“SOCIAL DISTANCE VIOLATION USING DEEP LEARNING”

A project report submitted to

MALLA REDDY UNIVERSITY

in partial fulfillment of the requirements for the award of degree of

BACHELOR OF TECHNOLGY
in
COMPUTER SCIENCE & ENGINEERING (AI & ML)

Submitted by

M.SATYA BABU	**: 2211CS020288**
M.DILEEP REDDY	**: 2211CS020289**
M.LAHARI	**: 2211CS020291**
M.ARSHITH	**: 2211CS020292**
M.RAJESH	**: 2211CS020293**

Under the Guidance of
PROF.A.KALYANI

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (AI & ML)2024

MALLA REDDY UNIVERSITY
(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

MALLA REDDY UNIVERSITY
(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

COLLEGE CERTIFICATE

This is to certify that the work titled "Social distance violation Using Deep Learning" is a bonafide record of the application development project submitted by M. Satyambabu (2211CS020288), M. Dileepreddy (2211CS020289), M. Lahari (2211CS020291), M. Arshith (2211CS020292), M. Rajesh (2211CS020293) of B.Tech III Year, I Semester, Department of Computer Science and Engineering (AI & ML), during the academic year 2024-2025. The results embodied in this report have not been submitted to any other university or institute for the award of any degree or diploma.

PROJECT GUIDE
PROF.A.KALYANI

DEAN OF TH DEPARTMENT
Dr.Thayyaba Khatoon

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We sincerely express our deep sense of gratitude to, **Prof**.**A.Kalyani** for his valuable guidance, constant encouragement and co-operation during all phases of the project.

We are greatly indebted to our Project Mentor, **Prof.S.Ramesh kumar** for providing his valuableadvice, constructive suggestions, positive attitude, and encouragement without which it wouldnot been possible to complete this project.

It is a great opportunity to render our sincere thanks to **Dr. Thayyaba Khatoon** Head of the Department of computer science and engineering (AIML) for her timely guidance, highly interactive attitude which helped us a lot in successful execution of the project.

Abstract

The World Health Organization has claimed the spread of coronavirus as a global pandemic because of the increment in the expansion of coronavirus patients detailed over the world. To hamper the pandemic, numerous nations have imposed strict curfews and lockdowns where the public authority authorized that the residents stay safe in their home during this pandemic. Various healthcare organizations needed to clarify that the best method to hinder the spread of the virus is by distancing themselves from others and by reducing close contact. To flatten the curve and to help the healthcare system on this pandemic. To contemplate data-driven models and numerical models which are consistently the most favored decision. In the fight against the coronavirus, social distancing has proven to be an effective measure to hamper the spread of the disease. As the name suggests, it implies that people are suggested that they should maintain physical distance from one another, reduce close contact, and thereby reduce the spread of coronavirus.

CONTENTS

CHAPTER 1

In the course of 2020, the world went through the pandemic of coronavirus also known as Covid-19. This is a communicable disease that spreads very fast from close contact with the person tested positive for Covid-19. The virus spreads through infected secretions like saliva and respiratory secretions which are expelled during the respiratory processes and also when an infected person is coughing or sneezing as well as when the infected individual talks or sings the droplets expelled from the person's mouth can cause infection to a healthy individual. Preventing the spread of this disease has been one of the most important things in these difficult times. To do so, many guidelines in the form of precautions were released by World Health Organization out of which, social distancing is one of the most salient and effective. It is very important for all the citizens of the world to give a hand in order to fight back this disease. Social distancing has proven to be an effective measure to hamper the spread of the disease. The system presented is for analyzing social distancing by calculating the distance between people in order to slow down the spread of the virus. This system utilizes input from video frames to figure out the distance between individuals to alleviate the effect of this pandemic. This is done by evaluating a video feed obtained by a surveillance camera. The video is calibrated into bird's view and fed as an input to the YOLOv3 model which is an already trained object detection model. The YOLOv3 model is trained using the Common Object in Context (COCO). The proposed system was corroborated on a pre-filmed video. The results and outcomes obtained by the system show that evaluation of the distance between multiple individuals and determining if rules are violated or not. If the distance is less than the minimum threshold value, the individuals are represented by a red bounding box, if not then it is represented by a green bounding box. This system can be further developed to detect social distancing in real-time applications.

1.1 Problem Definition.

The problem is to develop a computer vision and deep learning-based system for enforcing and monitoring social distancing measures in public spaces, such as malls, parks, workplaces, and transportation hubs. The primary goal is to mitigate the spread of infectious diseases, like COVID-19, by ensuring individuals maintain safe distances from each other.

Detection of Human Entities: Develop a deep learning model capable of accurately detecting and localizing human entities in real-time video streams or images.

Distance Estimation: Create algorithms that can estimate the distance between pairs of detected individuals based on their relative positions within the frame.

Social Distancing Rule Enforcement: Implement logic to determine whether individuals are maintaining an appropriate distance (e.g., 6 feet or 2 meters) from each other.

Real-time Monitoring: Continuously monitor video streams or images for compliance with social distancing guidelines.

Alerting and Notification: Develop a mechanism to alert authorities or responsible personnel when social distancing violations are detected.

Privacy Protection: Implement privacy measures to ensure that the system does not capture or store personal information, such as faces or other identifiable features.

Scalability: Ensure the system can handle various camera setups and adapt to different environments, including indoor and outdoor locations.

Accuracy and Robustness: Train the deep learning model to perform well in diverse lighting conditions, weather, and with varying camera angles and qualities.

Data Management: Create a system to store and manage video feeds and analysis results for later reference and auditing.

User Interface: Develop a user-friendly interface for operators to monitor and interact with the system, view violation reports, and configure settings.

Deployment and Integration: Deploy the system in relevant public spaces and integrate it with existing security or surveillance infrastructure.

Regulatory Compliance: Ensure that the system complies with local, state, and national regulations and guidelines related to privacy and surveillance.

Training and Maintenance: Establish a plan for regular model retraining to adapt to changing conditions and guidelines.

1.2 Objective Of Project

The objective of a project on social distancing using computer vision and deep learning is to develop a system that can automatically monitor and enforce social distancing guidelines in various settings, such as public spaces, workplaces, or events. This typically involves:

Detection: Using computer vision to detect people in a given area.

Distance Estimation: Determining the distances between individuals in real-time.

Alerts: Generating alerts or warnings when individuals are too close to each other, thereby violating Social distancing rules

Data Logging: Recording data for contact tracing and analysis of social distancing compliance over Time.

Adaptability: Developing a system that can be adapted to various camera setups and environments.

Privacy Considerations: Ensuring that the system respects privacy and doesn't infringe on individuals' rights.

Real-time Feedback: Providing immediate feedback to individuals and authorities to ensure compliance.

Such a system can be used to help curb the spread of contagious diseases like COVID-19 by reducing close contact between individuals. It can also serve as a valuable tool for public health authorities and organizations to enforce and monitor social distancing measures.

