Public Surveillance for preventing spread of Covid-19

By

Jenil Mehta
17bit048

NIRMA UNIVERSITY
INSTITUTE OF TECHNOLOGY
NAAC ACCREDITED 'A' GRADE

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
Ahmedabad 382481

Public Surveillance for preventing spread of Covid-19

Minor Project Report

Submitted in partial fulfillment of the requirements

For the degree of

Bachelor of Technology in Computer Engineering/Information Technology

By

Jenil Mehta
17bit048

Guided By
Dr Priyanka Sharma
[DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING]

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
Ahmedabad 382481

CERTIFICATE

This is to certify that the Minor Project entitled “Public Surveillance for preventing spread of Covid-19”submitted by Jenil Mehta (17bit048) , towards the partial fulfillment of the requirements for the degree of Bachelor of Technology in Computer Engineering/ Information Technology of Nirma University is the record of work carried out by him/her under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination.

Dr Priyanka Sharma
Associate Professor
Department of Computer Science & Engg.,
Institute of Technology,
Nirma University,
Ahmedabad

Dr. Madhuri Bhavsar HOD
Dept. of Computer Science & Engg.,
Institute of Technology,
Nirma University,
Ahmedabad

ACKNOWLEDGEMENT

We would like to express our deepest appreciation to all those who provided us the possibility to complete this report. We acknowledge with thanks, the support rendered by Dr . Priyanka Sharma , under whose aegis we were able to complete the
task in a given period of time. In addition, we might want to offer thanks toward senior understudies and different resources for familiarizing us with the point too for the assistance in the journey of the project. Moreover, we like to thank the individuals in our investigation, who have promptly shared their important time in the midst of the path toward talking. We also thank our batchmates, who have maintained us all through the entire system, both by keeping us pleasant and helping us amassing pieces. We will be grateful dependably for your reverence.

ABSTRACT/ Outline

The project was all about to develop methods to prevent the spread of Covid-19.

The project was divided into 2 parts:

1. Social monitoring system
2. Spit detector.

The first part can be done using YOLOV3 and the second part can be solved by pose estimation.

CONTENTS

Chapter 1	**Introduction**	**6**
Chapter 2	**Challenges**	**7**
Chapter 3	**Dataset**	**7**
Chapter 4	**Preprocessing** **This chapter will contain the details of project work.**	**7**
Chapter 5	**Social Monitoring System**	**7**
	5.1 Bird's eye view	**7**
	5.2 Yolo	**8**
	5.3 Flow Chart	**10**
	5.4 Evaluation	**11**
Chapter 6	**Method for spit detection**	**13**
Chapter 7	**Implementation**	**14**
Chapter 8	**Dynamic Time Warping**	**15**
Chapter 9	**References**	**16**

1. Introduction

The Problem is divided into 2 phase:

Social distancing has proved to be a very effective measure to slow down the spread of the disease in the fight against the coronavirus. Some of the employees in the manufacturing sector also have to go to work every day to ensure that our basic needs are met. So by analyzing real-time video streams from the camera, I have created an AI-enabled social distance detection system that can detect whether individuals maintain a safe distance from each other.

Spit Detection: In order to contain the spread of covid-19 I am working on a project that can detect whether a person is spitting in a given video or not.

2. Challenges

The task is complicated as we don’t have a top view of the frame. Top view is important as we won’t get an exact idea of people who are in a line. So getting a bird's eye view will help to get a better view of the video.

For spit detection the task is unique in itself as I have no available dataset. So I have created a small dataset for spit detection. We also have many complicated methods to classify an action of spitting in a frame.

3. Dataset

We used a pre-trained YoloV3 model. The model is trained to classify many objects in which people are also included. So we didn’t train a model from scratch but decided to proceed using YoloV3.

