**INTELLIGENCE VIDEO SURVEILLANCE SYSTEM**

**A project by**

**Sheibha S(RA2111047010217)**

**Praveen Kumar G M (RA2111047010191)**

**Nitheesh S(RA211047010194)**

**Thivakar R(RA2111047010194)**

***Under the Guidance of***

**R. Athilakshmi**

**Assistant Professor, Department of Computational Intelligence**

***In partial satisfaction of the requirements for the degree of***

## **BACHELORS OF TECHNOLOGY**

**In**

**ARTIFICIAL INTELLIGENCE**

## Logo, company name Description automatically generated

**SCHOOL OF COMPUTING**

# **COLLEGE OF ENGINEERING AND TECHNOLOGY**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**KATTANKULATHUR - 603203**

**August 2023**

Logo, company name

Description automatically generated

SRM INSTITUTION OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR-603203

**BONAFIDE CERTIFICATE**

Certified that this Course Project titled **“INTELLIGENCE VIDEO SURVEILLANCE SYSTEM”** is the bonafide work done by **Sheibha S (RA211047010217),Praveenkumar G M (RA2111047010191),Nitheesh S (RA2111047010194) and Thivakar R(RA2111047010197)** who carried out under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other work.

| **SIGNATURE**  Faculty In-Charge  **R. Athilakshmi**  Assistant Professor  Department of Computational Intelligence  SRM Institute of Science and Technology  Kattankulathur Campus, Chennai | **HEAD OF THE DEPARTMENT**  **Dr. R Annie Uthra**  Professor and Head,  Department of Computational Intelligence,  SRM Institute of Science and Technology  Kattankulathur Campus, Chennai |
| --- | --- |

TABLE OF CONTENTS

| SL NO | TITLE | PG NO |
| --- | --- | --- |
| 1 | INTRODUCTION |  |
| 2 | ABSTRACT |  |
| 3 | PROBLEM STATEMENT |  |
| 4 | REQUIREMENTS |  |
| 5 |  |  |

**INTRODUCTION**

The rapid advancement of technology has led to the increased need for effective and efficient surveillance systems. Image and video surveillance systems play a pivotal role in enhancing security, monitoring public spaces, and ensuring safety. These systems employ cutting-edge technologies to capture, process, and analyze visual data, providing real-time insights to operators and stakeholder.

In an era characterized by rapid technological advancements and the proliferation of data, the role of intelligence surveillance detection has emerged as a crucial component across various domains. This project, titled "Intelligence Video Surveillance System," delves into the intricate realm of surveillance technologies and their pivotal role in enhancing security, efficiency, and decision-making processes.

The contemporary landscape is witnessing an unprecedented surge in the utilization of intelligence surveillance detection systems across diverse sectors such as security, transportation, healthcare, and more. These systems leverage cutting-edge technologies like artificial intelligence, machine learning, computer vision, and data analytics to process and analyze copious amounts of data in real-time. By doing so, they provide invaluable insights, enabling stakeholders to proactively address challenges, mitigate risks, and optimize operational strategies.

The primary objective of this project is to explore the multifaceted nature of intelligence surveillance detection and its implications for Project 1. This comprehensive analysis will encompass both the theoretical underpinnings and practical implementations of surveillance technologies, shedding light on their capabilities, limitations, and ethical considerations. By delving into the intricacies of this field, we aim to facilitate a deeper understanding of how intelligence surveillance detection can be harnessed to achieve the goals and objectives of this Project.

**ABSTRACT**

The rapid advancements in computer vision, artificial intelligence, and data analytics have propelled the evolution of video surveillance systems beyond their conventional role as passive monitoring tools. This paper presents an innovative Intelligence Video Surveillance System (IVSS) designed and aimed at revolutionizing the way surveillance is conducted in various environments. The IVSS harnesses the power of deep learning, real-time analytics, and adaptive algorithms to enhance security, situational awareness, and decision-making processes.

The proposed IVSS integrates state-of-the-art object detection, tracking, and recognition models, enabling it to automatically identify and classify objects and individuals within the surveillance area. This real-time analysis ensures timely threat detection and alert generation, allowing security personnel to respond promptly to potential security breaches. Furthermore, the system employs anomaly detection techniques to identify unusual patterns and behaviors, enabling the anticipation of abnormal incidents.