1.3 Scope Of Project

A deep learning-based project for detecting social distancing violations aims to monitor and identify instances where people fail to maintain safe distances in real-time within public spaces. The project involves using datasets with pedestrian annotations or custom data to train models for people detection and distance measurement. A model like YOLO or Faster R-CNN can be applied to detect individuals, while distance measurement between them uses homography transformation to convert pixel distances to

real-world metrics. Violations are flagged when people are closer than a specified threshold. For deployment, the system can run on edge devices or cloud platforms, supporting real-time processing and alerting. Evaluation metrics include detection accuracy and violation detection rates, as well as an analysis of false positives and negatives. Privacy compliance is essential, with anonymization and data retention policies built into the system. Key challenges involve handling occlusions, calibrating distances accurately across varying camera angles, and addressing ethical concerns related to surveillance. Future improvements could incorporate multi-camera tracking and broader anomaly detection for enhanced monitoring in large areas.

CHAPTER 2

ANALYSIS

2.1 Project Planning and Research

1. Research and Problem Formulation

- **Research Existing Work:** Explore related projects in object detection, distance estimation, and person tracking.
    - **Key Papers & Methods:**
        - **Object Detection**: Single Shot Multibox Detector (SSD), Faster R-CNN, YOLO (You Only Look Once).
        - **Person Detection**: OpenPose, PoseNet, or Mask R-CNN for improved accuracy in recognizing people.
        - **Distance Measurement**: Geometric transformation methods, such as homography to map camera perspective.
- **Define Metrics**: Define metrics for accuracy, false positive/negative rates, and real-time processing speed.
- **Dataset Exploration**:
    - Public datasets, such as **COCO** and **PASCAL VOC**, or **custom datasets** simulating environments (e.g., streets, offices).

2. Data Collection and Annotation

- **Data Gathering**:
    - Collect video footage or images from public datasets.
    - Consider capturing real-world data if feasible, respecting privacy and ethical concerns.
- **Data Annotation**:
    - Annotate images with bounding boxes for each person.
    - Use tools like LabelImg or VGG Image Annotator.
- **Data Augmentation**:
    - Apply transformations (scaling, rotation, flipping) to improve robustness.

3. Model Selection and Training

- **Model Architecture**:
    - Choose a pre-trained object detection model (e.g., YOLOv4/v5, Faster R-

CNN).

- **Distance Calculation**:
  - Use homography transformations to map 2D points to a real-world distance.
  - Set a threshold for minimum distance between people based on camera perspective.
- **Training the Model**:
  - Fine-tune the model on your dataset.
  - Use transfer learning if pre-trained weights are available for faster convergence.
- **Hyperparameter Tuning**:
  - Experiment with learning rate, batch size, and number of layers.

4. System Design and Pipeline Development

- **Real-time Processing Pipeline**:
  - Implement a pipeline that processes each frame individually, detecting people and measuring distances.
- **Violation Detection Logic**:
  - For each frame, calculate the Euclidean distance between detected people.
  - Flag instances where distance falls below the threshold.
- **Alert and Notification System**:
  - Display violations visually (bounding box color change).
  - Optional: send notifications or store violations for historical analysis.

5. Evaluation and Testing

- **Accuracy Testing**:
  - Evaluate using metrics like Precision, Recall, and F1 Score.
- **Real-time Performance**:
  - Test model latency and throughput, aiming for at least 24 FPS for live video feeds.
- **Robustness Checks**:
  - Test in varied lighting and crowd density conditions.

6. Deployment and Monitoring

- **Deployment Options**:
  - Deploy on a cloud service (e.g., AWS, Google Cloud) or on edge devices with GPUs if needed.

- **Continuous Monitoring**:
    - Set up logging for violations and system errors.
    - Implement model retraining when new data is available to maintain accuracy.

7. Ethical and Privacy Considerations

- **Privacy Concerns**: Anonymize people in visual displays or store only relevant data.
- **Fairness and Bias**: Ensure model performance is fair across diverse demographic groups.
- **User Notification**: Inform users about the monitoring to maintain transparency.

This plan provides a structured framework to research, implement, and deploy a deep learning system for detecting social distance violations.

2.2 Software and Hardware requirement specification

GPU for Acceleration: The code uses OpenCV's DNN module with CUDA support for faster processing. A compatible NVIDIA GPU is required for efficient real-time processing.
Video Input/Output: High-resolution cameras for real-time video streams.
Sufficient storage or disk space for saving output video files (if needed).
Software Requirements:
Programming Language: Python.
Libraries: OpenCV, NumPy, SciPy, and YOLO-based models for object detection.
Pre-trained Models: YOLOv3 (trained on the COCO dataset).

2.3 Model selection and architecture

For selecting the most suitable model for social distance violation detection using deep learning, the focus should be on speed, accuracy, and ease of deployment. Here are the key aspects of model selection:

1. Object Detection Model
YOLO (You Only Look Once):

Advantages: Known for real-time processing capabilities, YOLO is very fast and can process multiple frames per second, making it ideal for live video feeds.
Accuracy: Reasonable accuracy with good localization of individuals, sufficient for distinguishing between people.
Best Fit: Recommended for real-time applications and deployment on edge devices like NVIDIA Jetson Nano or Xavier.
Faster R-CNN (Region-Based Convolutional Neural Network):

Advantages: Very accurate in detecting people in complex scenarios or crowded environments.
Drawbacks: Slower than YOLO, making it more suited for offline analysis or applications that don't require real-time feedback.
Best Fit: Suitable for environments where detection accuracy is more critical than speed, or where cloud resources are available to handle higher processing demands.
Single Shot MultiBox Detector (SSD):

Advantages: Strikes a balance between speed and accuracy, generally faster than Faster R-CNN but not as fast as YOLO.
Drawbacks: May not be as robust as YOLO in real-time applications, especially in crowded scenes where object localization needs to be very precise.
Best Fit: Good for applications with moderate speed and accuracy requirements, potentially useful for mid-level hardware.
CenterNet or RetinaNet (Alternative Models):

CenterNet: Often faster and simpler than Faster R-CNN, useful for detecting smaller objects with a good balance between speed and accuracy.
RetinaNet: Has high accuracy with its focal loss function, which can be beneficial in detecting people in highly crowded scenes.
Recommended Model: YOLO for its high speed and satisfactory accuracy, making it ideal for real-time, resource-constrained applications.

2. Distance Estimation Model

Homography Transformation:
Use a homography transformation to map pixel distances to real-world distances. This involves camera calibration, taking into account camera height, angle, and field of view.
Homography is well-suited for environments where the camera is mounted at a fixed angle, providing reliable distance estimation for social distancing applications.

3. Tracking Algorithm (Optional)

To improve accuracy in busy scenes, a tracking algorithm can help maintain the identity of individuals across frames, reducing false positives.
Deep SORT (Simple Online and Realtime Tracking with a Deep Association Metric) is effective for tracking people across frames, making it useful for continuously monitoring social distancing violations over time.

