

AI Powered Social Distancing surveillance system in Office and Public Places (COVID01)

An Industrial Internship Report Submitted in Partial Fulfilment of the
Requirements for Award
of
the Degree of Bachelor of Technology in Information and Communication
Technology

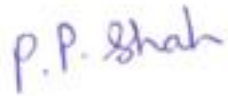
Submitted by
Parth Shah
Roll No. 17BIT051

Hackathon Team ID
T013
AI Powered Social Distancing Surveillance System in Office and Public places

Submitted to
Department of Information and Communication Technology
School of Technology
Pandit Deendayal Petroleum University (PDPU)
Gandhinagar, INDIA, 382007

Declaration

I hereby declare that the project work entitled “**Project Title: AI powered social distancing surveillance system in office and public places**” is an authentic record of my own work carried out as a part of Hackathon COVID19 activity and requirement of B. Tech dissertation for the award of **Bachelor of Technology in Information and Communication Technology**. I have duly acknowledged all the sources from which the ideas and extracts have been taken. The project is free from any plagiarism and has not been submitted elsewhere for any degree, diploma and certificate.



Signature:

Parth Shah
(17BIT051)

Abstract

Social distancing is the key tool and an effective measure to mitigate the spread of the virus amidst the ongoing pandemic. Hence, we bring you a system which can precisely detect whether people are following the distancing norms or not. We have used state of the art Mask RCNN pre-trained COCO model for detecting objects in frames of the video. Mask Region-based Convolutional Neural Network is based on ResNet 101 Architecture. The COCO model is pre-trained to detect Certain Objects in an image. We have used Frame by frame processing for detecting objects and then extracted persons RoI (Region of Interest) and then used our Algorithm to test for social distancing. Later, we visualize the frame by highlighting the person, based on certain parameters.

Contents

Abstract	ii
Contents	iii
1 Introduction	1
1.1 Problem statement	1
1.2 Motivation	1
1.3 Literature review	1
1.4 Plan of execution	2
2 Experimental Methods and Results	3
2.1 Methods	3
2.2 Algorithm	3
2.3 Results	4
3 Discussion and conclusion	5
Appendix	6
Bibliography	9

1. Introduction

1.1 Problem Statement

Topic: AI powered social distancing surveillance system in Office and Public places

As we get acclimated to being homebound to combat a viral pandemic, we have begun to internalize that this may be one of several changes we have to adapt to in the coming weeks. With lockdowns being uplifted in many places across the globe, people have been keenly following the news cycle to understand the measures being explored to ease restrictions.

Topmost on everyone's minds has been the imminent transition back to the workplace.

Social distancing is the key tool and an effective measure as part of the ongoing effort to mitigate the spread of Covid-19.

But maintaining social distance in public or workplaces isn't as easy as it sounds. Monitoring everyone at the same time is something beyond our reach, and hence the need of a technology which can monitor and alert people in such condition is the need of the hour.

1.2 Motivation

With the ongoing global pandemic, we all have been homebound for months and thus are gradually adapting ourselves with the "new normal". Now, with the lockdowns being uplifted, and the country following the new "unlock norms" the topmost thing in everyone's minds has been the imminent transition back to the workplace. And thus, various safety measures are to be considered along with resuming our routines. Apart from wearing masks, and sanitizing ourselves, social distancing is the key tool and effective measure in the ongoing effort to mitigate the spread of COVID-19 but, in a country like ours, social distancing isn't as easy as it sounds. Even with the cops moving around and keeping a watch on the people in public places, manually monitoring everyone at the same time is something which is beyond our reach and thus, an automated system which can monitor and alert the people in such condition is the need of the hour.

1.3 Literature Review

It is not easy for a person to always maintain social distancing at all times while moving or standing in a queue or at work place. Also, no one can monitor others to check if social distancing is being maintained at all times. Therefore, we need an

automatic system to check if social distancing is being maintained. Therefore, with the help of Mask-RCNN we solve the problem and check if the social distancing is being maintained or not.

The main issue here was to identify the person continuously for that the pre trained model of mask RCNN which detects objects in an image or video seemed to be the ideal choice as it is a very accurate model. Further challenge was to check the distance between persons as the distance from the camera and many other factors come into play. So, we simply solved it by taking centroids of the person detected and measuring the distance in pixels rather than meters with respect to all the persons in the frame.

