and artifacts that confuse patches in the image with the previous captured image being tracked. Since the scene is dynamic and the vehicle changes scaling continuously (and brightness changes as well) sometimes a random patch on the image has greater closeness/similarity (according to the model used by the tracker) with a previously captured image of the vehicle than with the new image of the vehicle in the current frame. Depending on the robustness of the feature extraction and model being used by the tracker this situation may occur with more or less frequency. Therefore repositioning of the tracker is required. Below we refer to detection-bounding-box to the bounding box obtained by neural network detection and tracker-bounding-box as the box generated by the tracker’s processing. Below are two situations you may encounter. Additionally, we have found that the "ok" output provided by OpenCV tracker is not always reliable to determine tracker failure (the tracker says that everything is ok and still and invalid jump happened) and thus some of the mechanisms mentioned below are needed.

1) Tracker Fails - Repositioning the Tracker:

Something that worked in the past was to keep a record of the last known X and Y coordinates of the detection-bounding-box’s centroid that is closest to the tracker’s current position (which is the X-Y coordinates of the tracker-bounding-box’s centroid). In this case the closest detection-bounding-box is what we call the one "associated" with the tracker. On initialization we are sure that the tracker position is the same as the detection-bounding-box position used for initialization. If in the next frame the tracker’s new position is too far away from the previously known detection-bounding-box associated with the tracker then this triggers a situation where an invalid jump may have happened and we need to reposition the tracker. The OpenCV Tracker needs to be reinitialized using the coordinates of the detection-bounding-box on the current frame that is closest to the last known position of the tracker. The invalid jump may happen towards some random patch on the image or possibly towards another detected vehicle in the image. The ability to determine if the jump was valid or not depends on assessing how far the distance is between current tracker position and last associated detection-bounding-box position.

It could be possible that under the situation described in the paragraph above when the tracker jumps far away, the neural network detection also fails and the detection-bounding-box that corresponds to the vehicle being tracked is non-existent. This poses a problem because there is no valid detection-bounding-box for repositioning. Since the dynamics of vehicle motion on roads can not accommodate large displacements (particularly between image frames which are 1/30 seconds apart) the distance between detection-bounding-boxes that correspond to the same vehicle on different frames can not be that far apart. Considering this last observation when the neural network detection fails a wait-period that could last a few frames could be implemented. The objective of such waiting period is to wait for the neural network detector to successfully detect the vehicle that was being previously tracked. To correctly identify the vehicle as the neural network detector re-engages into the detection a radius around the last known vehicle position needs to be considered (the last known vehicle position is the position of the last detector-bounding-box that was associated with the tracker before disappearing). All this requires some level of calibration (calibration regarding the radius for re-engagement of vehicle detection, and wait period which could be 3 o 4 frames). Once the detection re-engages with the generation of a detector-bounding-box that falls within the radius then the tracker can be re-initialized to this bounding box so tracker repositioning is accomplished. If the wait period expired then the tracker needs to be shutdown and a new vehicle needs to be selected to initiate a new tracking.

2) Neural Network Detection Fails:

Even in a situation when there is no abrupt jump of the tracker-bounding-box the neural network detector could fail and the detector-bounding-box vanishes. This is precisely one of the justifications for the existence of the tracker. In this case the

tracker uses its own model to keep moving from frame to frame and hopefully following the associated vehicle. How do we know that the associated bounding-box vanished?,
We know this by performing the following measurements:

> a) We measure the distance between the last known position of the detection-bounding-box associated with the tracker (which is a position obtained on the previous frame) and the position of the detection-bounding-box that is closest to the tracker in the current frame. If this distance is too large this triggers a condition where the neural network failed to the detect the associated vehicle.
> b) It is possible that the neural network failed to detect any vehicle (zero bounding boxes in the frame). In which case we can assume that the neural network failed as well.
> c) We measure the distance between the current tracker position and the last known position of the detection-bounding-box associated with the tracker. This is an alternate measure to the one mentioned in a) and it should be slightly different than the distance measured in a). If the distance is too large then this could trigger a failure of the neural network.

It is also plausible that the vehicle just went out of view and thus the detection vanished thus there is no neural network failure. In this case some record needs to be maintained about the current positioning and relative trajectory of the associated vehicle within the frame in order to detect that the vehicle is going "out of view" and thus a vanishing is likely to happen. If the associated vehicle's trajectory shows that the associated detection-bounding-box is moving towards one of the margins of the frame (for a leading vehicle) then this vehicle is going out-of-view. Therefore the tracker needs to be shutdown and we need to look for a new vehicle if it exists.

Therefore the strategy is to always keep a record of the last known X and Y coordinates of the detection-bounding-box's centroid that is closest to the tracker's current position. If the neural detector fails we have two references to work with. We have the position of the last detector-bounding-box associated with the tracker and we have the current tracker position. Similarly to what was described above, in this case we can set a wait period and a radius for re-engagement of the neural network detector. In this case a detector-bounding-box that falls within the radius of the last known associated detector-bounding-box provides the new bounding-box to be designated as the one associated with the tracker. In this case just to corroborate the designation: the current tracker positioning should also be "close" to the new associated bounding box.

It may seem redundant to look for a new associated bounding box since apparently the tracker is already following the vehicle. However the tracker could also fail and thus the associated detection-bounding-box is needed to overcome this failure (by repositioning). Therefore tracker and neural network work in a bootstrap manner in which one relies on the other when there is a failure. The situation where both fail simultaneous is considered in the third paragraph above.

In OpenCV when you need to reposition the tracker it is necessary to actually create the tracker again to re-initialize the tracker. Below are two lines in python that accomplish this.

```
tracker = cv2.TrackerTLD_create()
ok = tracker.init(frame, bbox)
```

Here bbox is the rectangle corresponding to the bounding box to be used for initialization. At least in python we have found that just by doing tracker.init(frame, bbox) is not enough to get the tracker initialized on a new bounding box, Thus the cv2.TrackerTLD_create() is also needed. Here bbox is the array (in python) that is used to initialize and to update the OpenCV tracker position:

The information below provides a reference with respect to the output of the tracker.

bbox[0] is the X coordinate of the top left corner of the bouding box
bbox[1] is the Y coordinate of the top left corner of the bouding box
bbox[2] is the width of the bounding box
bbox[3] is the height of the bounding box