International Journal of Multidisciplinary Research and Growth Evaluation.

Social distance monitor using AI vision based on python

Sakthi Sridevi S [1*], Sinduja K [2], Vinothini G [3], Dr. S Subashree [4]
[1-3] Department of Computer Science & Engineering, E.G.S Pillay Engineering College, Nagpattinam, Tamil Nadu, India
[4] Ph.D., Computer Science & Engineering Department, E.G.S Pillay Engineering College, Nagpattinam, Tamil Nadu, India

* Corresponding Author: **Sakthi Sridevi S**

Article Info

ISSN (online): 2582-7138
Volume: 04
Issue: 03
May-June 2023
Received: 24-03-2023
Accepted: 18-04-2023
Page No: 22-30

Abstract
Social distancing is a recommended solution by the World Health Organization (WHO) to minimize the spread of COVID-19 in public places. The majority of governments and national health authorities have set the 2-meter physical distancing as a mandatory safety measure in shopping centers, schools and other covered areas. In this research, we develop a generic Deep Neural Network-Based model for automated people detection, tracking, and inter-people distances estimation in the crowd, using common CCTV security cameras. The proposed model includes a YOLO v4-based framework and inverse perspective mapping for accurate people detection and social distancing monitoring in challenging conditions, including people occlusion, partial visibility, and lighting variations. We identify high-risk zones with the highest possibility of virus spread and infections. This may help authorities to redesign the layout of a public place or to take precaution actions to mitigate high-risk zones. The efficiency of the proposed methodology is evaluated on the Oxford Town Centre dataset, with superior performance in terms of accuracy and speed compared to three state-of-the-art methods.

Keywords: Social distancing, AI -Artificial intelligence, Machine learning, Deep learning, YOLO v4, DNN, Coviid-19

1. Introduction

The novel generation of the corona virus disease
(COVID-19) was reported in late December 2019 in Wuhan, China. After only a few months, the virus was hit by the global outbreak in 2020. On May 2020 The World Health Organization (WHO) announced the situation as the pandemic. The statistics by WHO on 26 August 2020 confirms 23.8 million infected people in 200 countries. The mortality rate of the infectious virus also shows a scary number of 815,000 people. With the growing trend of patients, there is still no effective cure or available treatment for the virus. While scientists, healthcare organizations, and researchers are continuously working to produce appropriate medications or vaccines for the deadly virus, no definite success has been reported at the time of this research, and there are no certain treatments or recommendation to prevent or cure this new disease. Therefore, precautions are taken by the whole world to limit the spread of infection. These harsh conditions have forced the global communities to look for alternative ways to reduce the spread of the virus. Social distancing, precaution actions to prevent the proliferation of the disease, by minimizing the proximity of human physical contacts in covered or crowded public places.

2. Related Work

In Recent paper [1] Centroid tracking algorithm is used for calculating pairwise distances between the objects. To automate the process of monitoring the social distancing it is an efficient real-time deep learning based framework.
Deep learning technique, addressed in paper [2] It is important to monitor the social distance and wear masks at public places and take actions accordingly. This tool was developed to alert humans to maintain a safe distance with each other by video feed. Deep learning gained more attention in object detection which is used for human detection purposes.

The video frame from the camera is used as input, and the open-source object detection YOLOv3 object detection is used to identify humans in video sequences.

Paper [3] proposed presented creating a system for person detection for monitoring crowd social distance. In the first part, we used deep learning approach for the object detection where we trained a model that reached good performances for the task. For the social distance and location calculation, we have developed a method based on triangulation equations to transform from the image pixel space to 3D world space and calculating the distance between each pair of persons, and then we did a proof of concept test using phone camera and measurements.

In paper [4] this tail tracking and social distance robot, we can calculate the distance between people and limit the development of viral diseases, especially where social distance is an important factor. Used in all queues of banks, government agencies, shopping malls, theatres, etc.

In paper [5]. This paper proposes a wearable social distancing detector that uses a microcontroller with an ultrasonic sensor to detect the distance between two persons and provides a warning if the person fails to obey the rule. The system could perform social distancing detection accurately and can assist in the Arduino UNO.

Paper [6] This tool could assist the efforts of the governments to regulate the virus. It are often implemented in closed areas or institutions, monitor the extent of people's commitment, and supply analysis and a faster approach to detect possibly corona suspicion cases. The results showed the success of our approach in detecting the distance with accurate measures of the important world coordinates.

Paper [7] present a queue management tool that can be used to allow people that wait for a service practice social distancing.

Paper [8] proposed a method of design optimisation for managing the rearrangement of physical spaces with social distancing constraints in the wake of the COVID-19 pandemic.

In paper [9] this paper first took advantage of interest inclusion and intersection, combined geographical information (location) and social information (interest). Then the concept of geo-social distance together with a data dissemination strategy (called GSD) have been presented to assist nodes to choose appropriate forwarders in a conference scenario.