1.4 Plan of Execution

First for the object detection on the frames we use Mask RCNN. Mask RCNN is a deep neural network aimed to solve instance segmentation problem in machine learning or computer vision. In other words, it can separate different objects in an image or a video. You give it an image, it gives you the object bounding boxes, classes and masks.

Further, we have to detect only humans in the frame. So, we use the labels assigned by Mask RCNN to sort humans aside from other objects detected by the Mask RCNN.

Then we plan to use the Euclidean distance in pixels to measure the distance between the people and compare it with a pre-defined threshold. We show the violation by changing the colour of the boxes red from blue surrounding the persons who are violating. Further at the bottom of the screen we show total number of violations in a particular frame at that point of time.

2. Experimental Methods and Results

2.1 Methods

Mask RCNN (Mask Regional Convolutional Neural Network)

Mask RCNN is a deep neural network aimed to solve instance segmentation problem in machine learning or computer vision. In other words, it can separate different objects in an image or a video. You give it an image, it gives you the object bounding boxes, classes and masks. The model.detect detects the objects present in the frame and returns the results obtained. Also, the objects detected are bound in rectangles relative to the area they cover. The Mask RCNN method returns the following:

1. rois - Region of Interests
2. masks - Refer to the object detected and selects it.
3. class_ids - Holds the value of class id it thinks the object is.
4. scores - returns the probability of the object detected

We further extract only humans from the frame and ignore the other classes predicted by the model.

The cdist function from the scipy.spatial.distance is used to calculate the Euclidean distance between each pair of the two centroids of the rectangle. Therefore, it gives us the distance (in pixels) between 2 persons which is further compared with the user entered threshold (minimum distance to be maintained) to calculate and show the violations.

2.2 Algorithm

Libraries used: *TensorFlow*, *Mask R-CNN*, *SciPy*, *OpenCV*

1. Object detection using state of the art Mask R-CNN from the given video
2. Filter human from the detected objects
3. Bound the detected humans into rectangular boxes
4. Calculate the centre of the rectangle with respect to the image
5. Calculate Euclidean distance of each centre with all other centres (in pixels)
6. If the distance is lesser than the threshold (set by user), consider it as a violation of social distancing norms (the respective box immediately turns red and line is drawn between violating centres)
7. The current number of violations occurred is shown at the bottom of the screen

2.3 Results

In the video the blue box around the person shows he is maintaining social distancing; the red box shows that the person is violating social distance a line will be drawn in between them to show which two are violating social distancing and below the video a counter is also shown.



3. Discussions and conclusion

Issues:

- Unable to distinguish between person nearby and faraway - this causes actual threshold distance being more for persons away from the camera and less for the persons nearby
- In market places, a major issue may arise wherein two (or more) people are walking together, might be considered as a violation to the social distancing norms, which is undesirable. (example: mother carrying her child in stroller)
- In offices, the area given being small, partitions are set amongst each cubicle for maintaining distance, however the designed software might not consider this and show a violation as the set distance is not maintained.

Future aspects:

- Develop 'Bird-eye view' instead of the current 'perspective view'
- Calculate dynamic distance
- Solve the issues (described in previous slide) and optimisation, wherever necessary (according to the changing norms or situation)
- Considering walls and partitions between two people as sufficient condition for social distancing
- Alert the people violating the norms; through some notification, so that they can take necessary action for their safety.