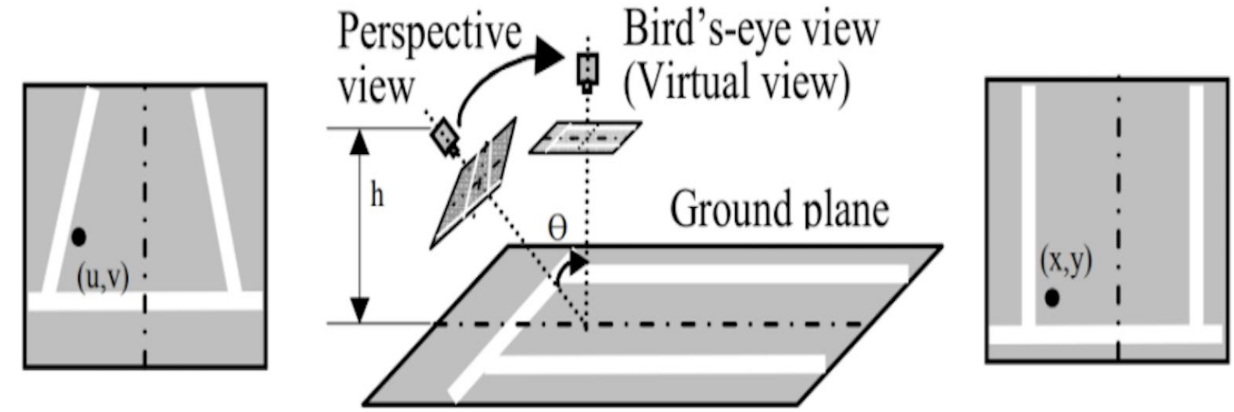
Created my own custom dataset for spit detection.

4. Preprocessing

We did simple preprocessing of converting video into frames. We used Opencv to achieve this task. The fps is a parameter and depending upon the computing power the fps is limited.

5. Learning Method for social monitoring system:

5.1 Bird eyes view : A bird's-eye see is a raised perspective on an item from above, with a viewpoint like the spectator were a feathered creature, frequently used in the making of outlines, floor plans, and maps.