In paper [10] propose computer vision-based social distancing monitoring by using background subtraction method. This method has potency to measure and detect persons position and measure the distance of each other. This method also offers low computational process so the need of an additional hardware such as GPU is unnecessary. In this works, we compared background several subtraction method such as Geometric Multigrid (GMG), k-Nearest Neighbor (KNN), Mixture of Gaussian (MOG), and Mixture of Gaussian 2 (MOG2).

In paper [11] The proposed framework leverages the Mask R-CNN deep neural network to detect people in a video frame. To consistently identify whether social distancing is practiced during the interaction between people, a centroid tracking algorithm is utilised to track the subjects over the course of the footage. With the aid of authentic algorithms for approximating the distance of people from the camera and between themselves. We determine whether the social distancing guidelines are being adhered to.

This paper [12] This paper proposes a different approach to monitor social distancing, using cameras, and combining different computer vision algorithms. The approach utilizes the concept of inverse perspective mapping (IPM) together with the camera's intrinsic information to produce a bird's eye view with real-world coordinates of the frame being processed from a video source. The process starts with image enhancement, foreground detection using Gaussian Mixture Model (GMM) background subtraction, tracking using Kalman filter, computing real-world distance measurements between individuals, and detecting those who have been in less than 2 meters apart as they are considered to be in contact.

In paper [13] There are many advantages for using the smart cap. It is easy to use and very cheap. It helps prevent the infection and spread of contagious diseases. It can be reused for a long period of time. Cleaning of the cap is also possible. The shield in the cap also helps replace the mask and gave clear visibility to the facial expressions and lip movement to increase the effectiveness of the conversations and understand emotions clearer.

3. Methodology

The proposed system focuses on how to identify the person on image/video stream whether the social distancing is maintained or not with the help of computer vision and deep learning algorithm by using the Open CV, Tensor flow library.

We propose and implement a AI based Social distance monitoring system to help to avoid human interaction in public places. Rising AI technology for a secure, transparent and decentralized coordination platform. To deploy pre-trained YOLOv3 for human detection and computing their bounding box centroid information. In addition, a transfer learning method is applied to enhance the performance of the model. The additional training is performed with overhead data set, and the newly trained layer is appended to the pre-trained model.

In order to track the social distance between individuals, the Euclidean distance is used to approximate the distance between each pair of the centroid of the bounding box detected. In addition, a social distance violation threshold is specified using a pixel to distance estimation. Utilizing a centroid tracking algorithm to keep track of the person who violates the social distance threshold.

A. Advantages

Having social distancing detection in the workplace is a great way of reassuring staff that the workplace has been made safe for their benefit. The solution is also safer than the thermal cameras – those without a fever can also be contagious.

With the detection software you will have the ability to see which areas gain the most traction and are the offices 'hotspots. From this data you will then be able to put the most relevant safety measures in place.

The technology isn't just for the office, for example, at a factory where employees are very close to each other, the software can be integrated into their security camera systems. Allowing them to monitor the working environment and highlight people who's distancing is below the minimum acceptable distance.

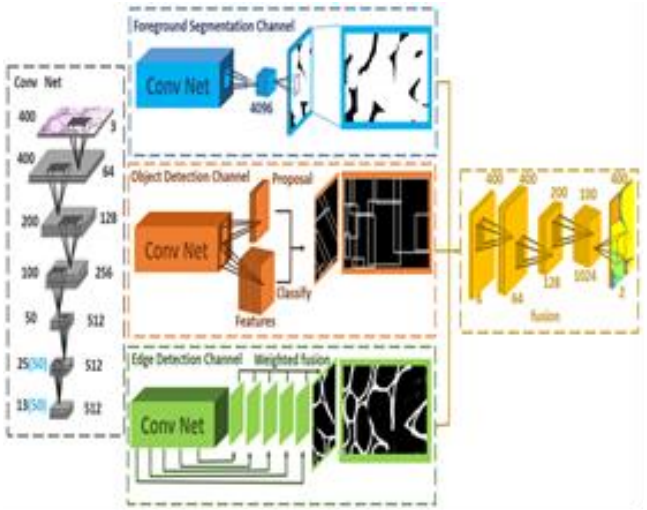

Fig 1: Architecture Diagram

4. Implementation

A. Tensorflow

Tensorflow is a symbolic math library based on dataflow and differentiable programming. It is used for both research and production at Google. Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

Tensor Flow computations are expressed as stateful dataflow graphs. The name Tensor Flow derives from the operations that such neural networks perform on multidimensional data arrays, which are referred to as tensors.

TensorFlow provides stable Python (for version 3.7 across all platforms) and C APIs; and without API backwards compatibility guarantee: C++, Go, Java, JavaScript and Swift (archived and development has ceased). Third-party packages are available for C#, Haskell, Julia, MATLAB, R, Scala, Rust, OCaml, and Crystal.

