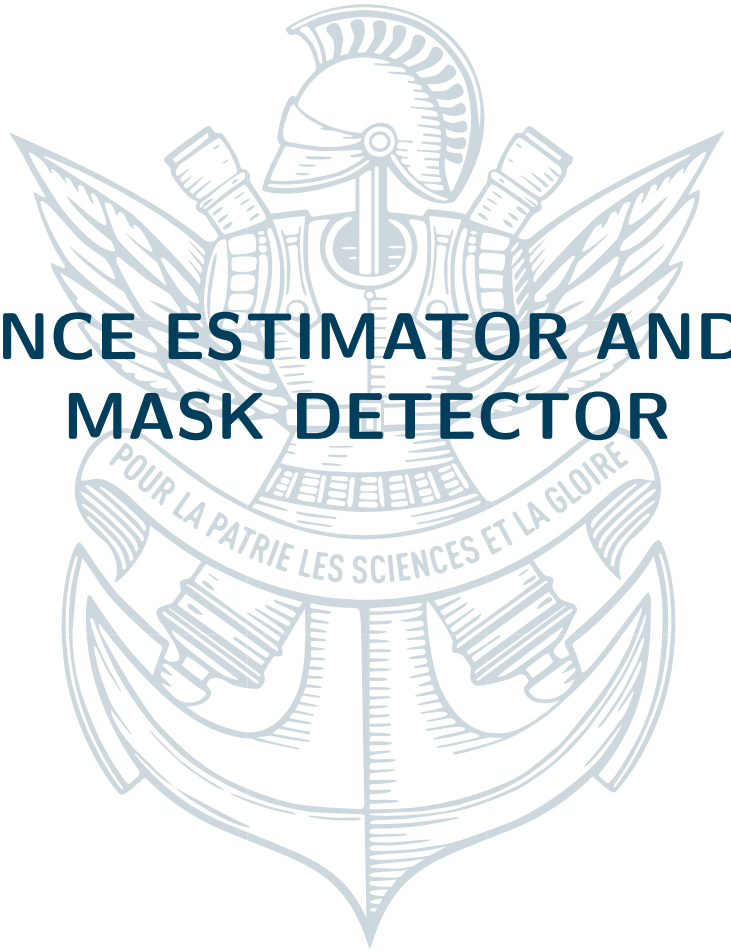

DISTANCE ESTIMATOR AND FACE MASK DETECTOR

December 2020

Celine Hajjar, Haolin Pan

CONTENTS

1 INTRODUCTION

As Covid-19 continues to spread across the world, health experts agree that reducing close face-to-face contact with others is the best way to limit contamination. This can be done by maintaining a safe distance between people (of at least 1m by CDC[1] guidelines) and by wearing a mask in public.

In fact, social distancing interventions can give communities vital time to strengthen health-care systems and restock medical supplies, so it is essential that people comply. Unfortunately, many people don't, and it is not possible to always verify that these measures are respected. For this reason, automated systems can be of great help. They can even be integrated into "RoboCops" that are being deployed to aid the police in patrolling public areas.

In this project, we present a model that processes images to give important information regarding social distancing measures. First, it detects all the persons present, it calculates the distance between them, and it verifies whether those people are wearing masks.

This report details the methods we used to achieve the desired results. In section 2, we state all the assumptions we made throughout our work, and in the subsequent sections we explain every part of the image processing: first we give a background view of the models used (mask R-CNN and Haar Cascade Face Classifier) then we describe our implementation and the results we obtained.

