2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE) 978-1-7281-4142-8/20/$31.00 ©2020 IEEE 10.1109/ic-ETITE47903.2020.163

Real Time Object Detection in Surveillance Cameras with Distance Estimation using Parallel Implementation

Meherdeep Thakur
Vellore Institute of Technology
Vellore, India
meherdeept@gmail.com

Saira Banu J
Vellore Institute of Technology
Vellore, India
jsairabanu@vit.ac.in

Abstract—**Object detection is not only shaping how Computers see and analyze things but it is also helping in the behavior of how an object reacts to the change in its environment. The main application of these object detection sensors or software is to find the location of an object in space or to track its movement. Object detection has infinitely many use cases and in this paper, we are introducing an application that will allow safety of users struck in a disaster and who need to be evacuated. In such cases the main thing to focus and to eradicate is camera noise, saturation and image compression. Our solution is to establish a connection between the person struck in a disaster with fire safety people. This works over a convolutional network that allows us to detect vulnerable things present inside a room that needs to be rescued and can also give an insight of any explosive inside the room. Our model uses Faster-RCNN and COCO which is a pretrained dataset. This allows real time object detection and classification on our network. Using this we were able to detect an object or a person and get him to rescue by providing them a shortest way out of that place. With this we were able to get an accuracy of more than 75% in our object detection model.**

Keywords—Object Detection, RCNN, Convolutional Neural Networks, Distributed Computing, Parallel Processing

I. INTRODUCTION

Object detection is a very powerful tool and it can be sometimes hard to implement and get best results out of it.Object detection is done using two kinds of networks namely Neural Networks and Convolutional Neural Networks (CNN). The main difference between the two type of networks is the way they take the input for an image.Traditionally Neural Networks are built and trained over vectors that is they take a 2-dimensional view of an image whereas in CNN it takes a tensor as an input where it has height, depth and width as its parameter. Moreover, we saw that in traditional methods an image used to be converted to gray scale and then used for computation. But when dealing with real time detection on a video, color acts as a major classifying element. Hence CNN takes color as one of its parameters. The tensor is taken as input for CNN, the 3 parameters for this tensor are: a) image with H rows, b) W columns, c) and 3 channels (R, G, B channels). The input goes through series of sequential steps or layers. There are many types of layers involved which processes the images and find details from an image, few of the common layers are: convolution layer, pooling layer, ReLU layer and loss layer. There has been many advancements in object detection mainly which is driven by success of region proposal method. In this model we have used Faster CNN which attains real time rates using deep neural networks. It is also possible that in an image that there can be multiple objects of interest, Faster CNN allows real-time multiple-object detection. This means that there can be multiple people, animals, birds and other objects of interest that can be detected at once. In object detection, all these objects are bounded by these rectangular frames which define the location and the object. The area of convergence of the object detection bounding to the real bounding is commonly known as Intersection over Union (IoU). Depending upon various parameters like feature extraction method, sliding window size, video quality, etc. Due to this, an IoU value over 0.5 is considered as a good detection. This value changes depending upon the severity of the situation.This CNN take advantage of GPU in computing these images. GPU have thousands of core which work in parallel as compared to a CPU which has a very limited number of cores. This increase in the number of cores increases the distributed computing and hence computation speed. The computation speed is so advanced that it takes 0.2s per image or even less.

II. RELATED WORK

[1] The paper proposes a framework combined of advantages of image embedding and label models by using CNN and RNN.
[2] The proposed model can focus on different image regions when predicting labels but small objects still poses a challenge. In future the work can be extended to not only predict labels but also predict segmentation.
[3] The paper proposes a framework combining the advantages of image embedding and label models by using CNN and RNN.