IVSS is not limited to security applications; it extends its utility to operational optimization. By analyzing crowd density, flow patterns, and occupancy rates, the system assists in crowd management and resource allocation. Additionally, the IVSS incorporates predictive analytics, learning from historical data to anticipate potential security vulnerabilities and optimizing resource deployment accordingly.

To address privacy concerns, the IVSS incorporates advanced anonymization techniques that blur or obscure personally identifiable information, ensuring compliance with data protection regulations. Moreover, the system provides customizable zones and permission-based access control to restrict monitoring in sensitive areas, striking a balance between security and privacy.

The implementation of IVSS is expected to yield significant benefits, including reduced response times to security incidents, improved resource allocation efficiency, and enhanced overall situational awareness. This system's adaptability and scalability make it applicable across diverse scenarios, from transportation hubs and critical infrastructure to commercial establishments and public spaces.

In conclusion, the Intelligence Video Surveillance System presented in this paper demonstrates a novel approach to modern surveillance challenges. By leveraging cutting-edge technologies, real-time analytics, and adaptive algorithms, IVSS enhances security, optimizes operations, and respects privacy considerations. The development and deployment of such an intelligent surveillance system mark a critical step forward in ensuring safer and more secure environments.

Keywords: intelligence video surveillance, deep learning, real-time analytics, object detection, anomaly detection, predictive analytics, privacy, security, situational awareness, adaptive algorithms.

**PROBLEM STATEMENT**

In today's rapidly evolving world, security concerns have become paramount, necessitating the development of advanced surveillance systems. Traditional video surveillance systems often fall short in efficiently and effectively detecting and responding to security breaches, incidents, or anomalies. There is a growing need for an Intelligent Video Surveillance System that leverages cutting-edge technologies to enhance security monitoring, automate incident detection, and improve overall response times.

This project aims to address these challenges by designing, developing, and implementing an Intelligent Video Surveillance System that goes beyond basic video recording and monitoring. The system will utilize advanced artificial intelligence and computer vision techniques to analyze video streams in real-time, identifying potential threats, unauthorized activities, or abnormal behaviors, and triggering appropriate responses.

Key Objectives:

**Real-time Monitoring:** The system should provide live video monitoring capabilities, allowing security personnel to access and observe multiple camera feeds simultaneously.

**Automated Threat Detection:** The core functionality of the system will involve the automatic detection of threats or suspicious activities, such as intrusion, theft, vandalism, and loitering.

**Anomaly Recognition:** The system should be capable of identifying abnormal behaviors or events, such as abandoned objects, erratic movements, or overcrowding.

**Alert Generation:** Upon detecting potential threats or anomalies, the system should generate real-time alerts to notify security personnel or relevant authorities.

**Integration with Existing Infrastructure:** The Intelligent Video Surveillance System should be designed to seamlessly integrate with existing security systems, including CCTV cameras, access control systems, and alarms.

**Scalability:** The system should be scalable, allowing the addition of new cameras and components without significant architectural changes.

**User-friendly Interface:** Develop an intuitive user interface that provides a comprehensive view of the surveillance network, alerts, and relevant data, enabling security personnel to make informed decisions quickly.

**Data Storage and Retrieval:** Implement a secure and efficient data storage mechanism that allows recorded video footage to be stored, indexed, and easily retrievable for investigative purposes.

**Performance and Reliability:** Ensure the system operates reliably, minimizing false positives while maintaining a high level of accuracy in threat detection.

**Privacy Considerations:** Implement privacy-conscious measures, such as video anonymization and access controls, to uphold ethical and legal standards regarding surveillance.

**Deliverables:**

The expected outcomes of this project include a functional Intelligent Video Surveillance System with a well-designed user interface, capable of real-time monitoring, automated threat detection, and alert generation. A detailed technical documentation describing the system architecture, algorithms used, data flow, and integration methods should also be provided.

**Significance:**

The Intelligent Video Surveillance System developed in this project has the potential to significantly enhance security monitoring across various domains, including public spaces, commercial establishments, industrial facilities, and residential complexes. By harnessing the power of AI and computer vision, the system can help prevent security breaches, reduce response times, and provide a safer environment for individuals and assets.

