George Mason University

Real Time YOLOv1 Object Detection

With Kalman Filter Stabilization

Benjamin Clark

May 10, 2022

Contents

Abstract

The purpose of this project was to take video input frame by frame and classify all the objects in the each frame with a YOLOv1 network architecture while stabilizing the boxes across time with Kalman Filtering and removing sporadic, short lived bounding boxes to make the final output smoother and more predictable

1 Introduction

While YOLO style networks have already been used on video input [1], and there do exist already neural networks such as Tiny YOLO [2] which have the express purpose of being light weight and suitable for live video input output, and at an impressive 442% speed up from the normal YOLO architecture, smaller networks like these can lose a lot of their accuracy in exchange for their speed [2]. Tiny YOLO in particular was measured to score mAP 23.7% where as the normal YOLO network was able to get mAP of about 51% [2]. After running the the YOLOv1 network on video input, it became clear that some filtering or cleaning up of the output bounding boxes was needed, as the boxes from one frame to the next were often sporadic. Sometimes boxes would shift drastically around the objects they surround, making the video as a whole harder interpret. There are also some instances of objects which might be far away or at weird angles where it was harder for the network to consistently recognize those objects, causing the associating bounding box to flash in and out of existence as the network cannot always capture the object in every single frame. The raw output from live video thus is not very useful as input to other systems which might rely on more consistent object tracking. Systems which might suffer heavily from chaotic object detection output might include self driving cars, which given bad input could cost lives, or cctv monitoring, where having clean facial detection would be necessary for security applications.

The goal of this project thus is to provide some mechanism by which the output of object detection networks can produce cleaner and more consistent output for these more sensitive types of systems where accuracy is of upmost importance.

2 Approach

My approach was to keep the underlying YOLO network the same and then add operations onto the output which would be able to fix artifacts which were temporal in nature. The methodology behind this is that the network should take up the task of training well enough to preform well on individual frames, and then receive aid from this new algorithm at its output to fix artifacts that persist over time.

I decided that my implementation should have two main steps following the same kind of implementation as Jeremy Cohens approach in stabilizing YOLOv3 output [3]. The first step would be to use some data structure to mark objects and track them across time. This would allow for the removal of bounding boxes which are too sporadic, and also aid in cleaning up areas of the video where there are many overlapping boxes which might actually be referring to the same image. Tracking objects this way would also help in solving the problem where the network can identify hard to detect objects in only every other frame by allowing a bounding box to persist for a couple of frames and only then be removed if it has not received new update after too many frames.

The second step was to introduce a filtering algorithm to make the bounding boxes for a particular object more consistent and less sporadic over time. My first idea was to go with some kind of running average or exponential decay of past inputs to produce more suitable predictions for where the corners of the bounding box would be, but instead decided that implementing a full Kalman Filter would likely be much more accurate to the ground truth of what the bounding box should look like [4, 5]. The results they achieved did indeed improve the quality of bounding box stability over time so I went ahead with the Kalman Filter stabilization technique.

2.1 Object Tracking

The first part of the filtering process is to setup a system for tracking objects over time. The input for this phase is the raw bounding box predictions from the YOLO network, and the output is a refined set of bounding boxes which can better predict which objects the network detected are real and which are random artifacts. The method I follow is the same idea as the The Hungarian Algorithm [3]. The process starts by doing a pass over all of the already known objects, and matches the incoming candidate boxes with these already existing objects which have been already determined to most likely be real objects. Candidates are matched with existing objects based on which object they have the highest intersection over union (IOU) with. Once a match is made, the candidate box is passed as the next updating input to the Kalman Filter for that object so that later a better prediction for the ground truth box can be determined. For all the objects which appear to be novel, and do not match with any of the loges existing boxes, a new structure to track that object is created and marked as a new object. New objects are assumed to be false and unreliable until they can prove themselves otherwise by claiming above some number of candidate objects in the future. Only after a new object has claimed said number of objects can it graduate to real object status and be sent as valid output from the function. Along with a prerequisite amount of time needed to be considered a live bounding box, there is also a limited amount of time that a given box can go without updates before it is considered dead. At the end of every update, any live bounding boxes which have not received an update for some given amount of frames then it is reaped.