[4] Color information is the most significant visual feature in agriculture products and it has good performance on invariance of size and view point change.
[5] Its advantage is that it is tolerant to fault and is powerful to classify. Through training the neural networks with specific color, shape pictogram, features of interest region can be recognized.
[6] The work proposes how computer vision can be used to classify fruits based on their live time feed. The work proposes an RSS model for HSIs classification.
[7]Mask-RCNN, relies on region proposals which are generated via a region proposal network. Mask-RCNN follows the Faster-RCNN model by having a feature extractor followed by this region proposal network. Composed of 5 main parts: a deep fully convolutional network, region proposal network, ROI pooling and fully connected networks, bounding box regressor, and classifier

III. PROCEDURE

A) YOLO - You Only Look Once
YOLO is an extremely fast real-time multi-object mapping algorithm used for object detection. YOLO algorithm divides a frame into grids which when processed gives a final output with bounding boxes over the object along with their probabilities. These probabilities are nothing but Intersection over Union (IoU) which signifies the bounding area convergence. On a live feed or a surveillance field take in all the frames as input and processes each frame for objects present inside it. On a system with CPU the time to process a single frame is less than 0.3 seconds and this time can be even reduced by the use of GPU.

A) COCO Dataset
COCO dataset is a JSON file that has the details of all the objects that we want to detect in our model. This JSON file takes a lot of details while registering an image, hence giving much accurate detections. Such parameters include-
height, width, id, box coordinates etc. Moreover, COCO also allows us to add more and more items to the dataset by training the data locally on their system. We have here trained data to find vulnerable items inside a room which can be vulnerable or can lead to some serious damage during a case of emergency. Example Gas Cylinder, Grenades any kind of weaponry.

B) Setting up a remote camera or a drone
Cameras on the drones can be connected to using an IP address that is assigned to the drone. This way whenever the script is run on the system it connects to the drone camera and all the frames from the drone are then processed locally
on our system.

C) Code
Object Detection Over here we have used OpenCV as a tool to make our object detection model. OpenCV is an open source Computer Vision library available for python language. The library has more than 2500 optimized algorithms.

```
while True:

   frame = vs.read()
   frame = imutils.resize(frame, width=800)

   (h, w) = frame.shape[:2]
   blob = cv2.dnn.blobFromImage(cv2.resize(frame,
(300, 300)),0.007843, (300, 300), 127.5)

   net.setInput(blob)
   detections = net.forward()

   for i in np.arange(0, detections.shape[2]):

      confidence = detections[0, 0, i, 2]
      if confidence > args["confidence"]:
         idx = int(detections[0, 0, i, 1])
         box = detections[0, 0, i, 3:7] *
np.array([w, h, w, h])
         (startX, startY, endX, endY) =
box.astype("int")
{:.2f}".format(CLASSES[idx],confidence * 100)
         cv2.rectangle(frame, (startX, startY),
(endX, endY),COLORS[idx], 2)
         y = startY - 15 if startY - 15 > 15 else
startY + 15
         cv2.putText(frame, label, (startX,
y),cv2.FONT_HERSHEY_SIMPLEX, 0.5, COLORS[idx], 2)

   cv2.imshow("Frame", frame)
   if cv2.waitKey(1) & 0xFF == ord('q'):
      break
   fps.update()
```

Moreover, when a frame is captured and passed through this code. The following steps are done. For example we take the below image as a frame from our camera.

Fig1. Example of a object detection.

Firstly the frame is passed through a foreground detector. In this the algorithm segments the objects from its background which is not needed while classifying these objects. Hence this algorithm help us provide the pixels that we are interested in. The resulting image is used as input for an object detector where bounding boxes are added around these objects and then these objects are classified according to the dataset on which the images are trained.

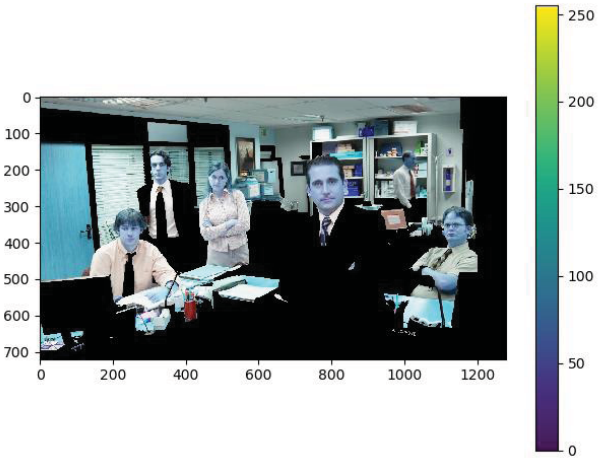

Fig2. Example of foreground detection.