Appendix

Functions:

```
def detect_person_only(r, acc):
    r_person = {}
    l_rois = []
    l_class_ids = []
    l_masks = []
    l_scores = []
    for (rois, class_ids, masks, scores) in zip(r['rois'], r['class_ids'],
    r['masks'].T, r['scores']):
        if class_names[class_ids] == 'person' and scores >= acc:
            l_rois.append(rois)
            l_class_ids.append(class_ids)
            l_masks.append(masks)
            l_scores.append(scores)
    r_person['rois'] = np.array(l_rois, dtype=np.int32)
    r_person['class_ids'] = np.array(l_class_ids, dtype=np.int32)
    r_person['scores'] = np.array(l_scores, dtype=np.float32)
    r_person['masks'] = np.array(l_masks, dtype=np.bool).T

    return r_person

# define random colors
def color_func(N, isViolateVect):
    colors = [tuple([255,100,0]) for _ in range(N)]
    for i in range(0,N):
        if isViolateVect[i] == True:
            colors[i] = tuple([0,0,255])
    return colors

#apply mask to image
def apply_mask(image, mask, color, alpha=0.5):

    for n, c in enumerate(color):
        image[:, :, n] = np.where(
            mask == 1,
            image[:, :, n] * (1 - alpha) + alpha * c,
            image[:, :, n]
        )
    return image

#take the image and apply the mask, box, and Label
def display_instances(image, boxes, masks, ids, names, scores, isViolateVect, centroids, pairs):
    n_instances = boxes.shape[0]
    colors = color_func(n_instances, isViolateVect)
    if not n_instances:
        print('NO INSTANCES TO DISPLAY')
    else:
```

```

    assert boxes.shape[0] == masks.shape[-1] == ids.shape[0]
for i, color in enumerate(colors):
    if not np.any(boxes[i]):
        continue
    y1, x1, y2, x2 = boxes[i]
    cX, cY = centroids[i]
    h = pairs[i]
    label = names[ids[i]]
    score = scores[i] if scores is not None else None
    caption = '{} {:.2f}'.format(label, score) if score else label
    mask = masks[:, :, i]
    image = apply_mask(image, mask, color)
    image = cv2.rectangle(image, (x1, y1), (x2, y2), color, 1)
    image = cv2.circle(image, (cX, cY), 5, color, 1)
    image = cv2.line(image, tuple(h[0]), tuple(h[1]), (0, 0, 255), 2)
    image = cv2.putText(image, caption, (x1, y1), cv2.FONT_HERSHEY_COMP
LEX, 0.32, color, 1)
    return image

def centroid_calc(point_list):
    centroid_list = []
    for box in point_list:
        y1, x1, y2, x2 = box
        width, height = x2 - x1, y2 - y1
        c_x, c_y = width // 2, height // 2
        centroid_list.append((c_x + x1, c_y + y1))
    return centroid_list

```

Main Program:

```

frame_count = 0
frames = []
min_distance = int(input("Enter the threshold(minimum distance to be ma
intained in Pixels) : "))
violations_num_people = set()
cap = cv2.VideoCapture('/content/TestVideo.avi')
fourcc = cv2.VideoWriter_fourcc(*'XVID')
new_width = 700
new_height = int((new_width / int(cap.get(3))) * int(cap.get(4)))
out = cv2.VideoWriter('/content/out.mp4', fourcc, int(cap.get(5)), (ne
w_width , new_height))
total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
pbar = tqdm(total=total_frames, position=0, leave=True)
while(cap.isOpened()):
    violate = set()
    pairs = list()
    isViolateVect = list()
    ret, frame = cap.read()

    if ret is True:

```

```

frame = imutils.resize(frame, width=new_width, height=new_height)
frame_count += 1
frames.append(frame)
if len(frames) == batch_size:
    results = model.detect(frames, verbose=0)
    for i, item in enumerate(zip(frames, results)):
        frame = item[0]
        r = item[1]
        r = detect_person_only(r, 0.80)
        centroids = np.array(centroid_calc(r['rois']))
        D = dist.cdist(centroids, centroids, metric="euclidean")
        for i in range(0, D.shape[0]):
            for j in range(i+1, D.shape[1]):
                if (D[i, j] > 0) and (D[i, j] < min_distance):
                    pairs.append([tuple(centroids[i]), tuple(centroids[j])])

                    violate.add(i)
                    violate.add(j)
                else:
                    pairs.append([[0,0], [0,0]])
        for i in range(0, D.shape[0]):
            if i in violate:
                isViolateVect.append(True)
            else:
                isViolateVect.append(False)
        violations_num_people = violations_num_people.union(violate)

    frame = display_instances(frame, r['rois'], r['masks'], r['class_ids'], class_names, r['scores'], isViolateVect, centroids, pairs)
    text = "Social Distancing Violations: {}".format(len(violations_num_people))
    frame = cv2.putText(frame, text, (10, frame.shape[0] - 25),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.85, (0, 0, 255), 3)
    out.write(frame)
    frames = []
    pbar.update(1)
    # Clear the frames array to start the next batch
else:
    break

cap.release()
out.release()
pbar.close()

```

Bibliography

1. https://wwwnc.cdc.gov/eid/article/26/8/20-1093_article
2. https://github.com/matterport/Mask_RCNN
3. <http://www.robots.ox.ac.uk/ActiveVision/>