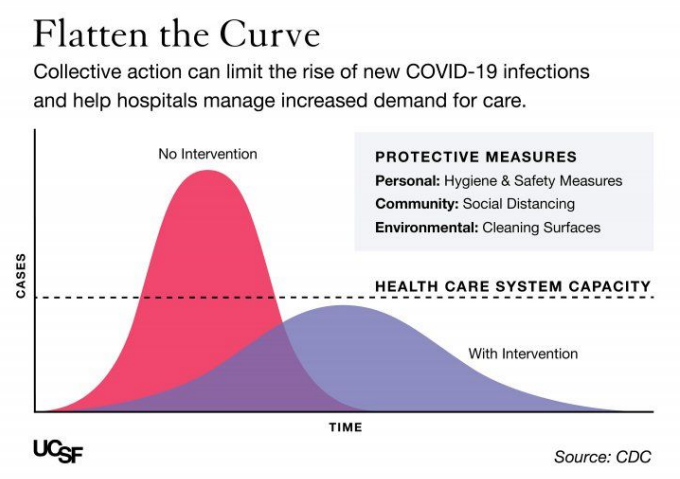

Figure 1: Social distancing and masks help "flatten the curve". Source: CDC

[1]Centers for Disease Control and Prevention

2 ASSUMPTIONS

To ensure the feasibility of our models and facilitate our research, we made the following assumptions to the situation of the application of this project.

1. The people in the photo are all adults of height $170cm - 180cm$ and are all in standing pose. Else, we're very likely to have a wrong estimation of his height and his position.
2. The photo is shot by a camera with an angle of view around 90 degree.
3. The people in the photo are standing on the same surface
4. The people's bodies are not blocked by obstacles

3 HUMAN DETECTION

3.1 State of the Art: Mask RCNN

In order to retrieve the positions of people in 3D space, we first need to detect their positions and their contours in the 2D picture. To accomplish this task, we decided to apply a pre-trained *mask-RCNN* (Region based Convolutional Neural Network)[1], which is an extension of *Faster-RCNN* [3].

In short, *Faster-RCNN* has two parts. The first one is a Region Proposed Network, which detects the region where there could be an object of interest. The second part is the *fast-RCNN* which extracts the regional features and performs classification and bounding-box regression on them. *mask-RCNN* has the additional ability of learning the contours of the object, which is done by adding a new neural network to fit the mask of the object. The design of *mask-RCNN* is shown in the Figure 1.

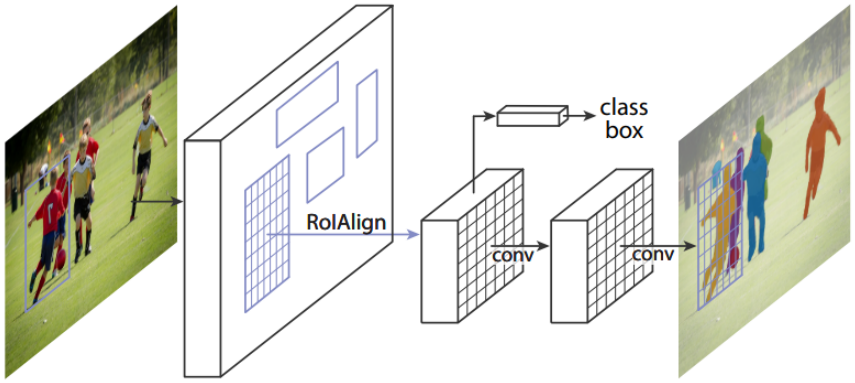

Figure 2: Model design of Mask RCNN

3.2 Application

The pre-trained model we have chosen was trained on *COCO* set, which has humans as objects of interest. So it's easier to make use of it in our projects. However, to test and tune the model, we found another data set, *PennFudanPed*, which is a data set of pedestrians. The results are very satisfactory.