Fig3.. Example of bounding boxes in object detection.

```
import cv2

cap = cv2.VideoCapture('http://<IP
Address>:<Port>')

while(True):
    ret, frame = cap.read()

    cv2.imshow('frame',frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Fig4.. Code to connect to remote feed.

.

All the remote surveillance cameras and drones have an IP address assigned to it which it uses for broadcasting. Most of these broadcasting happens over a http or rstp channel which allows easier and faster broadcasting of the video from these cameras. The above code refers to the above address to play the footage of the remote camera.

Once we have the real time video from the surveillance cameras we need to have a dataset of items which we think are vulnerable and can be a possible threat in case of disaster or in general for safety of the workplace or household. Here we will train a dataset of fire extinguishers so that in case of a disaster or rescue a fire extinguisher can be detected easily.

IV. OBJECT DETECTION ON MULTIPLE CORES

Real time object detection can be a very intensive task that takes up a lot of computation time and might need a good CPU along with a GPU to process the data given to it. The best solution to get things working with limited CPU and GPU usage is to use multi-core processes in the computation. In today's world we see very high computing chips placed in our cell phones, laptops, computers and tables which go from 2 to 8 cores. These cores are nothing but multiple CPU running on the same chip. Multicore processors help divide the work to the multiple CPUs embedded on the chip. These CPUs work in parallel to do the same task or a different task. Since all these tasks are being computed parallelly they decrease the complexity and the computation time. This gives us the ability to do a real time scan of the surveillance cameras and do multiple object detection and that too real time. Object detection has various tasks involved which leads to the real time object detection. In parallel processing we reorder and split these tasks in an effective manner such that there is no dependency or there is no redundant entry.

A. Task splitting

Splitting or dividing the task to multiple cores present in the system is known as task splitting. Task splitting is one the major steps involved in parallel computing as it decreases the workload from a single core and distributes the work to all the cores available or assigned to the system. While splitting all these tasks we need to make sure that there are no dependencies which might be required to compute the second task. One simple example of this can be doing a group project where everyone is assigned a certain task so as to achieve one common goal in the end.

B. Task reordering

Ordering and then reordering the flow of tasks so that there are no dependencies while executing a program. This can be a case when input of task 2 is the output of task 1. One common example of this can be cooking. Because while cooking a dish we need to make sure that all the steps are followed in the same order, otherwise you might not get the dish you were looking for or not get the desired output.