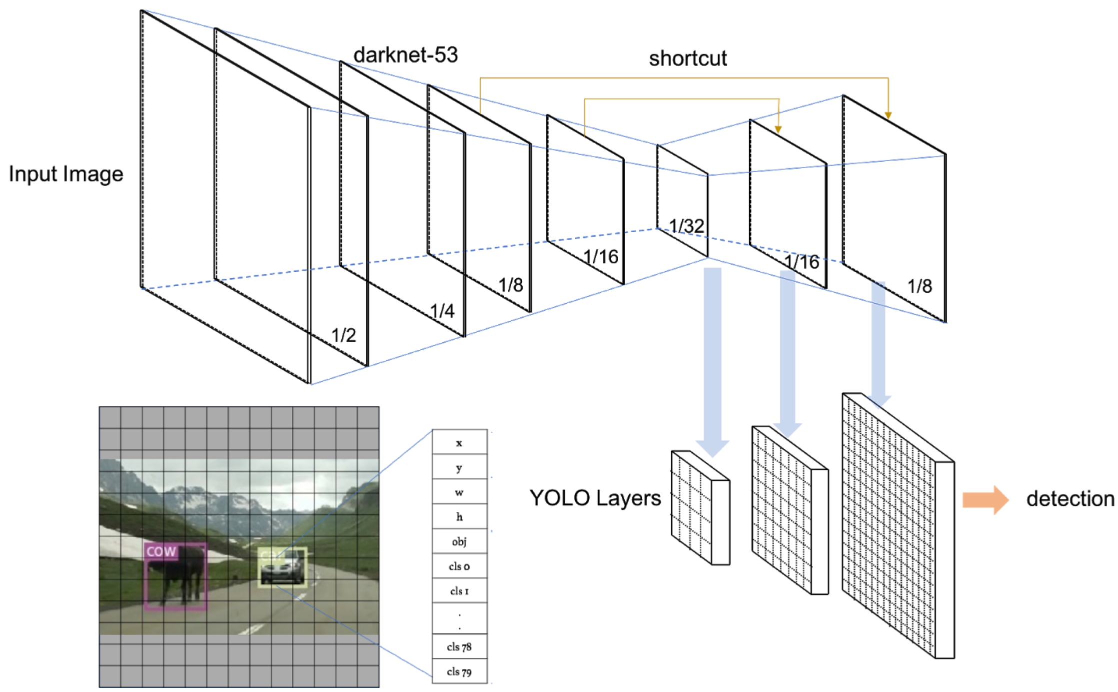


(a) Perspective view image
(b Perspective transformation
(c)Bird's-eye view image

Fig. 1. Illustration of perspective transformation in a parking lot scene

5.2 Yolo Algorithm:

Object Detection is a complex task in computer vision where you can specify what objects are there in the image and also where the objects are there in the image. It uses Convolutional Neural Network for detecting the object in real time. The algorithm adds the entire image to a single neural network, then separates the image into regions and forecasts bounding boxes and probabilities for each region. These bounding boxes are weighted by the probabilities predicted.

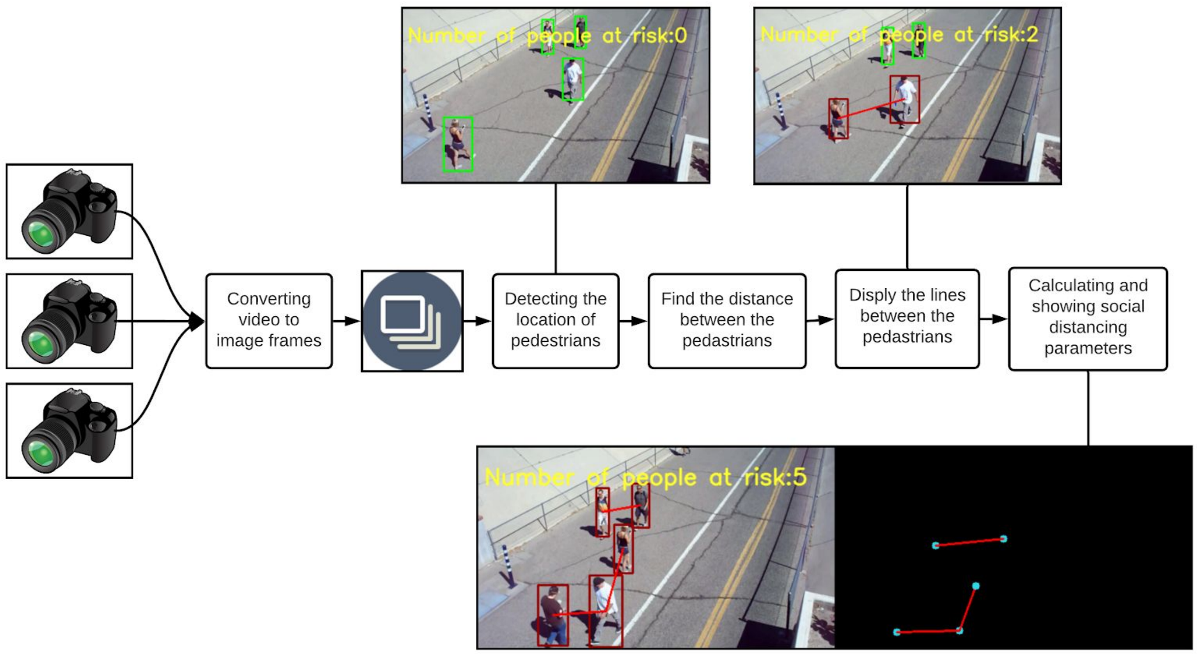


We first trained the model for pedestrian detection. We then convert the perspective into bird's eye view also known as the top-down view. After detecting the objects using the YOLO algorithm we will be mapping the points in current perspective with the bird's eye view perspective.