Figure 2 and Figure 3 show an example of the results. Generally, the model is already very accurate, but it could fail when the bodies of people are covered by some obstacle or

Figure 3: The original photo

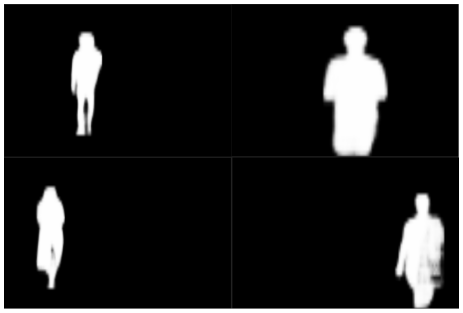

Figure 4: The mask of each object

the people are too far from the camera. There also exist cases where other objects, like road lamps and bicycles, are predicted as pedestrians. So we set up a threshold of reliability for our predictions. A prediction is validated only when its reliability reaches this threshold. After tuning on the *PennFudanPed* data set, we set this threshold to 0.75.

4 DISTANCE CALCULATION

Now, with the people in the picture detected, we're supposed to project them into 3d space. By the assumption, the height of the detected people is around 1.70m. At the same time, with the bounding boxes and contours of people, we can estimate the height of people in pixels. Using these two facts, we can translate the distance in pixels to the distance in meter. The formulas are shown below, with Xp and Yp the coordinates of the person detected in pixels.

$$x = Xp \times \frac{1.70}{hp} \tag{1}$$

$$y = Yp \times \frac{1.70}{hp} \tag{2}$$

Furthermore, if we can precise the angle of view of the camera (which is a constant for all photos), we can calculate the distance between the person and the camera, by equation 1, with Hp being the height of the photo in pixels, hp being the person's height in pixels (the height of the person's bounding box).

$$d = \frac{1}{f} \times \frac{Hp \times 1.70}{hp} \tag{3}$$

Adding this third dimension, we retrieve the positions of people in 3D space. Here, f is a constant factor determined by the camera. $f = 2 \times \tan(\alpha/2)$, where α is the angle of view of the camera. Figure 5 shows the relations between these variables.

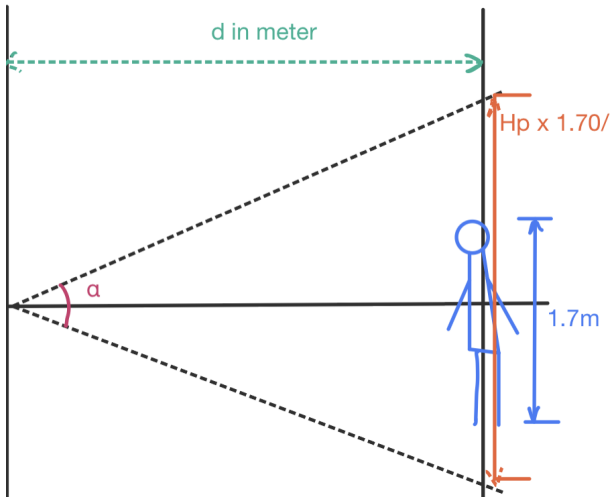

Figure 5: Focal length and distance to camera

4.1 Focal Length Estimation

In this section, we propose a set of experiences to estimate the factor f. Equation 3 above defines a direct relation between the estimated distance d and f. Thus, if we know the exact distance separating the individual from the camera, we can assess the quality of the estimation of f (more specifically, the quality of the assumptions made).

By running a series of tests with images we took ourselves (with an iPhone X), we found that the values of f that generated the closest estimations to the real distances ranged from 0.95 to 0.98. With four images where the real distance was $2m80$, the average focal length was 0.963. For similar tests run with a real distance = 2m, the average focal length was 0.967. Thus, we fixated on the value 0.965 as the estimated focal length for our camera. It is the value we use for all later computations.

4.2 Plane Projection

After getting the positions of each person: (x_i, y_i, d_i), we can directly calculate the distance between any two of them. However, according to another assumption we made, the people in this scene stand on the same surface. So when we have more than 3 points, we could perform a linear regression to estimate the equation of this common surface. Then, we project all points to this surface, so that we can get the relative position of the people to the camera on this surface. Figure 6 and Figure 7 show an example of result. We can see that in *Figure* 7, the relative positions of the six people in the scene are clearer and more meaningful.