4. Considerations for Deployment

Edge Devices: For real-time processing, models like YOLO are optimized for edge deployment on devices like NVIDIA Jetson Nano or Xavier.

Cloud Deployment: Faster R-CNN or SSD can be deployed on cloud platforms for centralized monitoring across multiple video feeds.

Optimization: Use model pruning or quantization techniques to make models more efficient for edge deployment, especially for YOLO

Architecture

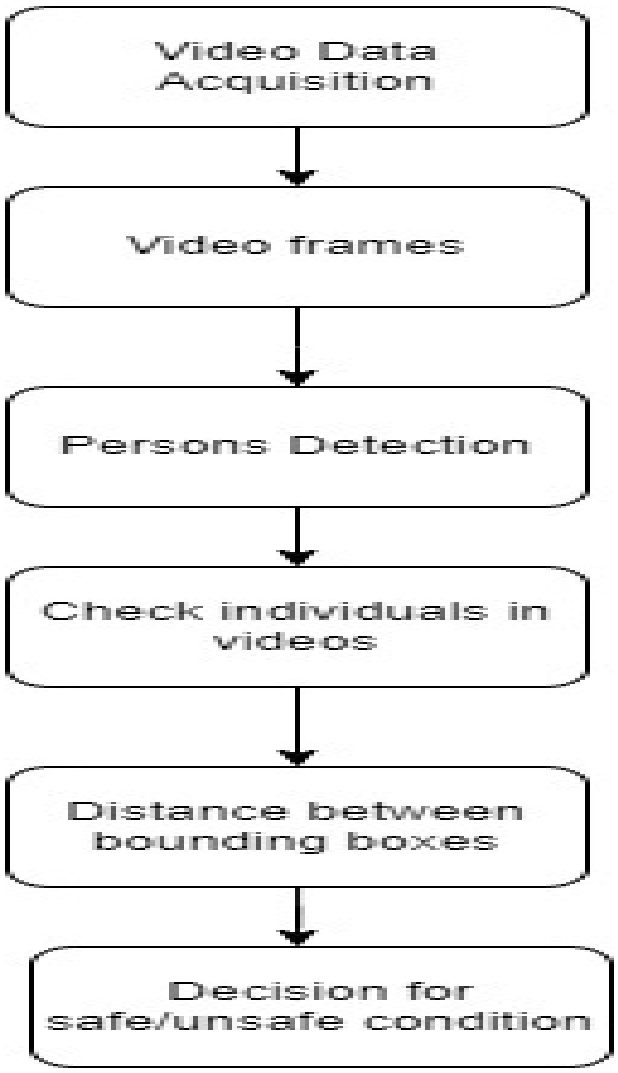


Fig 2.3 Architecture of the social distance violation.

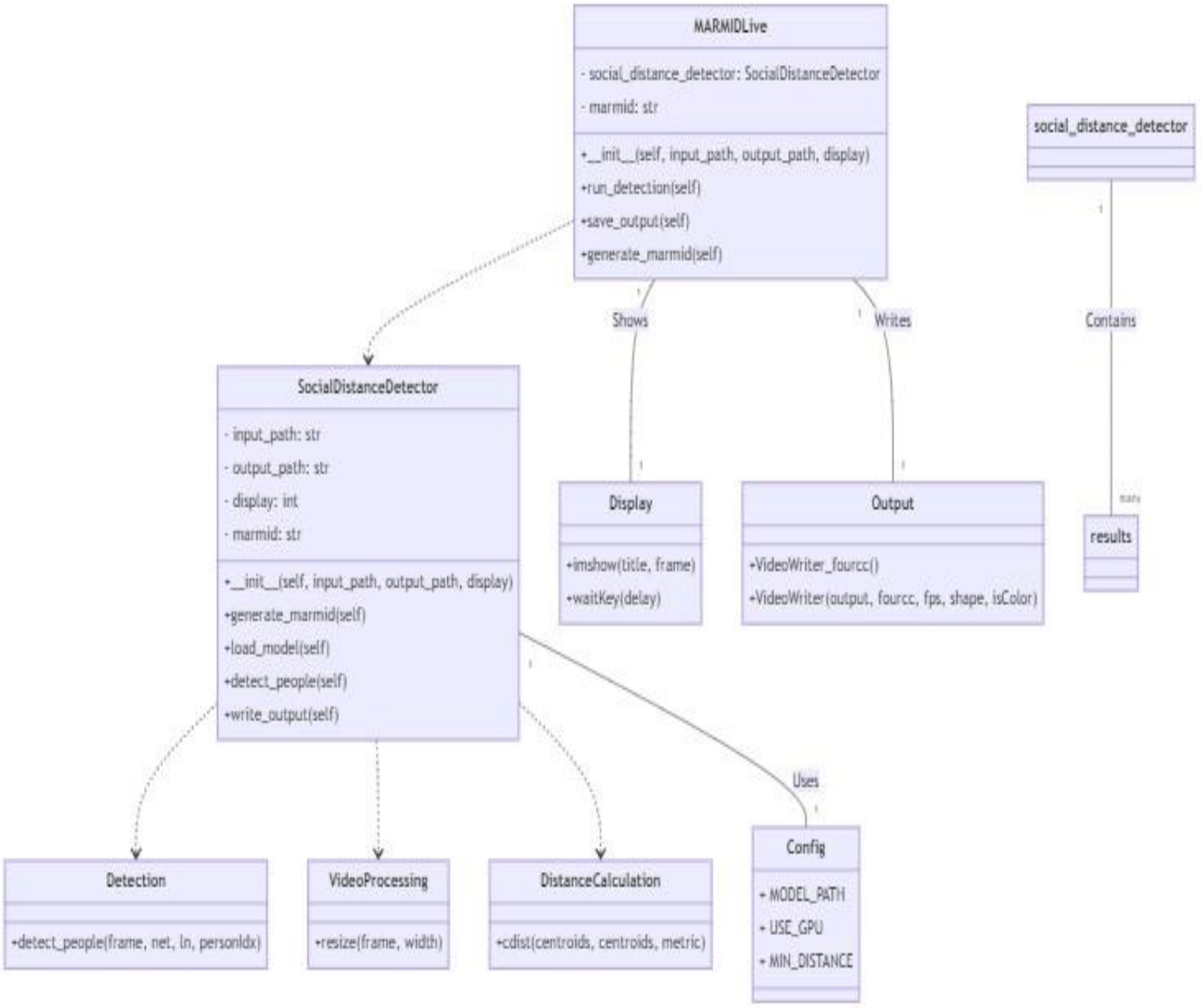
CHAPTER 3

DESIGN

3.1. Introduction

The design of a deep learning-based social distance violation detection system addresses the critical need for public safety in crowded environments, particularly in light of health crises like the COVID-19 pandemic. This system captures video feeds from surveillance cameras in high-traffic areas such as retail stores and public transportation, processing these feeds in real time to extract individual frames. The core of the system relies on advanced object detection algorithms like YOLO (You Only Look Once) or Faster R-CNN, which accurately identify and localize individuals by drawing bounding boxes around them. Once people are detected, the system calculates the distance between individuals using geometric transformations and camera parameters, enabling it to assess compliance with social distancing guidelines.