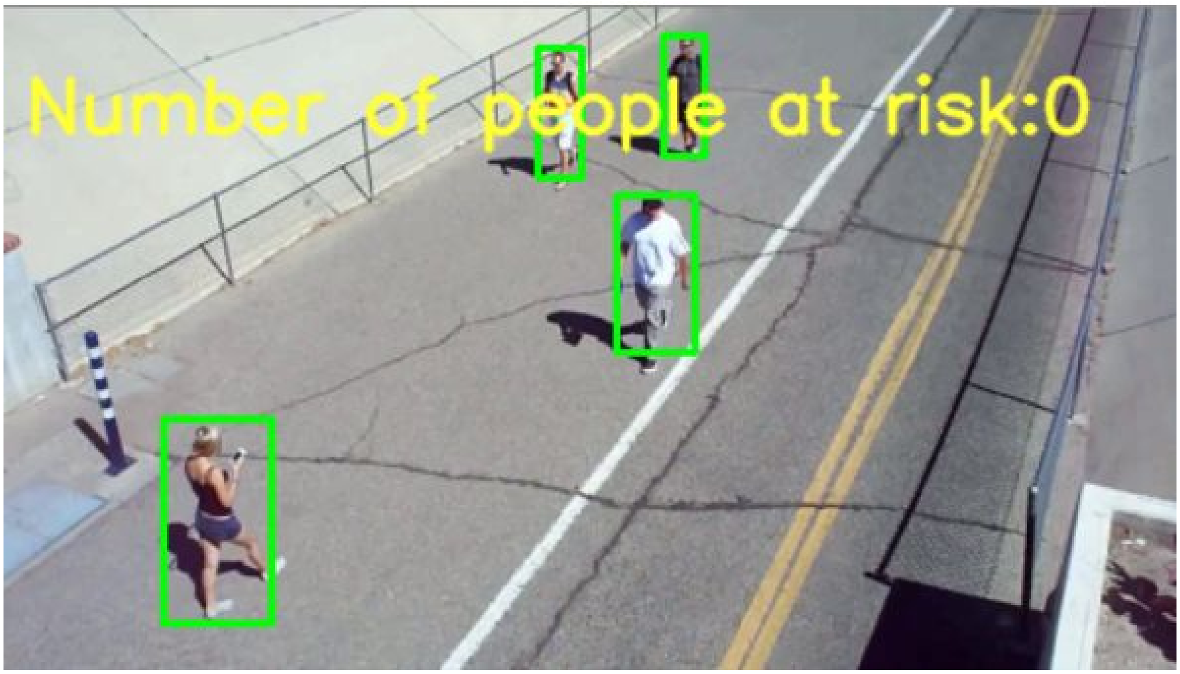
After doing these steps, we will calculate the distance between two points in BEV perspective and all the points that are close to each other will be highlighted.