Figure 6: Original photo

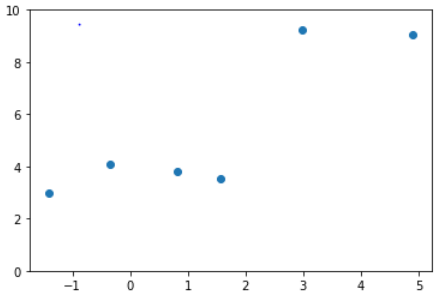

Figure 7: Projections of people detected to horizontal surface

5
FACE AND MASK DETECTION

The second part of our project is to check whether the people identified in the photo are wearing masks. To do that, we divide the task into two parts:

1. For every human box detected, we find all the faces it contains.
2. For every detected face, we check whether it has a mask.

5.1 Face Detection

For the face detection part, we implement a Haar Cascade Classifier.

5.1.1 • Background

Object Detection using Haar feature-based cascade classifiers is a machine learning-based method proposed by Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001[4]. It consists in training a cascade function on several positive and negative images, in order to detect three main types of features (figure 8):

- Edge Features (or two-rectangle features) are evaluated by the difference between the sum of the pixels within two rectangular regions.
- Line Features (or three-rectangle features) compute the sum within two outside rectangles subtracted from the sum in a center rectangle.
- Four-rectangle Features compute the difference between diagonal pairs of rectangles.

In figure 9, we can see that it is relevant to search for features around the eye and nose area. The first feature selected portrays the change in intensity between the region of the eyes and the region of the nose and cheeks. The second feature selected shows the separation of the eyes by the nose bridge. However, feature selection wouldn't be relevant in the lower parts of the face, as there are no clear changes in intensity. In order to optimize this operation and avoid searching for all possible features, the AdaBoost learning algorithm is used to choose which features are the most relevant for the classification.

Also, instead of applying all the features on a certain region in the photo, Viola and Jones introduced the concept of Cascade Classifiers. Features are grouped into different stages of classifiers (with less features in the early stages) and applied one-by-one. If a region fails the first stage, it is immediately discarded. If not, it advances to the next stage and the process is repeated. The window that passes all stages is a classified as a face region.

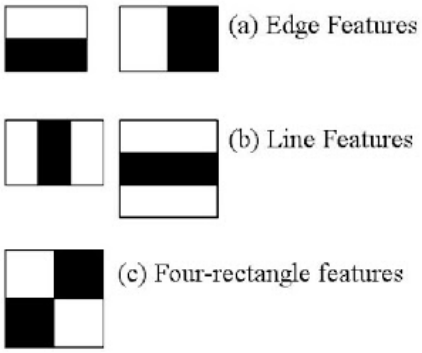

Figure 8: Haar Features

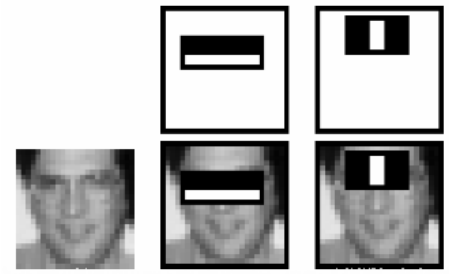

Figure 9: Feature Detection in a Face

5.1.2 • Implementation

In our project, we used a pre-trained Haar Cascade Face Classifier provided by OpenCV. In some test cases, the classifier failed to detect the face, probably for two reasons: either the face isn't clear so the features aren't detectable (for example, if the person is wearing reflective sunglasses and her hair is covering her face), or the presence of masks is negatively affecting the model's performance. In figure 10, the person on the left embodies both cases: unsurprisingly, the face isn't detected, and neither is the mask.

One way to solve that problem is to manually define an area that we consider to be a face, and detecting a mask within it. Given the human box, if no face was detected, we cut the bounding box 1/6 by height and 1/2 by width, and suppose that the resulting rectangle represents a face. Mask detection is then applied on this area (figure 11).