**REQUIREMENTS**

### Softwares Required

* Python: Language in which code is written
* CMake: For compiling openCV
* Visual Studio Code: For building openCV and darknet code
* Nvidia GPU Driver: For faster GPU performance
* CUDA: For parallel computing using GPU
* CuDNN: A GPU-accelerated library of primitives for deep neural networks
* OpenCV: For working on images/videos in python
* Darknet: Neural network framework for this project

### **Usage:**

There are four ways to perform detection on videos:

1. Video from Webcam
2. Local Stored Video
3. YouTube video
4. Video from Mobile Camera ( [DroidCam](https://www.dev47apps.com/) )

## **Social Distancing Detection**

### **Working**

We take the input video from a source and divide the video into several frames. Now these frames are converted into black and white. On each frame a person is detected using YOLO.

Now we write the code to draw rectangles on the detected persons. We check the distances between each detected person on the frame from each other. If the distance between the two persons is less than a particular value then we color the box red and draw a line between these boxes and add the no. of social distancing violations in a variable and display it.

All of the above processes happen for a single frame. Now all of this is set in a loop for each frame of the video and People at Risks are detected.

**CODING:**

#================================================================

from ctypes import \*

import math

import random

import os

import cv2

import numpy as np

import time

import darknet

from itertools import combinations

import pafy

import youtube\_dl

def is\_close(p1, p2):

"""

#================================================================

# Purpose : Calculate Euclidean Distance between two points

#================================================================

:param:

p1, p2 = two points for calculating Euclidean Distance

:return:

dst = Euclidean Distance between two 2d points

"""

dst = math.sqrt(p1\*\*2 + p2\*\*2)

#=================================================================#

return dst

def convertBack(x, y, w, h):

#================================================================

# Purpose : Converts center coordinates to rectangle coordinates

#================================================================

"""

:param:

x, y = midpoint of bbox

w, h = width, height of the bbox

:return:

xmin, ymin, xmax, ymax

"""

xmin = int(round(x - (w / 2)))

xmax = int(round(x + (w / 2)))

ymin = int(round(y - (h / 2)))

ymax = int(round(y + (h / 2)))

return xmin, ymin, xmax, ymax

def cvDrawBoxes(detections, img):

"""

:param:

detections = total detections in one frame

img = image from detect\_image method of darknet

:return:

img with bbox

"""

#================================================================

# Purpose : Filter out Persons class from detections and get

# bounding box centroid for each person detection.

#================================================================

if len(detections) > 0: # At least 1 detection in the image and check detection presence in a frame

centroid\_dict = dict() # Function creates a dictionary and calls it centroid\_dict

objectId = 0 # We inialize a variable called ObjectId and set it to 0

for detection in detections: # In this if statement, we filter all the detections for persons only

# Check for the only person name tag

name\_tag = str(detection[0].decode()) # Coco file has string of all the names

if name\_tag == 'person':

x, y, w, h = detection[2][0],\

detection[2][1],\

detection[2][2],\

detection[2][3] # Store the center points of the detections

xmin, ymin, xmax, ymax = convertBack(float(x), float(y), float(w), float(h)) # Convert from center coordinates to rectangular coordinates, We use floats to ensure the precision of the BBox

# Append center point of bbox for persons detected.

centroid\_dict[objectId] = (int(x), int(y), xmin, ymin, xmax, ymax) # Create dictionary of tuple with 'objectId' as the index center points and bbox

objectId += 1 #Increment the index for each detection

#=================================================================#

#=================================================================

# Purpose : Determine which person bbox are close to each other

#=================================================================

red\_zone\_list = [] # List containing which Object id is in under threshold distance condition.

red\_line\_list = []

for (id1, p1), (id2, p2) in combinations(centroid\_dict.items(), 2): # Get all the combinations of close detections, #List of multiple items - id1 1, points 2, 1,3

dx, dy = p1[0] - p2[0], p1[1] - p2[1] # Check the difference between centroid x: 0, y :1

distance = is\_close(dx, dy) # Calculates the Euclidean distance

if distance < 75.0: # Set our social distance threshold - If they meet this condition then..