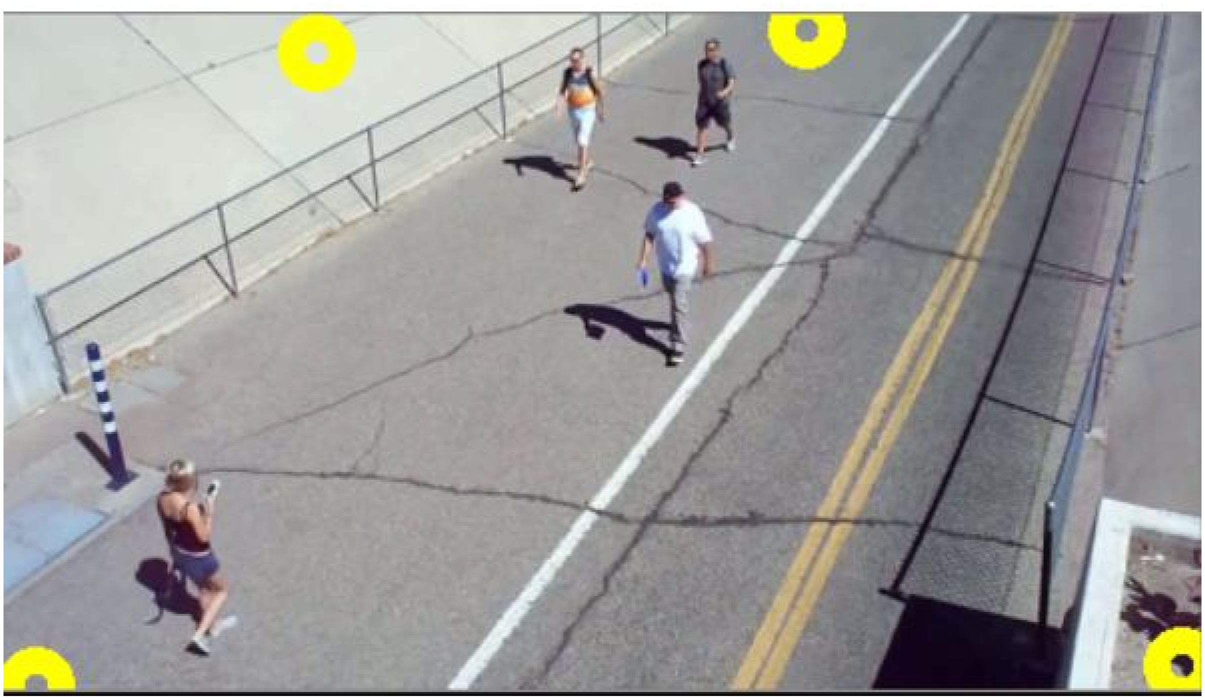
Number of people that are at risk would be displayed in the frame.