Figure 10: Classifier misses a face

Figure 11: Manual Face Detection

By running the code with the *minNeighbors* argument set to 5, we obtained false positive results for face detection. An example is shown in figure 12. This problem is solved by augmenting the minimum number of neighbors a potential rectangle should have in order to be retained. With *minNeighbors* set to 10, the same image gives the result shown in figure 13, which is a correct prediction.

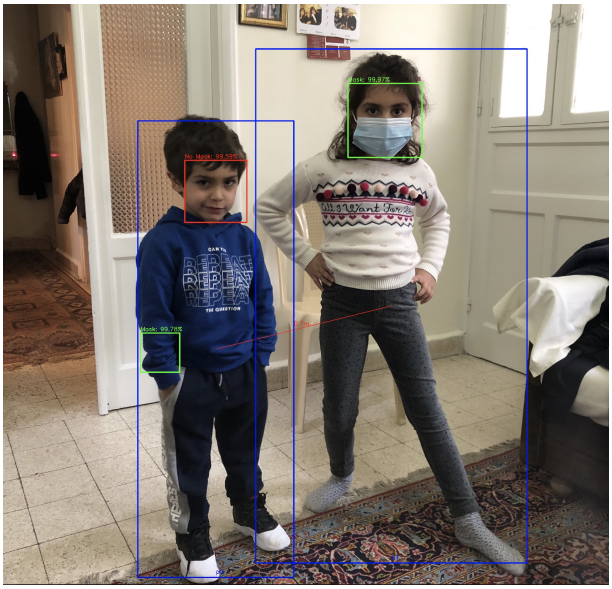

Figure 12: False Positive, minNeighbors = 5

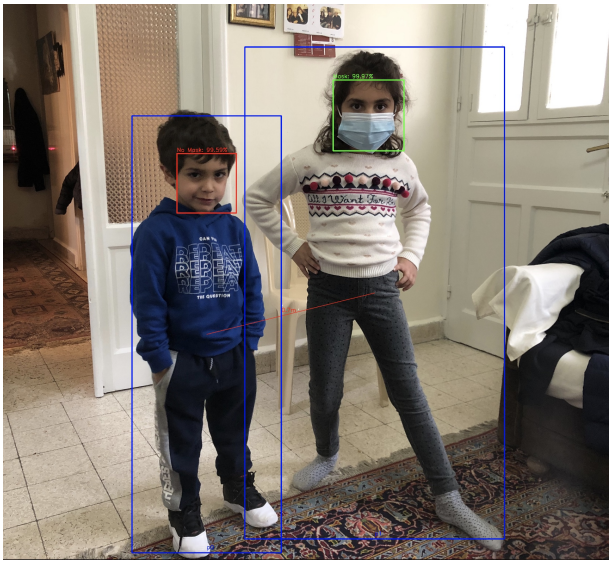

Figure 13: True Negative, minNeighbors = 10

5.2 Mask Detection

For mask detection, we imported a model that was trained and published to Github by Chandrika Deb. The repository can be found **here**. The characteristics of the model, as presented by the author, are the following:

Training Data set The data set used for training consists of 3835 images belonging to two classes: with_mask (1916 images) and without_mask (1919 images).

Model Architecture The model is based on the *MobileNetV2* architecture, designed to effectively maximize accuracy while being computationally efficient[2].

Training Results The model gave a F1-score of 0.93 after training via `tensorflow-gpu==2.0.0`. The classification report is presented in figure 14.

	precision	recall	f1-score	support
with_mask	0.99	0.86	0.92	383
without_mask	0.88	0.99	0.93	384
accuracy			0.93	767
macro avg	0.93	0.93	0.93	767
weighted avg	0.93	0.93	0.93	767

Figure 14: Classification Report

6 INTEGRATION

After integrating all the functionalities presented in the previous sections, we built a system that processes images following the general algorithm presented in the first subsection below. Results are shown in the second subsection.