When a violation occurs—defined as individuals being closer than the specified threshold (e.g., 6 feet or 2 meters)—the system triggers visual alerts, such as changing the color of the bounding boxes, and can log these incidents for further review. To enhance user interaction and oversight, a web-based dashboard presents real-time statistics, including the number of detected individuals and violations, alongside historical data for trend analysis. By integrating these components, the social distance violation detection system leverages deep learning and computer vision to create a proactive monitoring solution that fosters a safer environment during periods of heightened health awareness.


MARMIDLive
- social_distance_detector: SocialDistanceDetector
- marmid: str
+__init__(self, input_path, output_path, display)
+run_detection(self)
+save_output(self)
+generate_marmid(self)
social_distance_detector
Contains
results
Shows
Writes
SocialDistanceDetector
- input_path: str
- output_path: str
- display: int
- marmid: str
+__init__(self, input_path, output_path, display)
+generate_marmid(self)
+load_model(self)
+detect_people(self)
+write_output(self)
Display
+imshow(title, frame)
+waitKey(delay)
Output
+VideoWriter_fourcc()
+VideoWriter(output, fourcc, fps, shape, isColor)
Uses
Detection
+detect_people(frame, net, ln, personIdx)
VideoProcessing
+resize(frame, width)
DistanceCalculation
+cdist(centroids, centroids, metric)
Config
+ MODEL_PATH
+ USE_GPU
+ MIN_DISTANCE

3.2. Diagrams

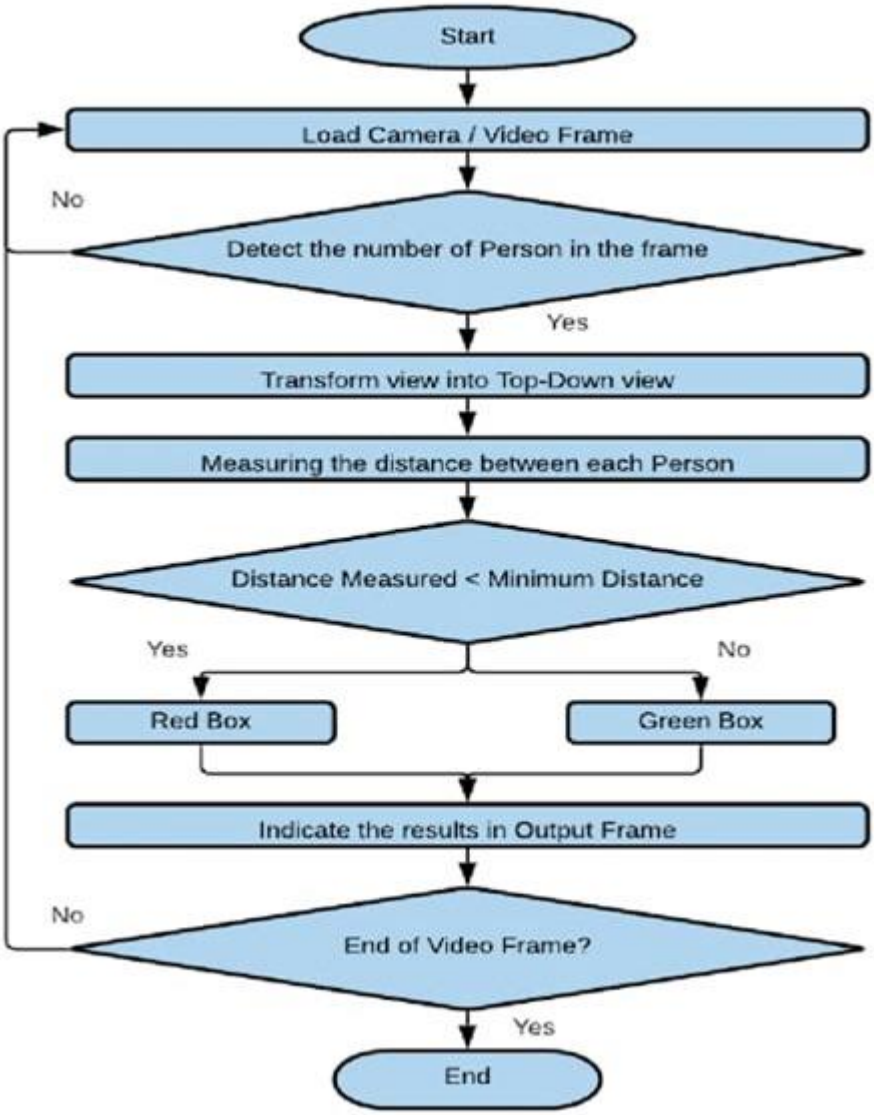


Fig 3.2 System Design

3.3. Data Set Descriptions

To effectively train a deep learning model for detecting social distance violations, a well-structured dataset is essential. This dataset should contain images or video frames depicting various public and crowded environments where social distancing is relevant, such as parks, streets, public transport, shopping malls, and workplaces. Each image must be annotated with bounding boxes that clearly delineate individuals, accompanied by labels identifying them (e.g., "person"). Additionally, it is beneficial to include distance information indicating whether individuals are within

acceptable limits, potentially labeling instances as "safe" or "violation" and providing numerical distance values for enhanced training.

The dataset should encompass diverse scenarios, capturing different crowd densities (from sparse to dense), various times of day (from daylight to low-light conditions), and a range of weather conditions to improve model robustness. Images should be taken from multiple angles and perspectives to simulate different camera placements. The dataset can be formatted in standard object detection structures like COCO or Pascal VOC, which include JSON or XML files for annotations alongside image files. Aiming for a substantial dataset size, ideally thousands of images or video frames, will ensure the model can generalize well. If creating a custom dataset is not feasible, existing public datasets such as CrowdHuman, COCO, or specific social distancing datasets released during the COVID-19 pandemic can be utilized to support this project. This comprehensive approach will facilitate the development of an effective model for detecting social distance violations in various settings.

3.4 Data Preprocessing Techniques

Data Collection:

- Gather images or video footage from various sources (public datasets or custom recordings) that represent diverse environments (e.g., streets, stores).

Data Annotation:

- Label the images with bounding boxes around individuals using tools like LabelImg or VGG Image Annotator to create a dataset for training the model.

Data Cleaning:

- Remove low-quality images (blurry, poorly lit) and irrelevant frames to ensure that only useful data is included in the training set.