"New language support should be built on top of the C API.

B. Yolo v3

YOLO V3 is an improvement over previous YOLO detection networks. Compared to prior versions, it features multi-scale detection, stronger feature extractor network, and some changes in the lossfunction. As a result, this network can now detect many more targets from big to small. And, of course, inference possible on GPU devices. Well, as a beginner to object detection, you might not have a clear image of what do they mean here. But you will gradually understand them later in my post. For now, just remember that YOLO V3 is one of the best models in terms of real-time object detection as of 2019.

Fig 2: Network Architecture

First of all, let's talk about how this network look like at a high-level diagram (Although, the network architecture is the least time-consuming part of implementation). The whole system can be divided into two major components: Feature Extractor and Detector; both are multi-scale. When a new image comes in, it goes through the feature extractor first so that we can obtain feature embeddings at three (or more) different scales. Then, these features are feed into three (or more) branches of the detector to get bounding boxes and class information.

C. Darknet-53

The feature extractor YOLO V3 uses is called Darknet-53. You might be familiar with the previous Darknet version from YOLO V1, where there're only 19 layers. But that was like a few years ago, and the image classification network has progressed a lot from merely deep stacks of layers. ResNet brought the idea of skip connections to help the activations to propagate through deeper layers without gradient diminishing. Darknet-53 borrows this idea and successfully extends the network from 19 to 53 layers, as we can see from the following diagram.

Table 1: Darknet-53

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

D. Diagram from YOLOv3: An Incremental Improvement

This is very easy to understand. Consider layers in each rectangle as a residual block. The whole network is a chain of multiple blocks with some strides 2 Conv layers in between to reduce dimension. Inside the block, there's just a bottleneck structure (1x1 followed by 3x3) plus a skip connection. If the goal is to do multi-class classification as ImageNet does, an average pooling and a 1000 ways fully connected layers plus softmax activation will be added.

However, in the case of object detection, we won't include this classification head. Instead, we are going to append a "detection" head to this feature extractor. And since YOLO V3 is designed to be a multi-scaled detector, we also need features from multiple scales. Therefore, features from last three residual blocks are all used in the later detection. In the

diagram below, I'm assuming the input is 416x416, so three scale vectors would be 52x52, 26x26, and 13x13. Please note that if the input size is different, the output size will differ too.

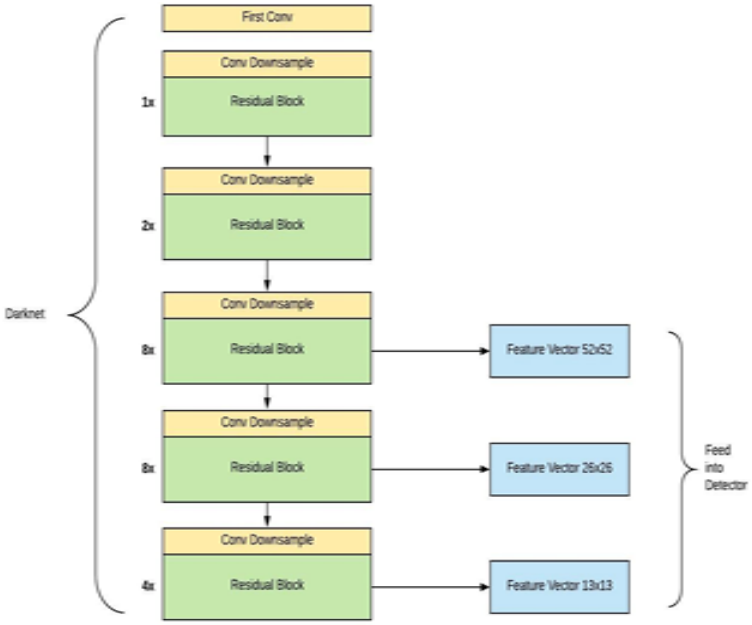

Fig 3: YOLOV3

E. Multi-scale Detector

Once we ave three features vectors, we can now feed them into the detector. But how should we structure this detector? Unfortunately, the author didn't bother to explain this part this his paper. But we could still take a look at the source code he published on Github. Through this config file, multiple 1x1 and 3x3 Conv layers are used before a final 1x1 Conv layer to form the final output. For medium and small scale, it also concatenates features from the previous scale. By doing so, small scale detection can also benefit from the result of large scale detection.