Task Splitting and reordering provides an effective and simple method to achieve maximum work from our processors. Although while doing this we need to make sure that only the desired amount of processors are assigned so that the system doesn't get overheated and doesn't cause any physical damage to it. To calculate the shortest way out we have used Dijkstra's algorithm which takes in all the nodes from point A to point B.

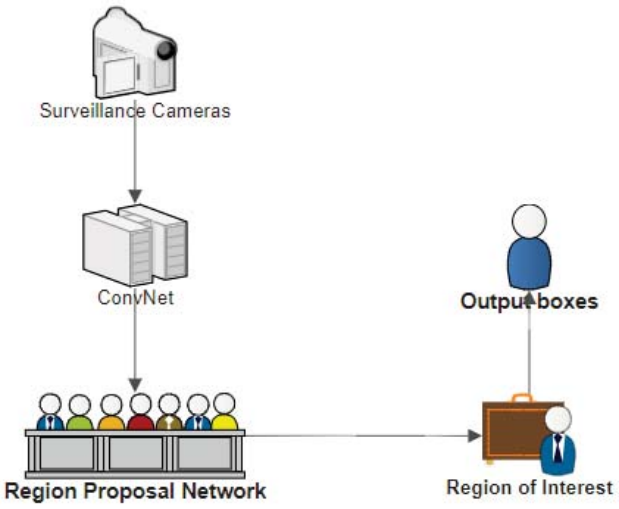

Fig5.Program Flow.

C. Architecture

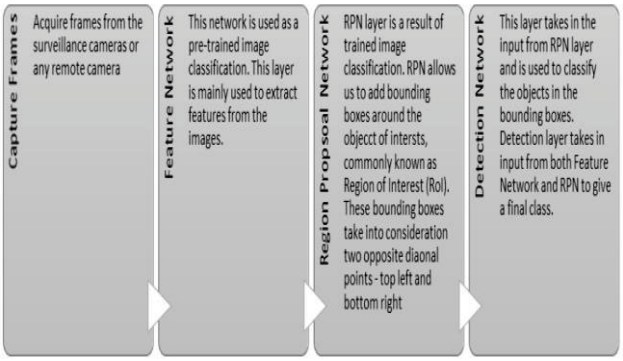

Fig6.Architecture

D. Distance Calculation

Since all these surveillance cameras are placed outside a door or a safe exit. We assume the camera to be a home for an exit door. So, the second safety model that we propose is calculating the distance and marking it with a safest and shortest way out.

Distance cameras come in handy when a person who is in a need of emergency and has no vision for an exit door. These cameras can find an exit way and can reflect the same through a mobile app on the person's mobile.
All the maps that are available currently works perfectly when asked to calculate distance from point A to point B which is connected by road and has some significant distance between it. But these maps fail to work in case of buildings as they are not aware of any obstacles in the path. Hence these surveillance cameras can be best fit to find a map or a route for a safe exit. As these maps can detect an obstacle and then can be used to mark the shortest way. Here point A is the location of the person and point B is an exit door and in between there can be several other nodes that the person has to skip or these points can simply be an obstacle like desk or chair or any other thing that you should avoid while crossing.

Fig7. .Sample Route map of an office..

Fig8. .Safest route out

Over here following steps are done using Dijkstra's algorithm:

1. A Shortest Path Tree Set is created that accounts all the available or all the objects detected in a frame.
2. Using a camera, we calculate distance between these nodes and assign these distances to the 2 adjoining nodes.
3. Assign start point and end point, over here the start point is the location of the person and end point is an exit door.
4. From the start point start comparing the distances to the adjoining nodes.
5. Store the node with the shortest distance and move forward.
6. Repeat this process until we reach the end point i.e. the exit door.

The computation time for a sequential Dijkstra's algorithm is O (V^2). Where V is set of all the vertices or nodes in a tree.

This Dijkstra's algorithm can be applied in parallel too so as to reduce the computation and to achieve maximum real time simulation possible. This is done by going through the following steps:

1. Each node searches for the closest node to the source node.
2. Perform a parallel prefix to select globally closest node.
3. Broadcast the result to all the nodes.
4. Each node updates the distance in a common list.

Using this parallel Dijkstra's algorithm, we can achieve a substantially quicker way to calculate the shortest way with computation time as:

$$O(V^2/P)+O(V*\log(P)) \quad (1)$$

Here V is the number of nodes in a tree, and P is the number of cores or processor it is running on.

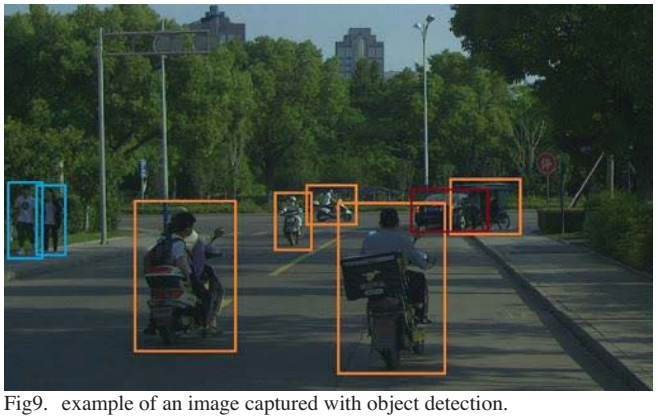

Fig9. example of an image captured with object detection.