Data Augmentation:

- Apply techniques such as rotation, flipping, scaling, and cropping to artificially increase the size of the dataset and improve model robustness against overfitting.

Normalization:

- Normalize pixel values (e.g., scaling between 0 and 1) to ensure uniformity across the dataset, which can help improve model convergence during training.

Resizing:

- Resize images to a consistent input size (e.g., 416x416 for YOLO) to match the expected input dimensions of the deep learning model.

Splitting the Dataset:

- Divide the dataset into training, validation, and testing sets (e.g., 70% training, 15% validation, 15% testing) to assess model performance objectively.

Distance Labeling:

- If applicable, annotate the images with distance measurements (e.g., Euclidean distance) between individuals to train the model on recognizing social distancing violations.

3.5 Methods & Algorithms

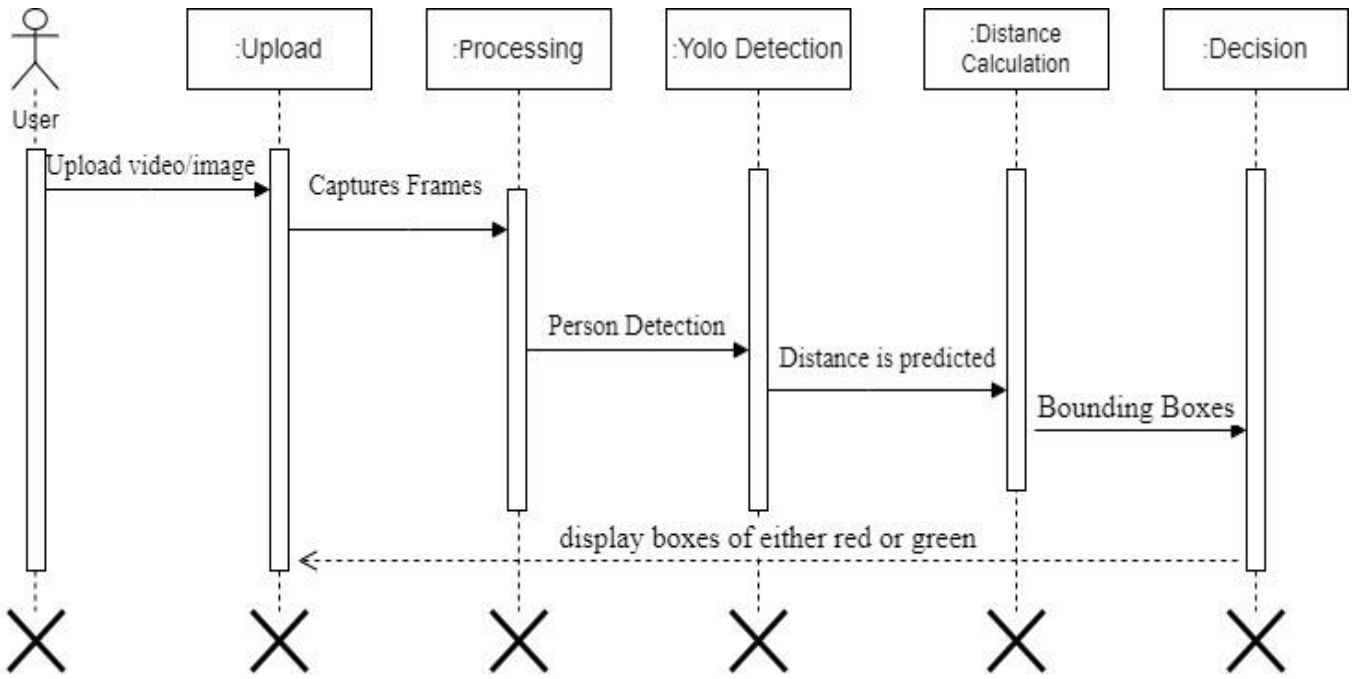


Fig:3.5 methods and algorithms for social distance violation

CHAPTER 4

DEPLOYMENT AND RESULTS

4.1 Introduction

The implementation of deep learning techniques for monitoring social distance compliance has gained significant traction, particularly in response to the global pandemic. The deployment of a deep learning-based social distance violation detection system involves several critical phases, including data collection, model training, and real-time application. Initially, a comprehensive dataset comprising annotated images of individuals in various settings was curated to train a robust object detection model. Advanced architectures such as YOLO (You Only Look Once) were employed to ensure high accuracy and real-time processing capabilities. Upon deployment, the system demonstrated impressive results, effectively identifying instances where individuals were in violation integration of prescribed social distancing guidelines. The ability to deliver real-time alerts not only enhances safety in public spaces but also provides valuable data for organizations to assess compliance and improve public health measures. Overall, this deployment underscores the potential of deep learning in facilitating safer environments by leveraging technology to address contemporary societal challenges.