2.2 Kalman Filtering

Kalman Filter usage specifics and equations came from sources 4 and 5. Each of the bounding boxes being tracked over time has two of its own associated Kalman Filter which attempts to predict bounding box coordinates closer to ground truth values. In the Object tracking phase, if a new object is being tracked, then the initial state of the Kalman Filters is set to the given coordinates from the candidate in a matrix as follows:

$$\hat{\boldsymbol{x}}_{0|0} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$$

The Kalman filters then are updated whenever the object tracking phase determines that a given candidate box is suitable for a given object it is tracking, the corners of the candidate box are then passed in to update the Kalman Filter each respective corner. The for the Kalman Filter has a few main hyper-parameters that can be fine tuned for better predictions include the process noise covariance matrix denoted by $\boldsymbol{Q}_k$, and the measurement noise covariance matrix denoted as $\boldsymbol{R}_k$ which both, in this implementation, take the following form form:

$$\boldsymbol{Q}_k = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{yx} & \sigma_y^2 \end{bmatrix}$$

$$\boldsymbol{R}_k = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{yx} & \sigma_y^2 \end{bmatrix}$$

The process noise parameter $\boldsymbol{Q}_k$ is a description of the expected variance in the state of system and the covariance between the x and y coordinates, that is, how much we expect the output of the network in this case to naturally very randomly. The measurement noise parameter $\boldsymbol{R}_k$ on the other hand describes the variance to be expected between ground truth and the measurement read from the Kalman Filter. The values of these hytper-parameters will very depending on the network. Generally if a network is less accurate, then there should be more variance expected in the process noise, and the process noise parameters should be raised accordingly. Depending on the type of video footage that is being fed into the network, the process noise covariance can help get better bounding box values as well. For instance, if the footage is shaky like that of hand held camera, it is usually the case that if the x coordinate is changing, then the y coordinate will change some as well because the movement of objects in frame not be locked on a single x or y axis.

One other hyper-parameter that must be set is the posteriori estimate covariance matrix denoted as $\boldsymbol{P}_{k|k}$ which holds values for the variance expected from the state transition model. If $\boldsymbol{P}_{k|k}$

were set to the zero matrix then we are saying that this measurement taken is known to have no error and be the real ground truth. In practice, we can be sure the network has some idea of where the ground truth bounding box is but not without error, so variance of the state must be accounted for in the posteriori estimate covariance matrix:

$$\boldsymbol{P}_{k|k} = \begin{bmatrix} \sigma_x^2 & \sigma_{x\hat{x}} \\ \sigma_{\hat{x}x} & \sigma_{\hat{x}} \end{bmatrix}$$

The last thing to consider while designing a Kalman Filter is how to construct the state transition model which describes the evolution from state $\boldsymbol{x}_{k-1}$ to $\boldsymbol{x}_k$. The full, generalized state estimation model takes the following form:

$$\hat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{F}_k \hat{\boldsymbol{x}}_{k-1|k-1} + \boldsymbol{B}_k \boldsymbol{u}_k$$

Where $\boldsymbol{F}_k$ is the state transition matrix, and $\boldsymbol{B}_k$ and $\boldsymbol{u}_k$ are control inputs. For the state transition model $\boldsymbol{F}_k$, we want to construct a model which should ideally map $\hat{x}_{k-1}$ to $\hat{x}_k$. The system we need to model is how to the bounding box coordinates move over time. Unlike some systems such as predicting an objects path through space as it falls, where there is a clear pattern in its increasing velocity, there is no pattern by which all bounding boxes follow over time. For this reason, my underlying assumption in constructing the state transition matrix is that a bounding box at time t should at time $t+1$ ideally be almost the same, if not moved by some small amount from one time step to the next. The state transition matrix then is simply the identity matrix, which will capture this idea of bounding box states staying the same over time. In my implementation, I determined that the $\boldsymbol{B}_k \boldsymbol{u}_k$ term can be dropped for the same reason as the state transition matrix be left as the identity matrix, because we do not expect the bounding boxes to follow a particular pattern. If we expected the boxes in the footage to have a constant drift to them over time for example, then this term would be useful, however in the general case we cannot necessarily make this assumption.