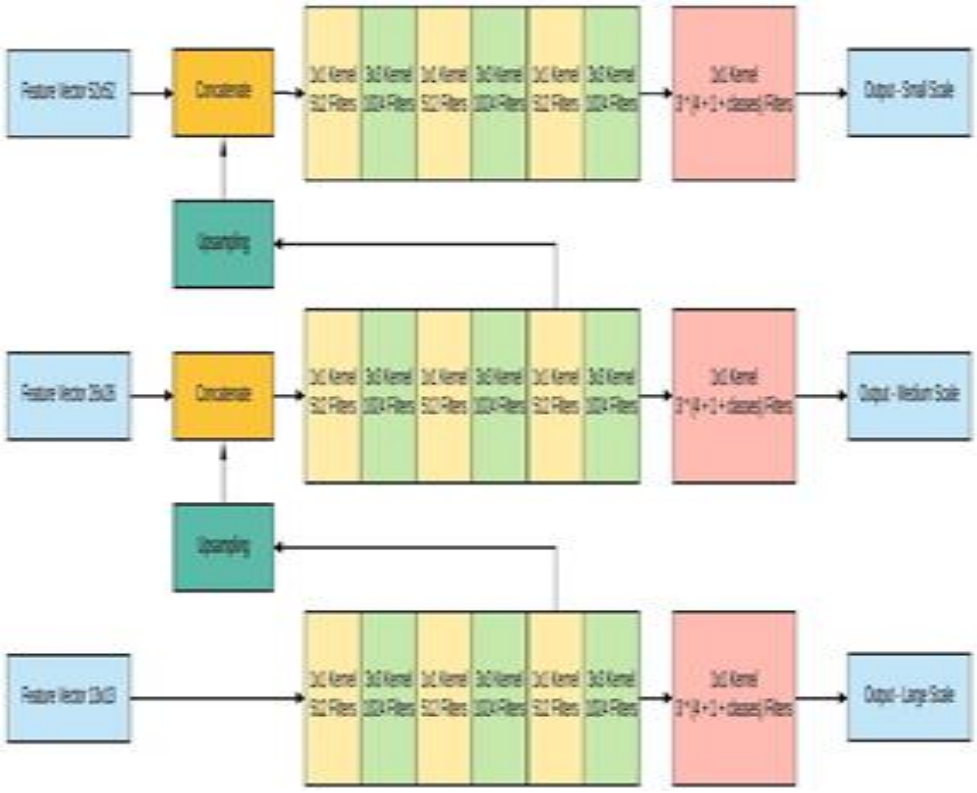

Fig 4: Multi-scale Detector

Assuming the input image is (416, 416, 3), the final output of the detectors will be in shape of [(52, 52, 3, (4 + 1 + num_classes)), (26, 26, 3, (4 + 1 + num_classes)), (13, 13, 3, (4 + 1 + num_classes))]. The three items in the list represent detections for three scales. But what do the cells in this 52x52x3x(4+1+num_classes) matrix mean? Good questions.

This brings us to the most important notion in pre-2019 object detection algorithm: anchor box (prior box).

F. Anchor Box

The goal of object detection is to get a bounding box and its class. Bounding box usually represents in a normalized xmin, ymin, xmax, ymax format. For example, 0.5 xmin and 0.5 ymin mean the top left corner of the box is in the middle of the image. Intuitively, if we want to get a numeric value like 0.5, we are facing a regression problem. We may as well just have the network predict for values and use Mean Square Error to compare with the ground truth. However, due to the large variance of scale and aspect ratio of boxes, researchers found that it's really hard for the network to converge if we just use this "brute force" way to get a bounding box. Hence, in Faster-RCNN paper, the idea of an anchor box is proposed. Anchor box is a prior box that could have different pre-defined aspect ratios. These aspect ratios are determined before training by running K-means on the entire dataset. But where does the box anchor to? We need to introduce a new notion called the grid. In the "ancient" year of 2013, algorithms detect objects by using a window to slide through the entire image and running image classification on each window. However, this is so inefficient that researchers proposed to use Conv net to calculate the whole image all in once (technically, only when your run convolution kernels in parallel.) Since the convolution outputs a square matrix of feature values (like 13x13, 26x26, and 52x52 in YOLO), we define this matrix as a "grid" and assign anchor boxes to each cell of the grid. In other words, anchor boxes anchor to the grid cells, and they share the same centroid. And once we defined those anchors, we can determine how much does the ground truth box overlap with the anchor box and pick the one with the best IOU and couple them together. I guess you can also claim that the ground truth box anchors to this anchor box. In our later training, instead of predicting coordinates from the wildwest, we can now predict offsets to these bounding boxes. This works because our ground truth box should look like the anchor box we pick, and only subtle adjustment is needed, whhich gives us a great head start in training.