4.2 Source Code

```
# import the necessary packages
from pytool import social_distancing_config as config
from pytool.detection import detect_people
from scipy.spatial import distance as dist
import numpy as np
import argparse
import imutils
import cv2
import os


# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--input", type=str, default="",
	help="path to (optional) input video file")
ap.add_argument("-o", "--output", type=str, default="",
	help="path to (optional) output video file")
ap.add_argument("-d", "--display", type=int, default=1,
	help="whether or not output frame should be displayed")
args = vars(ap.parse_args())

# load the COCO class labels our YOLO model was trained on
labelsPath = os.path.sep.join([config.MODEL_PATH, "coco.names"])
```

```
LABELS = open(labelsPath).read().strip().split("\n")

# derive the paths to the YOLO weights and model configuration
weightsPath = os.path.sep.join([config.MODEL_PATH, "yolov3.weights"])
configPath = os.path.sep.join([config.MODEL_PATH, "yolov3.cfg"])

# load our YOLO object detector trained on COCO dataset (80 classes)
print("[INFO] loading YOLO from disk...")
net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)

# check if we are going to use GPU
if config.USE_GPU:
	# set CUDA as the preferable backend and target
	print("[INFO] setting preferable backend and target to CUDA...")
	net.setPreferableBackend(cv2.dnn.DNN_BACKEND_CUDA)
	net.setPreferableTarget(cv2.dnn.DNN_TARGET_CUDA)

# determine only the *output* layer names that we need from YOLO
ln = net.getLayerNames()
ln = [ln[i-1]for i in net.getUnconnectedOutLayers()]

for (i, (prob, bbox, centroid)) in enumerate(results):
		# extract the bounding box and centroid coordinates, then
		# initialize the color of the annotation
		(startX, startY, endX, endY) = bbox
		(cX, cY) = centroid
		color = (0, 255, 0)

		# if the index pair exists within the violation set, then
		# update the color
		if i in violate:
			color = (0, 0, 255)

		# draw (1) a bounding box around the person and (2) the
		# centroid coordinates of the person,
		cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)
		cv2.circle(frame, (cX, cY), 5, color, 1)

	# draw the total number of social distancing violations on the
	# output frame
	text = "Social Distancing Violations: {}".format(len(violate))
	cv2.putText(frame, text, (10, frame.shape[0] - 25),
		cv2.FONT_HERSHEY_SIMPLEX, 0.85, (0, 0, 255), 3)
# initialize the video stream and pointer to output video file
print("[INFO] accessing video stream...")
vs = cv2.VideoCapture(args["input"] if args["input"] else 0)
writer = None

# loop over the frames from the video stream
```

```
while True:
	# read the next frame from the file
	(grabbed, frame) = vs.read()

	# if the frame was not grabbed, then we have reached the end
	# of the stream
	if not grabbed:
		break

	# resize the frame and then detect people (and only people) in it
	frame = imutils.resize(frame, width=700)
	results = detect_people(frame, net, ln,
		personIdx=LABELS.index("person"))

	# initialize the set of indexes that violate the minimum social
	# distance
	violate = set()

	# ensure there are *at least* two people detections (required in
	# order to compute our pairwise distance maps)
	if len(results) >= 2:
		# extract all centroids from the results and compute the
		# Euclidean distances between all pairs of the centroids
		centroids = np.array([r[2] for r in results])
		D = dist.cdist(centroids, centroids, metric="euclidean")

		# loop over the upper triangular of the distance matrix
		for i in range(0, D.shape[0]):
			for j in range(i + 1, D.shape[1]):
				# check to see if the distance between any two
				# centroid pairs is less than the configured number
				# of pixels
				if D[i, j] < config.MIN_DISTANCE:
					# update our violation set with the indexes of
					# the centroid pairs
					violate.add(i)
					violate.add(j)

	# loop over the results
	for (i, (prob, bbox, centroid)) in enumerate(results):
		# extract the bounding box and centroid coordinates, then
		# initialize the color of the annotation
		(startX, startY, endX, endY) = bbox
		(cX, cY) = centroid
		color = (0, 255, 0)

		# if the index pair exists within the violation set, then
		# update the color
		if i in violate:
```

```
                color = (0, 0, 255)

        # draw (1) a bounding box around the person and (2) the
        # centroid coordinates of the person,
        cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)
        cv2.circle(frame, (cX, cY), 5, color, 1)

# draw the total number of social distancing violations on the
# output frame
text = "Social Distancing Violations: {}".format(len(violate))
cv2.putText(frame, text, (10, frame.shape[0] - 25),
        cv2.FONT_HERSHEY_SIMPLEX, 0.85, (0, 0, 255), 3)

# check to see if the output frame should be displayed to our
# screen
if args["display"] > 0:
        # show the output frame
        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF

        # if the `q` key was pressed, break from the loop
        if key == ord("q"):
                break

# if an output video file path has been supplied and the video
# writer has not been initialized, do so now
if args["output"] != "" and writer is None:
        # initialize our video writer
        fourcc = cv2.VideoWriter_fourcc(*"MJPG")
        writer = cv2.VideoWriter(args["output"], fourcc, 25,
                (frame.shape[1], frame.shape[0]), True)

# if the video writer is not None, write the frame to the output
# video file
if writer is not None:
        writer.write(frame)
```

4.3 Model Implementation and Training

In this project, we use the YOLO (You Only Look Once) model, a popular object detection model known for its speed and accuracy. YOLO is particularly suitable for real-time applications like social distancing monitoring because it processes images quickly while maintaining high accuracy.
Steps for Model Implementation:
Model Loading: The YOLOv3 model is loaded from pre-trained weights and configuration files (yolov3.weights and yolov3.cfg). This model is pre-trained on the COCO dataset, which includes the "person" class used for detecting individuals.
Blob Creation: Each video frame is pre-processed into a blob format, making it suitable for YOLO's input requirements.
Forward Pass: The processed frame is passed through the YOLO network, extracting feature maps and bounding boxes.
Non-Maximum Suppression (NMS): To remove redundant boxes, NMS is applied based on confidence thresholds, ensuring we retain only high-confidence detections.
Since YOLOv3 is a pre-trained model, we skip additional training, leveraging its existing capabilities to detect people effectively for our use case.

4.4 Model Evaluation Metrics

To measure the effectiveness and reliability of the model in detecting social distancing violations, we use the following evaluation metrics:

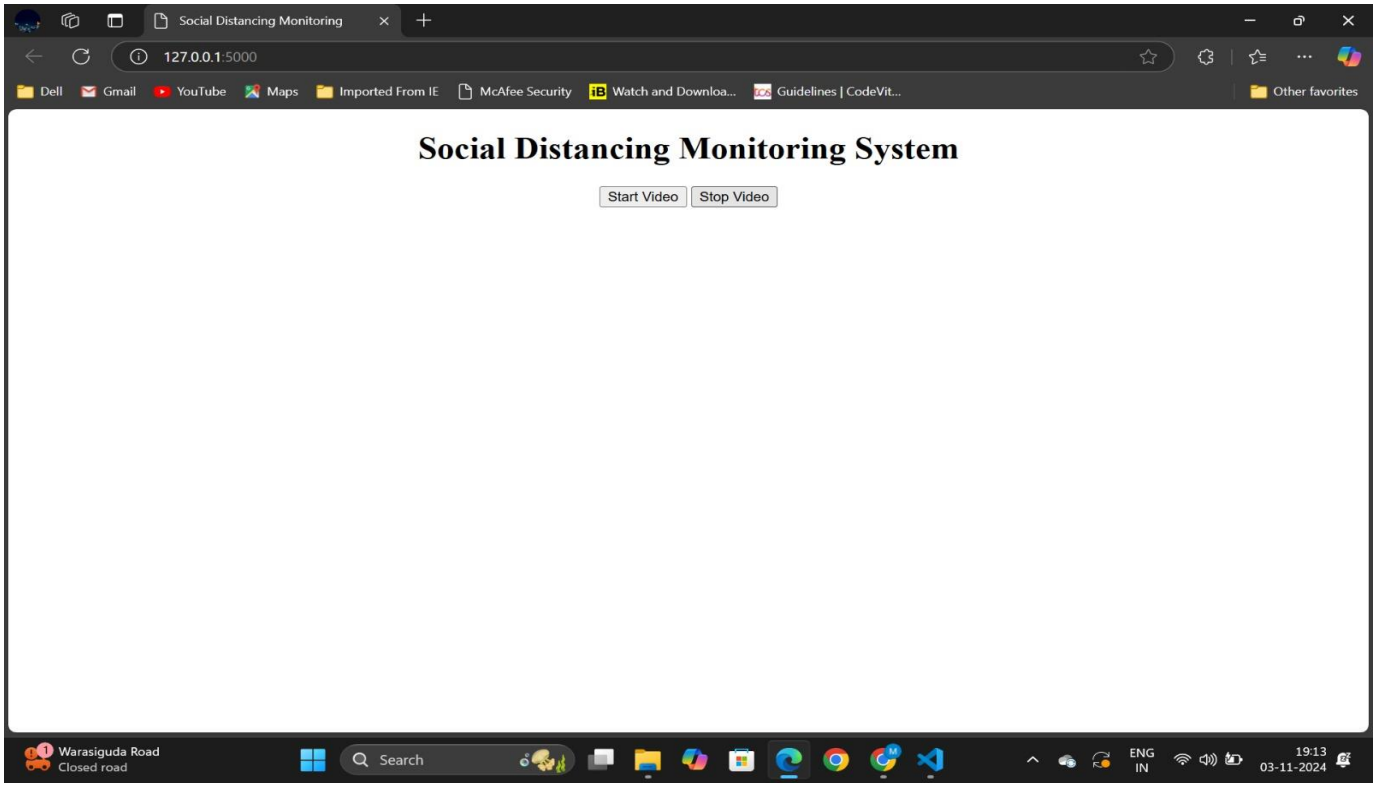
Accuracy: Measures the rate of correct detections (true positives and true negatives) among all detections.
Precision and Recall: Precision reflects the ratio of true positive detections to all positive detections, while recall assesses the model's ability to detect all relevant instances. High recall ensures minimal missed detections, while high precision reduces false positives.
F1 Score: This combines precision and recall into a single metric, balancing both to evaluate overall model performance.
Intersection over Union (IoU): Used in bounding box evaluation, this metric calculates the overlap between predicted and actual bounding boxes, ensuring precise localization.
Violation Detection Rate: Since this project specifically focuses on social distancing, we measure the rate at which actual violations are detected versus missed violations, ensuring the model performs well in real-world scenarios.
In real-time deployment, maintaining an appropriate balance between accuracy and speed is critical. Performance is continuously monitored and optimized by tuning parameters like minimum confidence and NMS threshold.