6.1 Pipeline

Input: A photo
Output: The same photo, with bounding boxes precising the positions and sizes of humans in the photo, the distance between each person, the faces detected, and the estimated probability that the person is wearing a mask.

1. Apply the *mask RCNN* to detect the people and their bounding boxes. Filter out any detected object with reliability lower than a certain threshold.

2. Project the positions of people into 3D space and calculate the euclidean distances between them. If the distance is smaller than the determined safe distance, alert the user.

   If the number of detected people is larger than 3, estimate the function of the plane they're standing on. Then project them onto the plane and calculate the distances between their projections.

3. For each bounding box, apply the face detection model. If there is exactly one face detected, continue.

   If there is no face detected, manually define a rectangle that has 1/6th the height of the bounding box and 1/2 its width, and assume that it defines a face.

4. For each face detected, apply the mask detection model.

6.2 Results

We have two kinds of outputs: the image with the annotations, and the detected information printed out in the command prompt. An example output is shown in Figure 15.

Note that in this example, the individuals were placed at exactly 2.4m from the photographer, and 0.7m from each other. The predictions are close to the expected results.

7 CONCLUSION

In this project, we have accomplished a system to detect humans, to retrieve their relative positions, and to tell if they're masked, by applying a group of deep learning models. This system could be used in some public areas like restaurants, to ensure that the number of clients is smaller than the threshold, and the clients are wearing masks and keeping social distance. Generally speaking, its performance is quite satisfactory.

Here I want to present some further work to do. Firstly, we observe that our face detection models have a lower recall in detecting faces with masks. This could be caused by the fact our model is trained on faces without mask. Secondly, our distance estimation has an about 10% uncertainty, because of the assumption we made about human height. This uncertainty is negligible in our situation of application. However, in situations like autonomous vehicles, this uncertainty could be very dangerous.

In conclusion, our model is reliable in situations with the assumptions we have made. But there is still a lot of work to do to remove these assumptions.

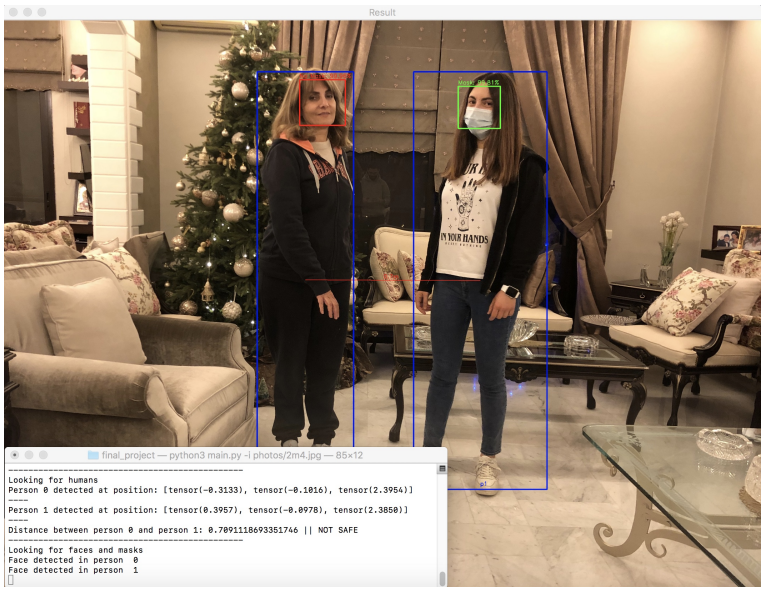

Figure 15: Sample Output

REFERENCES

[1] Piotr Dollár Ross Girshick Kaiming He, Georgia Gkioxari. Mask r-cnn. *CVPR*, 2017.

[2] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019.

[3] Ross Girshick Jian Sun Shaoqing Ren, Kaiming He. Faster r-cnn: Towards real-time object detection with region proposal networks. *CVPR*, 2015.

[4] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I, 2001.