Once we have the shortest route, the surveillance camera clicks a picture and locates those points on that image and then plots it, giving a safest route or an escape route for that person.

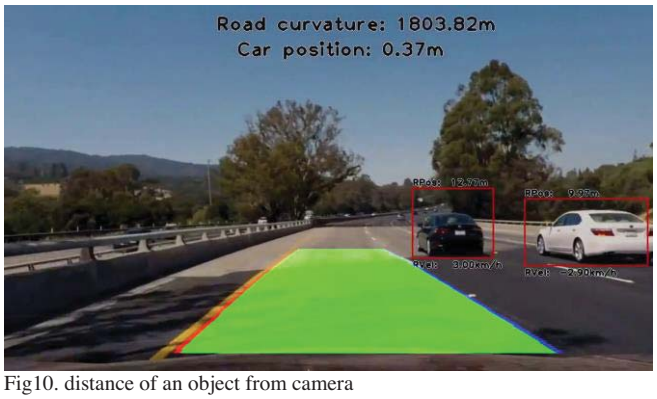

Fig10. distance of an object from camera

V. CONCLUSION

Every life matters, and in case of disasters and attacks it becomes extremely crucial and a high priority to get the civilians into rescue. So, to do so they need a mode of communication, now since these civilians are trapped inside a building or a structure and in most of these cases the telecommunication fails as well. So one of the way we can implement this communication is through sign languages which can be read and interpreted by our security cameras and tell the respective authority about the steps that need to be taken.

They say safety doesn't happen by accident. We need to have certain steps built up in order to take the right step at the right time. To ensure this safety and security our system comes in handy and can be plugged in with the existing CCTV camera or the surveillance cameras installed in the building. By the means of computer vision we can create a safer world for all of us. This system can not only be used in case of disasters but also in case of any attack on a building or any kind of intrusion.

It provides complete safety and ensures that the person gets to the rescue as soon as possible. So even if the security police is delayed they already have a rescue plan that they can use and hence get to the safety.

It also provides detection of any kind of object that doesn't suit the living environment. For example, the system can be trained to detect any kind of explosive.

With detection of accuracy of more than 75% we can ensure that a person is always safe and sound and has life in his control. Even in case of disaster our system provides the things which a naked eye could miss and hence can cause a sense of panic in them.

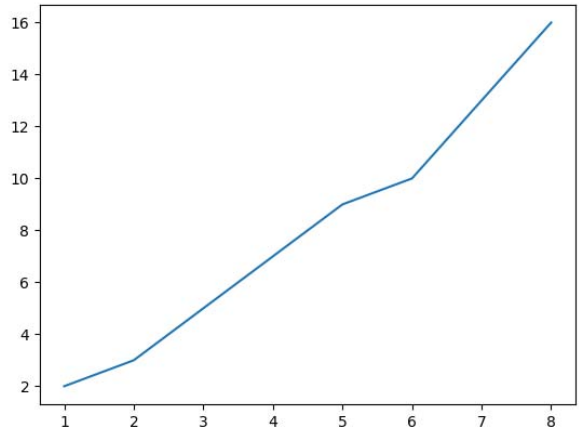

Fig11. Number of processors vs speedup

Also by using parallel computation of distance in Dijkstra's algorithm we ensure that computation is real time and has no lag. This distance once calculated can be used to direct the person to safety.

All these systems ensure that safety comes first and can prevent civilians from painful hospital trips.

References

[1] CNN-RNN: A Unified Framework for Multi-label Image Classification by Jiang Wang Yi Yang Junhua Mao Zhiheng Huang Chang Huang

Wei Xu Baidu Research University of California at Los Angles Facebook Speech Horizon Robotics

[2] Research on Computer-Vision based Object detection and Classification by Juan WuBo PengZhenxiang HuangJietao Xie

[3] Study Of Object Detection Based On Faster R-CNN by BIN LIU, Wencang ZHAO and Qiaoqiao SUN

[4] A Comparative evaluation of the GPU vs. the CPU for Parallelization of Evaluation Algorithms through multiple independent runs by Anna Syberfeldt and Tom Ekblom.

[5] Deep convolutional neural networks for brain image analysis on magnetic resonance imaging: a review by Jose Bernal∗ , Kaisar Kushibar, Daniel S. Asfaw, Sergi Valverde, Arnau Oliver, Robert Mart´ı, Xavier Llad

[6] Ray: A Distributed Framework For Emerging AI Applications by Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I. Jordan, Ion Stoica

[7] Refining Faster-RCNN for Accurate Object Detection" by Myung-Cheol Roh ; Ju-young Lee