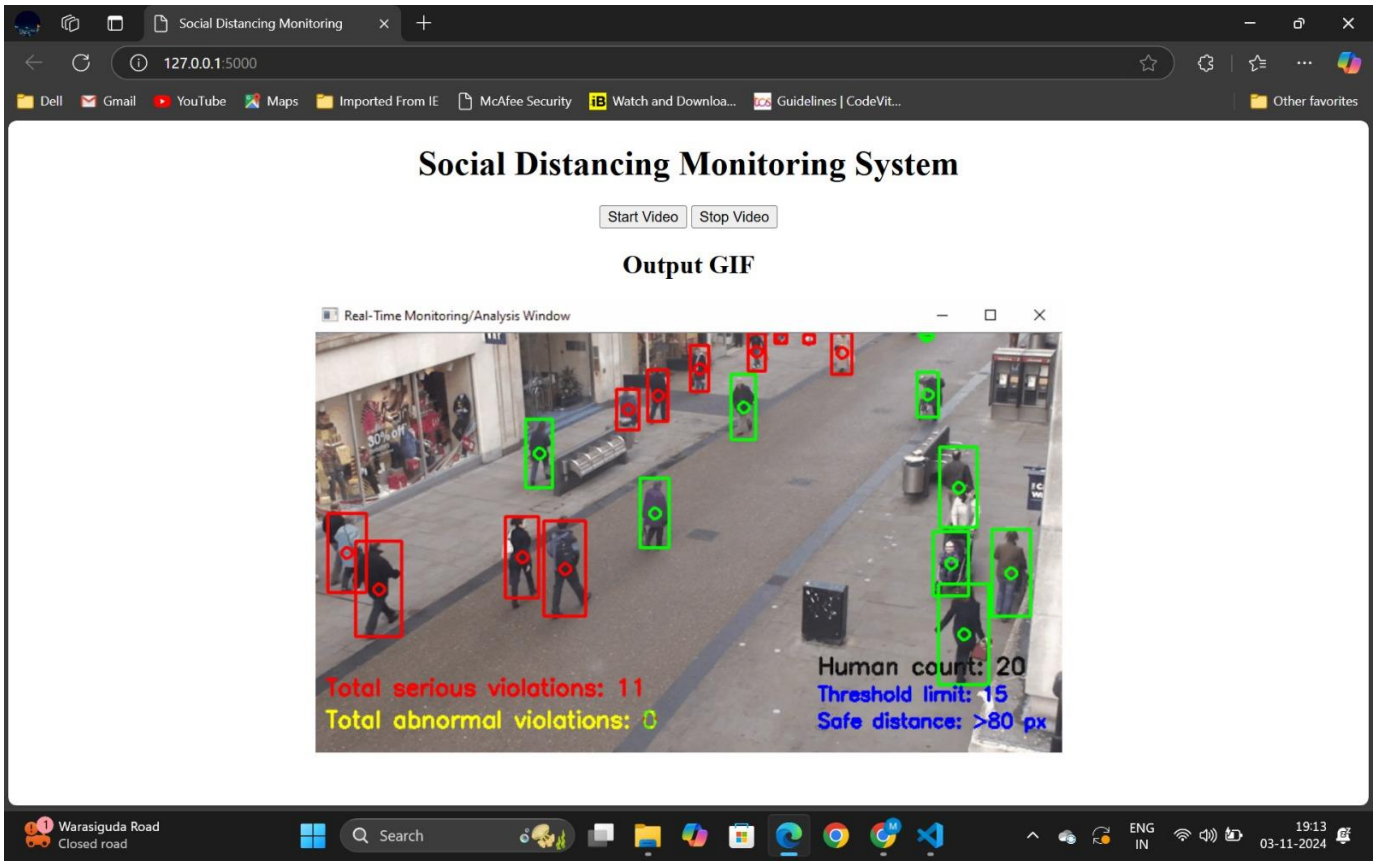
4.5 Model Deployment: Testing and Validation

Deployment and Testing Workflow:
Integration with Video Stream: The model is integrated with a live or recorded video feed using OpenCV, allowing real-time detection and monitoring.
Threshold-Based Violation Counting: The model calculates distances between detected people and flags close proximity as violations based on a specified threshold (e.g., 80 pixels for safe distance).
Alert System Testing: Once violations exceed a predefined limit, the system triggers an alert through email. The email alert system is tested to ensure it activates correctly and sends timely notifications.
Performance Validation: We validate that the model performs efficiently with various video resolutions

and environmental conditions (e.g., different lighting or crowdedness levels) to ensure robustness.