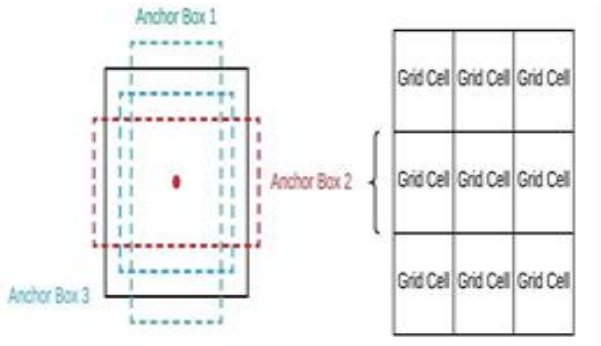

Fig 5: Anchor Box

In YOLO v3, we have three anchor boxes per grid cell. And we have three scales of grids. Therefore, we will have 52x52x3, 26x26x3 and 13x13x3 anchor boxes for each scale. For each anchor box, we need to predict 3 things:

1. The location offset against the anchor box: tx, ty, tw, th. This has 4 values
2. The objectness score to indicate if this box contains an object this has 1 value.
3. The class probabilities to tell us which class this box belongs to. This has num_classes values.

In total, we are predicting 4 + 1 + num_classes values for one anchor box, and that's why our network outputs a matrix in shape of 52x52x3x(4+1+num_classes) as I mentioned before. tx, ty, tw, th isn't the real coordinates of the bounding box. It's just the relative offsets compared with a particular anchor box. I'll explain these three predictions more in the Loss Function section after.

G. Anchor

Anchor box not only makes the detector implementation much harder and much error-prone, but also introduced an extra step before training if you want the best result. So, personally, I hate it very much and feel like this anchor box idea is more a hack than a real solution. In 2018 and 2019, researchers start to question the need for anchor box. Papers like Corner Net, Object as Points, and FCOS all discussed the possibility of training an object detector from scratch without the help of an anchor box.

A. Loss Function

With the final detection output, we can calculate the loss against the ground truth labels now. The loss function consists of four parts (or five, if you split noobj and obj): centroid (xy) loss, width and height (wh) loss, objectness (obj and noobj) loss and classification loss. When putting together, the formula is like this:

```
Loss = Lambda_Coord *
Sum(Mean_Square_Error((tx, ty), (tx', ty') * obj_mask)
+ Lambda_Coord * Sum(Mean_Square_Error((tw, th), (tw',
th') * obj_mask)
+ Sum(Binary_Cross_Entropy(obj, obj') * obj_mask) +
Lambda_Noobj * Sum(Binary_Cross_Entropy(obj, obj') * (1
-obj_mask) * ignore_mask)
+ Sum(Binary_Cross_Entropy(class, class'))
```

The first part is the loss for bounding box centroid. tx and ty is the relative centroid location from the ground truth. tx' and ty' is the centroid prediction from the detector directly. The smaller this loss is, the closer the centroids of prediction and ground truth are. Since this is a regression problem, we use mean square error here. Besides, if there's no object from the ground truth for certain cells, we don't need to include the loss of that cell into the final loss. Therefore we also multiple by obj_mask here. obj_mask is either 1 or 0, which indicates if there's an object or not. In fact, we could just use obj as obj_mask, obj is the object ness score that I will cover later. One thing to note is that we need to do some calculation on ground truth to get this tx and ty. So, let's see how to get this value first. As the author says in the paper:

```
xy_loss = Lambda_Coord *
Sum(Mean_Square_Error((tx, ty), (tx', ty')) * obj_mask)
```

Here bx and by are the absolute values that we usually use as centroid location. For example, bx = 0.5, by = 0.5 means that the centroid of this box is the center of the entire image. However, since we are going to compute centroid off the

anchor, our network is actually predicting centroid relative the top-left corner of the grid cell. Why grid cell? Because each anchor box is bounded to a grid cell, they share the same centroid. So the difference to grid cell can represent the difference to anchor box. In the formula above, sigmoid (tx) and sigmoid (ty) are the centroid location relative to the grid cell. For instance, sigmoid (tx) = 0.5 and sigmoid (ty) = 0.5 means the centroid is the center of the current grid cell (but not the entire image). Cx and Cy represents the absolute location of the top-left corner of the current grid cell. So if the grid cell is the one in the SECOND row and SECOND column of a grid 13x13, then Cx = 1 and Cy = 1. And if we add this grid cell location with relative centroid location, we will have the absolute centroid location bx = 0.5 + 1 and by = 0.5 + 1. Certainly, the author won't bother to tell you that you also need to normalize this by dividing by the grid size, so the true bx would be 1.5/13 = 0.115. Ok, now that we understand the above formula, we just need to invert it so that we can get tx from bx in order to translate our original ground truth into the target label. Lastly, Lambda_Coord is the weight that Joe introduced in YOLO v1 paper. This is to put more emphasis on localization instead of classification. The value he suggested is 5.