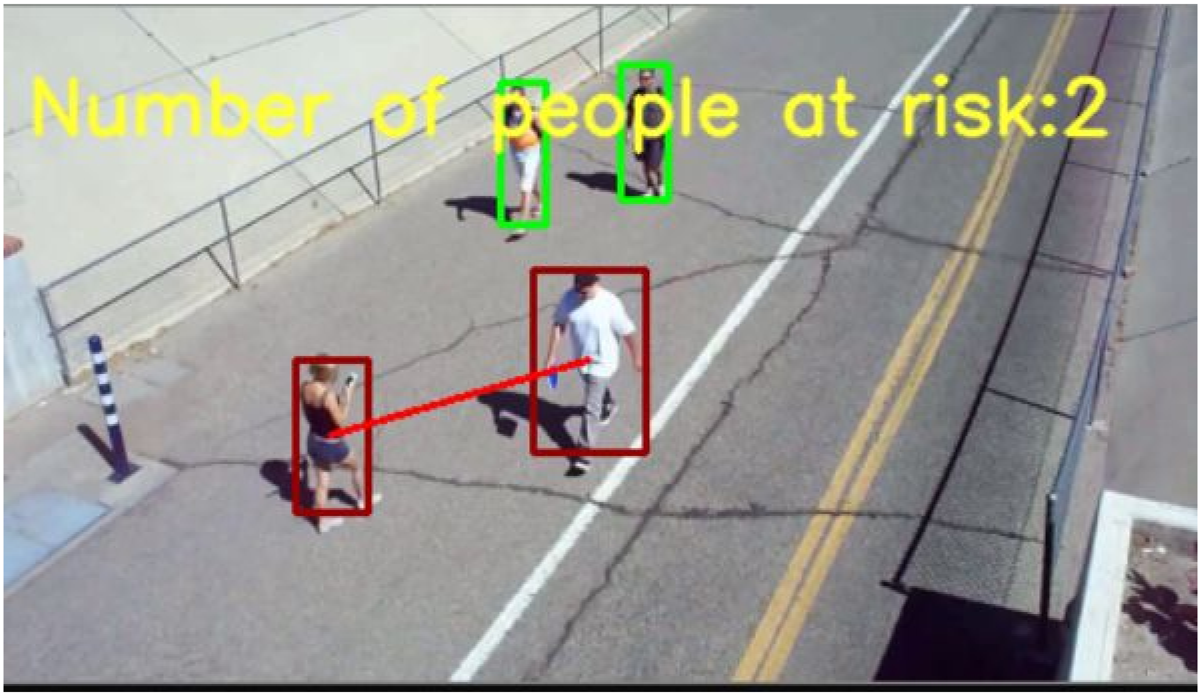
5.3 Flow Chart



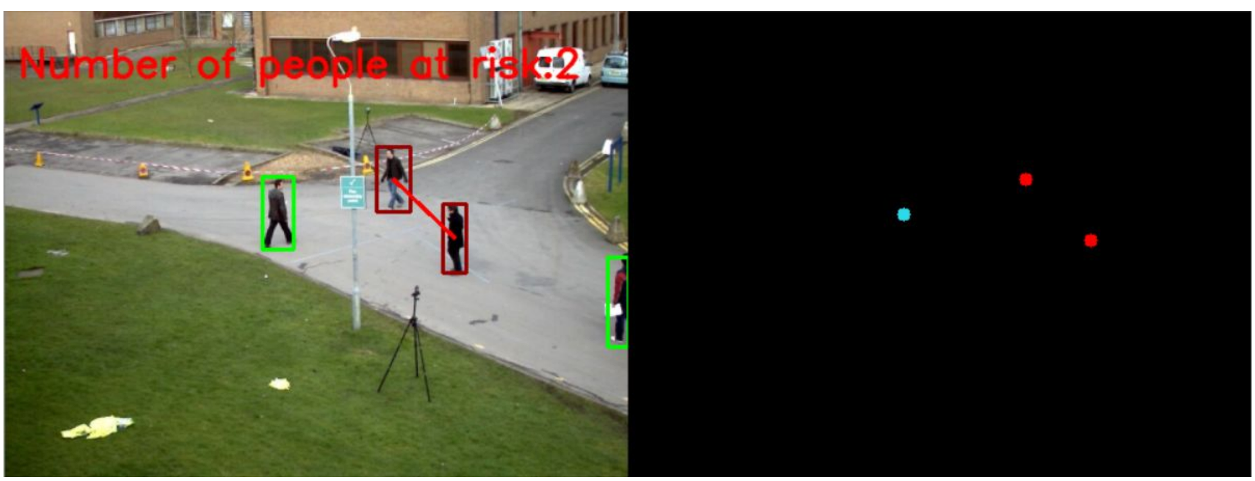
5.4 Evaluation:



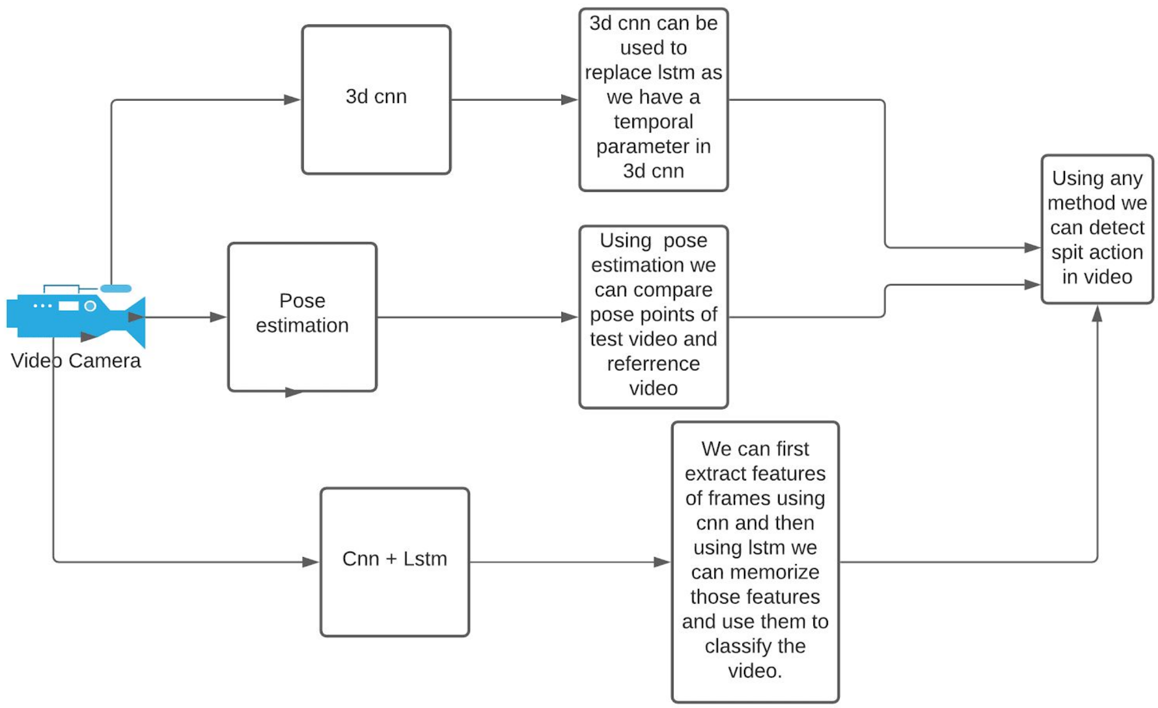



Number of people at risk:2


Number of people at risk:5


Number of people at risk:2

6. Methods for Spit detection:



7. Implementation:

Pose-Estimation Method:

This method can be used in place of Cnn or Cnn + LSTM as neural networks need large amounts of data in order for them to work perfectly. As the dataset was custom and not at all large so the model was not performing well. Following were some issues that were encountered by me:

1. Due to very low data the model was overfitting itself.
2. In the spit dataset the videos were 5 seconds long but most of the videos were normal so the model would not be able to identify spitting videos as accurately as normal videos.

The action of Spitting differs from normal action in terms of it's pose. So what we can do is that we can first get key-points of normal video and spit video. After that we can test our model on test video by following algorithm:

1. Get key-points of each second of the test video.
2. Compute Edit distance(Dynamic Time Wrapping) between test points-normal_pose points and test points-spit-video points.
3. The one with low edit distance will be the class of the current second of test video.
4. Repeat the process until the end of the video.

8. Dynamic Time Warping:

Dynamic Time Warping is used to compare the similarity or calculate the distance between two arrays or time series with different length.

A concise explanation of DTW from wiki,

> **In time series analysis, dynamic time warping (DTW) is one of the algorithms for measuring similarity between two temporal sequences, which may vary in speed. DTW has been applied to temporal sequences of video, audio, and graphics data — indeed, any data that can be turned into a linear sequence can be analysed with DTW.**

```
int DTWDistance(s: array [1..n], t: array [1..m]) {
    DTW := array [0..n, 0..m]

    for i := 1 to n
        for j := 1 to m
            DTW[i, j] := infinity
    DTW[0, 0] := 0

    for i := 1 to n
        for j := 1 to m
            cost := d(s[i], t[j])
            DTW[i, j] := cost + minimum(DTW[i-1, j  ],    // insertion
                                        DTW[i  , j-1],    // deletion
                                        DTW[i-1, j-1])    // match

    return DTW[n, m]
}
```

8 References:

https://nikolasent.github.io/opencv/2017/05/07/Bird's-Eye-View-Transformation.html

https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b?gi=69337abffe38

https://towardsdatascience.com/dynamic-time-warping-3933f25fcdd

https://nanonets.com/blog/human-pose-estimation-2d-guide/

https://innovationincubator.com/human-pose-comparison-and-action-scoring-using-deep-learningopencv-python/