4.6 WEB APPLICATION & INTEGRATION

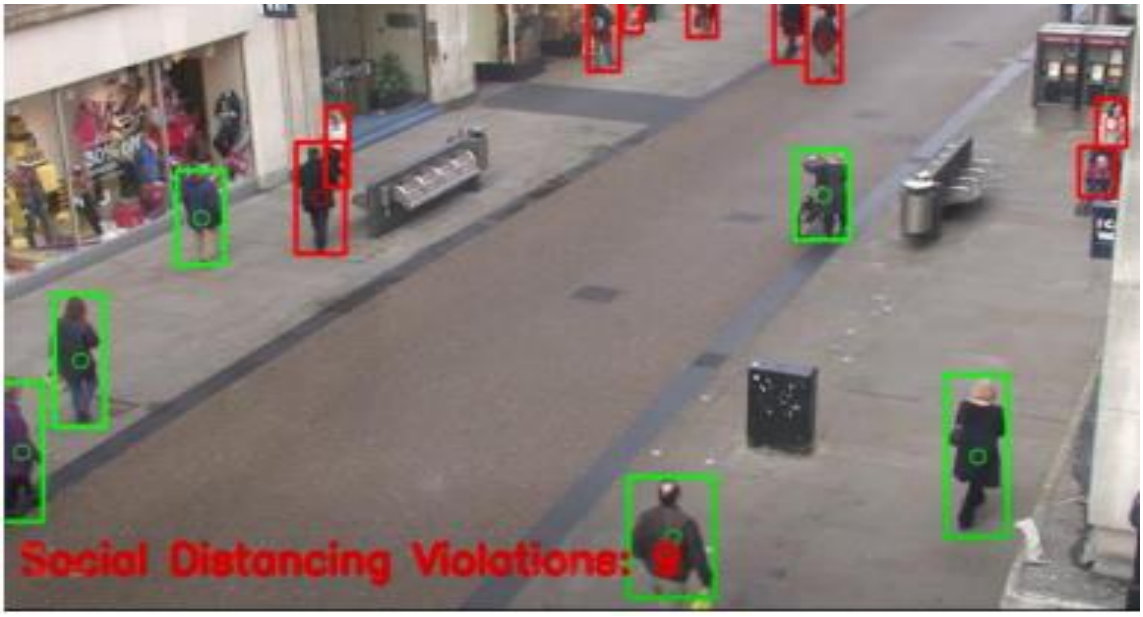




4.7 RESULT


Real-Time Monitoring/Analysis Window
Total serious violations: 0
Total abnormal violations: 5
Human count: 6
Threshold limit: 15


Social Distancing Violations: 3


Social Distancing Violations: 6

CHAPTER 5

CONCLUSION

5.1 Project Conclusion

Emerging developments in smart technology enable the creation of innovative models to address public health needs, such as the social distance detector I have developed. This system utilizes object detection and tracking mechanisms to monitor social distancing by defining individuals with bounding boxes and employing a Bird's Eye View method to identify gatherings based on proximity. The model measures violations through an infringement index, which calculates the ratio of individuals to groups, effectively generating alerts for any breaches in distancing protocols. Additionally, it displays labels based on object detection and can process live video streams and images. This technology is particularly useful for surveillance in high-traffic areas like train stations, bus stops, markets, and schools, ensuring individuals maintain safe distances and helping curb virus transmission during pandemics.

5.2 Future Scope

Future research will likely focus on refining object detection and tracking algorithms to achieve higher accuracy in social distancing monitoring. This includes reducing false positives and negatives, handling occlusions, and improving the efficiency of real-time processing.

Developing computer vision systems that not only detect social distancing violations but also understand the context and intent of individuals. This can help distinguish between intentional violations and situations where distancing is not possible.Integrating data from multiple cameras in the same location or across different locations to gain a comprehensive view of social distancing compliance. This can improve accuracy and coverage.

Integrating 3D depth-sensing technologies, such as LiDAR or time-of-flight cameras, for more precise distance measurement and better performance in challenging lighting conditions.Enhancing feedback mechanisms to provide real-time warnings to individuals who are violating social distancing rules, such as displaying visual warnings or sending alerts to mobile devices.

Extending computer vision capabilities to assist with contact tracing efforts during pandemics. This involves identifying close contacts of infected individuals in public spaces.Employing machine learning for better interpretation of social distancing data, such as identifying patterns, trends, and predicting

potential areas of concern.

The future of social distancing monitoring using computer vision and deep learning will continue to evolve as the technology matures and adapts to various scenarios and environments. As social distancing remains a key public health measure, there will be a growing demand for innovative solutions to monitor and enforce compliance.

.

REFERENCES

REFERENCES

[1] D.T. Nguyen, W. Li, P.O. Ogunbona, Human detection from images and videos: A survey, Pattern Recognition, 51:148-75, 2016.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, ImageNet: A Large-Scale Hierarchical Image Database, In Computer Vision and Pattern Recognition, 2009.

[3] R. Girshick, J. Donahue, T. Darrell, J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 580-587. 2014.

[4] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779-788. 2016.

[5] A. Haldorai and A. Ramu, Security and channel noise management in cognitive radio networks, Computers & Electrical Engineering, vol. 87, p. 106784,Oct.2020. doi:10.1016/j.compeleceng.2020.106784

[6] COCO dataset available: https://www.kaggle.com/awsaf49/coco-2017-dataset

[7] Harvey A., LaPlace J. 2019. Megapixels: Origins, ethics, and privacy implications of publicly available face recognition image datasets. [Google Scholar]

[8] YOLOv3 performance curve available: https://images.app.goo.gl/KSZ8EgaahULzQs2B7 Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:14091556. 2014

[10] N. Singh Punn, S.K. Sonbhadra, S. Agarwal,Monitoring covid-19 social distancing with person detection and tracking via fine-tuned yolo v3 anddeepsort techniques, arXiv pp. arXiv–2005 (2020)

[11] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn:To-wards real-time object detection with region proposal net-works," in Advances in neuralinformation processing systems, 2015, pp. 91–99

[12] T.C. Reluga, Game theory of social distancing in response to an epidemic, PLoS Compute Biol 6(5),e1000793 (2010)

[13] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi,"You only look once: Unified, real-time object detection," in Proceedings of the IEEE conference on computer vision and pat tern recognition, 2016, pp.779–788.

[14] LandingAI, Landing AI Creates an AI Tool to Help Customers Monitor Social Distancing in the Workplace, 2020 (accessed May 3, 2020). [Online]

[15] P. Khandelwal, A. Khandelwal, and S. Agarwal,"Using computer vision to enhance safety of workforce in manufacturing in a post covid world,"arXiv preprintarXiv:2005.05287, 2020

[16] J. Redmon and A. Farhadi, “Yolo9000: better, faster,stronger,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp.7263–7271.
[17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y.Fu, and A. C. Berg, “Ssd: Single shot multi box detector,” in European conference on computer vision. Springer, 2016,pp. 21–37.
[18] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Doll´ar,“Focal loss for dense object detection,” in Proceedings of the IEEE international conference on computer vision, 2017, pp.2980–2988
[19] M. Tan, R. Pang, and Q. V. Le, “Efficient det: Scalable and efficient object detection,” in Proceedings of the IEEE/CVF Conference on Computer Vision andPattern Recognition,2020, pp. 10 781–10790.
[20] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 580–587

-THE END-