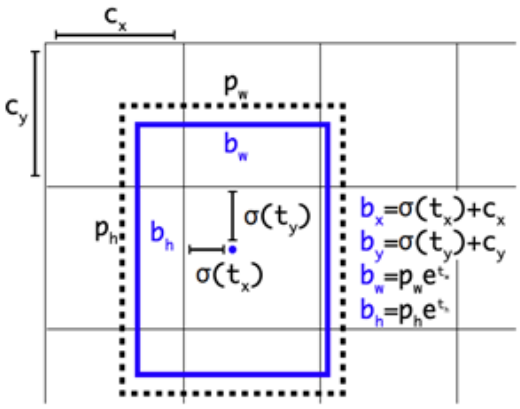

Fig 2: Boundary boxes with dimension priors and location prediction

Diagram from YOLOv3: An Incremental Improvement

```
wh_loss = Lambda_Coord * Sum(Mean_Square_Error((tw, th), (tw', th')) * obj_mask)
```

The next one is the width and height loss. Again, the author says:

```
bw = exp(tw) * pw
bh = exp(th) * ph
```

Here bw and bh are still the absolute width and height to the whole image. pw and ph are the width and height of the prior box (aka. anchor box, why there're so many names). We take e^(tw) here because tw could be a negative number, but width won't be negative in real world. So this exp() will make it positive. And we multiply by prior box width pw and ph because the prediction exp(tw) is based off the anchor box. So this multiplication gives us real width. Same thing for height. Similarly, we can inverse the formula above to translate bw and bh to tx and th when we calculate the loss.

```
obj_loss = Sum(Binary_Cross_Entropy(obj, obj') * obj_mask)noobj_loss = Lambda_Noobj * Sum(Binary_Cross_Entropy(obj, obj') * (1 — obj_mask) * ignore_mask)
```

The third and fourth items are object ness and non-object ness score loss. Object ness indicates how likely is there an object in the current cell. Unlike YOLO v2, we will use binary cross-entropy instead of mean square error here. In the ground truth, object ness is always 1 for the cell that contains an object, and 0 for the cell that doesn't contain any object. By measuring this obj_loss, we can gradually teach the network to detect a region of interest. In the meantime, we don't want the network to cheat by proposing objects everywhere. Hence, we need noobj_loss to penalize those false positive proposals. We get false positives by masking prediction with 1-obj_mask. The `ignore_mask` is used to make sure we only penalize when the current box doesn't have much overlap with the ground truth box. If there is, we tend to be softer because it's actually quite close to the answer. As we can see from the paper, "If the bounding box prior is not the best but does overlap a ground truth object by more than some threshold we ignore the prediction." Since there are way too many noobj than obj in our ground truth, we also need this Lambda_Noobj = 0.5 to make sure the network won't be dominated by cells that don't have objects.
class_loss = Sum (Binary_Cross_Entropy (class, class') * obj_mask)
The last loss is classification loss. If there're 80 classes in total, the class and class' will be the one-hot encoding vector that has 80 values. In YOLO v3, it's changed to do multi-label classification instead of multi-class classification. Why? Because some dataset may contains labels that are hierarchical or related, eg, woman and person. so each output cell could have more than 1 class to be true. Correspondingly, we also apply binary cross-entropy for each class one by one and sum them up because they are not mutually exclusive. And like we did to other losses, we also multiply by this obj_mask so that we only count those cells that have a ground truth object.
To fully understand how this loss works, I suggest you manually walk through them with a real network prediction and ground truth. Calculating the loss by your calculator (or tf math) can really help you to catch all the nitty-gritty details. And I did that by myself, which helped me find lots of bugs.

5. Work Done

A. Approach

1. Detect humans in the frame with yolov3.
2. Calculates the distance between every human who is detected in the frame.
3. Shows how many people are at High, Low and Not at risk.