The values of these hyper parameters will be dependant upon the type of video being analyzed, and the inherent inaccuracy of the network. The smaller and more light weight the network being used, generally the inaccuracy will be higher, and the covariance for noise would need to be adjusted. If the Kalman Filter is to be fine tuned for footage where the assumption can be made that everything in frame will be moving in a particular direction rather than for general purpose footage, then the $\boldsymbol{B}_k \boldsymbol{u}_k$ term can be used for that type of situation to make better predictions.

3 Results

The results looked promising, as the output video was less cluttered with bounding boxes that didn't seem to box anything novel, and the shifting of the bounding boxes over time was greatly reduced. Boxes which had previously flashed in and out of existence because of the networks inability to capture the object in every single frame became smoother and easy to follow over time.

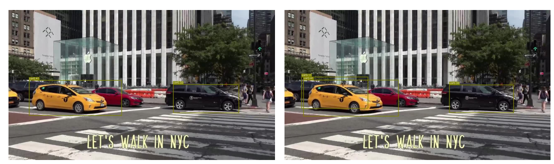

Figure 1: Removal of unnecessary boxes
In the raw video output, it can be seen that there is an extra bounding box for the yellow taxi. The bad bounding box from the network output is removed in the filtered version. (original footage from source 6)

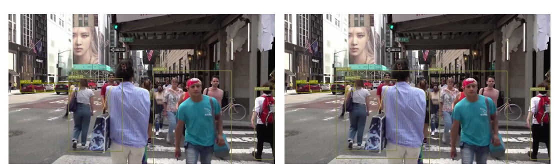

Figure 2: More removal of unnecessary boxes
Here again in the raw output the network puts a large box around a group of people instead of a single person. Since the box is an error, it is only visible in a few frames of the raw footage and is recognized in the filtered output as a bad box. (original footage from source 6)

4 Future Improvements

Some areas where I would try to improve upon would be my implementation of the Hungarian Algorithm. In some parts of the filtered video, some boxes which are justified in their existence are wrongly removed. While this is a bit of an issue however, it does do well at removing flickering boxes surrounding nothing which are just incorrect network outputs. Another area I would like to do more work on would be the initial states of the hyper-parameters for the Kalman Filter. Given enough time to fully label training data in a video, one can study the difference in network output from ground truth over time and see if there is some consistent error. If this were carried out, then the noise covariance matrices could be set to more accurately reflect the noise in the output of the network. During this experiment I did not have fully labeled video footage, and came to a set of values through trial and error which seemed to preform well.

References

[1] *Medium. 2022. Real-time Object Detection with YOLO, YOLOv2 and now YOLOv3. [online] Available at: ¡https://jonathan-hui.medium.com/real-time-object-detection-with-yolo-yolov2-28b1b93e2088¿ [Accessed 9 May 2022]..*

[2] *Rosebrock, A., 2022. YOLO and Tiny-YOLO object detection on the Raspberry Pi and Movidius NCS - PyImageSearch. [online] PyImageSearch. Available at: ¡https://pyimagesearch.com/2020/01/27/yolo-and-tiny-yolo-object-detection-on-the-raspberry-pi-and-movidius-ncs/¿ [Accessed 9 May 2022]..*

[3] *Medium. 2022. Computer Vision for tracking. [online] Available at: ¡https://thinkautonomous.medium.com/computer-vision-for-tracking-8220759eee85¿ [Accessed 9 May 2022]..*

[4] *Wikipidia. 2022. Kalman Filter. [online] Available at: ¡https://en.wikipedia.org/wiki/Kalman˙filter5¿ [Accessed 9 May 2022]..*

[5] *Medium. 2022. Computer Vision for tracking. [online] Available at: ¡https://medium.com/@jaems33/understanding-kalman-filters-with-python-2310e87b8f485¿ [Accessed 9 May 2022]..*

[6] *Youtube. 2021. Walking in New York City 4K. 5th Avenue. People, Cars and Street Sounds. [online] Available at:¡https://www.youtube.com/watch?v=yhkbg8p2Gtst=4s¿ [Accessed 9 May 2022]. .*