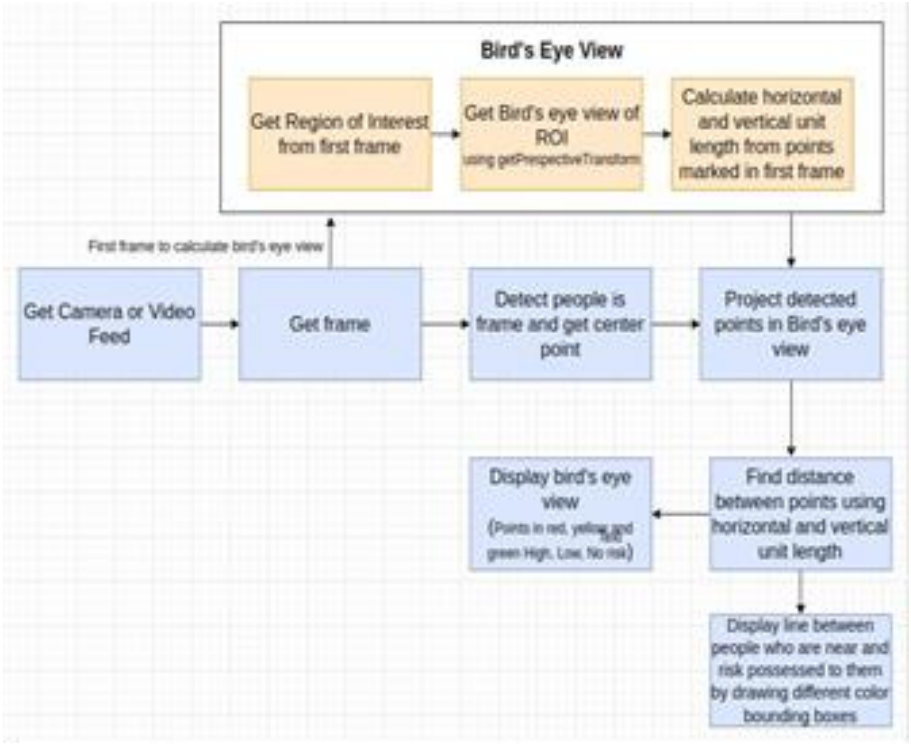

Fig 6: Flow diagram of social distancing detector model

B. Camera Perspective Transformation or Camera Calibration

As the input video may be taken from an arbitrary perspective view, the first step is to transform perspective of view to a bird's-eye (top-down) view. As the input frames are monocular (taken from a single camera), the simplest transformation method involves selecting four points in the perspective view which define ROI where we want to monitor social distancing and mapping them to the corners of a rectangle in the bird's-eye view also these points should form parallel lines in real world if seen from above (bird's eye view). This assumes that every person is standing on the same flat ground plane. This top view or bird eye view has the property that points are distributed uniformly horizontally and vertically (scale for horizontal and vertical direction will be different). From this mapping, we can derive a transformation that can be applied to the entire perspective image.

Fig 7: Bird eye view has the property that points are distributed uniformly horizontally and vertically

Above image shows how we can select Region of Interest (ROI) and this is one time step. We draw 8 points on first frame using mouse click event. First four points will define ROI where we want to monitor social distancing. Next 3 points will define 180 cm (unit length) distance in horizontal and vertical direction and those should form parallel lines with ROI. In above image we can se point 5 and point 6 defines 180 cm in real life in horizontal direction and point 5 and point 7 defines 180 cm in real life in vertical direction. As we can see ROI formed by first 4 points has different length in horizontal and vertical direction, so number of pixels in 180 cm for horizontal and vertical direction will be different in rectangle (bird's eye view) formed after transformation.

So from point 5, 6, 7 we are calculating scale factor in horizontal and vertical direction of the bird's eye view, e.g. how many pixels correspond to 180 cm in real life.

C. Detection

The second step to detect pedestrians and draw a bounding box around each pedestrian. To clean up the output bounding boxes, we apply minimal post-processing such as non-max suppression (NMS) and various rule-based heuristics, so as to minimize the risk of over fitting.

D. Distance Calculation

Now we have bounding box for each person in the frame. We need to estimate person location in frame. i.e we can take bottom center point of bounding box as person location in frame. Then we estimate (x,y) location in bird's eye view by applying transformation to the bottom center point of each person's bounding box, resulting in their position in the bird's eye view. Last step is to compute the bird's eye view distance between every pair of people and scale the distances by the scaling factor in horizontal and vertical direction estimated from calibration.

E. Working

Running the program will give you frame (first frame) where you need to draw ROI and distance scale. To get ROI and distance scale points from first frame Code to transform perspective to Bird's eye view (Top view) and to calculate horizontal and vertical 180 cm distance in Bird's eye view

ROI and Scale points' selection for first frame. The second step to detect pedestrians and draw a bounding box around each pedestrian. To detect humans in video and get bounding box details.

Now we have bounding box for each person in the frame. We need to estimate person location in frame. i.e we can take bottom center point of bounding box as person location in frame. Then we estimate (x,y) location in bird's eye view by applying transformation to the bottom center point of each person's bounding box, resulting in their position in the bird's eye view. To calculate bottom center point for all bounding boxes and projecting those points in Bird's eye view. Last step is to compute the bird's eye view distance between every pair of people (Point) and scale the distances by the scaling factor in horizontal and vertical direction estimated from calibration.

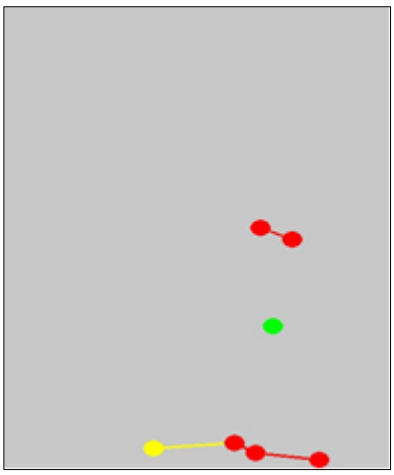

Fig 8: Bird Eye view

F: Bird Eye view

Lastly we can draw Bird's Eye View for region of interest (ROI) and draw bounding boxes according to risk factor for humans in a frame and draw lines between boxes according to risk factor between two humans. Red, Yellow, Green points represents risk to human in Bird's eye view. Red: High Risk, Yellow: Low Risk and Green: No Risk. Red, Yellow lines between two humans in output tell they are violating social distancing rules.

6. Conclusion

The emerging trends and the availability of intelligent technologies make us to develop new models that help to satisfy the needs of emerging world so we have developed a novel social distancing detector which can possibly contribute to public healthcare. The model proposes an efficient real-time deep learning based framework to automate the process of monitoring the social distancing via object detection and tracking approaches, where each individual is identified in the real-time with the help of bounding boxes. Identifying the clusters or groups of people satisfying the closeness property computed with the help of Bird's eye view approach. The number of violations is confirmed by computing the number of groups formed and violation index term computed as the ratio of the number of people to the number of groups. The extensive trials were conducted with popular state-of-the-art object detection models Faster RCNN, SSD, and YOLO v3, since this approach is highly sensitive to the spatial location of the camera, the same approach can be fine-tuned to better adjust with the corresponding field of view. This system works very effectively and efficiently in identifying the social distancing between the people and generating the alert that can be handled and monitored.

7. Future Enhancement

The Present work is fully focused on the scheduling based process. We have a proposed AI based model and presented a Map Reduce to solve the social distance monitoring process. Simulations have shown that our Map Reduce process can significantly reduce the time of job execution and resource allocation.

7. References

1. Manthri S, Allani LS, Gandla A, Jindam S. Social distancing detector using deep learning. International Journal of Recent Technology and Engineering; 2022:10(5):146-9. Available from: https://www.ijrte.org/wp-content/uploads/papers/v10i5/E55600110522.pdf
2. Kokila B, Muthukumar N, Mounikha R. Social distancing detection with deep learning model. International Journal of Advanced Research and Innovative Ideas in Education; 2022:8(3):334-338.
3. El Habchi A, Baibai K, Moumen Y, Zerouk I, Khiati W, Rahmoune N, *et al*. Social distance monitoring using YOLOv4 on aerial drone queue. E3S Web of Conferences. 2022;351:01035. Available from: https://doi.org/10.1051/e3sconf/202235101035
4. Vijay A, Gupta A, Pal A, Sriswathi B, Mathur G, Alaria SK. IoT social distancing & monitoring robot for queue. International Journal of Engineering Trends and Applications; 2021:8(4):20-25.
5. Rinisha CP, Aruna B. Wearable social distancing detection system. International Journal of Creative Research Thoughts. 2021:9(10):901-906. Available from: https://www.ijcrt.org/papers/IJCRT2105522.pdf
6. Jayanthi G, Gayathri R, Vishalakshi M. Social distancing detector and indicator using Arduino. Ilkogretim Online - Elementary Education Online. 2020;19(3):4722-4732. doi: 10.17051/ilkogretim.2020.03.735626
7. Kyritsis, Deriaz M. A queue management approach for social distancing and contact tracing. In: Proceedings of the Third International Conference on Artificial Intelligence for Industries (AI4I); c2020.
8. Ugail H, Aggarwal R, Iglesias A, Suarez P. An optimization model for designing social distancing enhanced physical spaces. In: Proceedings of the International Conference on Internet of Things and Intelligent Applications (ITIA); c2020. doi: 10.1109/ITIA50152.2020.9312372
9. Li J, Ning Z, Jedari B, Xia F, Lee I, Tolba A. Geo-social distance-based data dissemination for socially aware networking. IEEE Access. 2016;4:4771-4781. doi: 10.1109/ACCESS.2016.2553698
10. Adinanta H, Suryadi, Prakosa JA. Physical distancing

monitoring with background subtraction methods. In: Proceedings of the International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications; c2020.
11. Gupta S, Kapil R, Kanahasabai G, Joshi SS, Joshi AS. SD-measure: A social distancing detector. In: Proceedings of the 12th International Conference on Computational Intelligence and Communication Networks; c2020.
12. Gad A, ElBary G, Alkhedher M, Ghazal M. Vision-based approach for automated social distance violators detection. In: Proceedings of the Department of Electrical and Computer Engineering, Abu Dhabi University; c2020.
13. Ragha S, Vijay G, Harika PS, Rao AV, Gopinath A, Shibu SNB, *et al*. Low cost device to maintain social distancing during COVID-19. Amrita Vishwa Vidyapeetham, Amritapuri; c2020.
14. Rusli ME, Ali M, Yussof S. A smart social distancing monitoring system. Universiti Tenaga Nasional; c2020. Available from: https://www.researchgate.net/publication/348122877_A_Smart_Social_Distancing_Monitoring_System
15. Degadwala S, Vyas D, Dave H. Visual social distance alert system using computer vision & deep learning. Proceedings of the Conference on Computer Vision and Pattern Recognition; c2020.