if id1 not in red\_zone\_list:

red\_zone\_list.append(id1) # Add Id to a list

red\_line\_list.append(p1[0:2]) # Add points to the list

if id2 not in red\_zone\_list:

red\_zone\_list.append(id2) # Same for the second id

red\_line\_list.append(p2[0:2])

for idx, box in centroid\_dict.items(): # dict (1(key):red(value), 2 blue) idx - key box - value

if idx in red\_zone\_list: # if id is in red zone list

cv2.rectangle(img, (box[2], box[3]), (box[4], box[5]), (255, 0, 0), 2) # Create Red bounding boxes #starting point, ending point size of 2

else:

cv2.rectangle(img, (box[2], box[3]), (box[4], box[5]), (0, 255, 0), 2) # Create Green bounding boxes

#=================================================================#

#=================================================================

# Purpose : Display Risk Analytics and Show Risk Indicators

#=================================================================

text = "People at Risk: %s" % str(len(red\_zone\_list)) # Count People at Risk

location = (10,25) # Set the location of the displayed text

cv2.putText(img, text, location, cv2.FONT\_HERSHEY\_SIMPLEX, 1, (246,86,86), 2, cv2.LINE\_AA) # Display Text

for check in range(0, len(red\_line\_list)-1): # Draw line between nearby bboxes iterate through redlist items

start\_point = red\_line\_list[check]

end\_point = red\_line\_list[check+1]

check\_line\_x = abs(end\_point[0] - start\_point[0]) # Calculate the line coordinates for x

check\_line\_y = abs(end\_point[1] - start\_point[1]) # Calculate the line coordinates for y

if (check\_line\_x < 75) and (check\_line\_y < 25): # If both are We check that the lines are below our threshold distance.

cv2.line(img, start\_point, end\_point, (255, 0, 0), 2) # Only above the threshold lines are displayed.

#=================================================================#

return img

netMain = None

metaMain = None

altNames = None

def YOLO():

"""

Perform Object detection

"""

global metaMain, netMain, altNames

configPath = "./cfg/yolov4-tiny.cfg" # Path to cfg

weightPath = "./yolov4-tiny.weights" # Path to weights

metaPath = "./cfg/coco.data" # Path to meta data

if not os.path.exists(configPath): # Checks whether file exists otherwise return ValueError

raise ValueError("Invalid config path `" +

os.path.abspath(configPath)+"`")

if not os.path.exists(weightPath):

raise ValueError("Invalid weight path `" +

os.path.abspath(weightPath)+"`")

if not os.path.exists(metaPath):

raise ValueError("Invalid data file path `" +

os.path.abspath(metaPath)+"`")

if netMain is None: # Checks the metaMain, NetMain and altNames. Loads it in script

netMain = darknet.load\_net\_custom(configPath.encode(

"ascii"), weightPath.encode("ascii"), 0, 1) # batch size = 1

if metaMain is None:

metaMain = darknet.load\_meta(metaPath.encode("ascii"))

if altNames is None:

try:

with open(metaPath) as metaFH:

metaContents = metaFH.read()

import re

match = re.search("names \*= \*(.\*)$", metaContents,

re.IGNORECASE | re.MULTILINE)

if match:

result = match.group(1)

else:

result = None

try:

if os.path.exists(result):

with open(result) as namesFH:

namesList = namesFH.read().strip().split("\n")

altNames = [x.strip() for x in namesList]

except TypeError:

pass

except Exception:

pass

#cap = cv2.VideoCapture(0) # Uncomment to use Webcam

#cap = cv2.VideoCapture("Video\_for\_Testing.mp4") # Uncomment for Local Stored video detection - Set input video

#url = "https://www.youtube.com/watch?v=isveXCH4NcM" # Uncomment these lines for video from youtube

#video = pafy.new(url)

#best = video.getbest(preftype="mp4")

#cap = cv2.VideoCapture()

#cap.open(best.url)

#cap = cv2.VideoCapture('http://192.168.0.106:4747/mjpegfeed') # Uncomment for Video from Mobile Camera (DroidCam Hosted Camera)

frame\_width = int(cap.get(3)) # Returns the width and height of capture video

frame\_height = int(cap.get(4))

new\_height, new\_width = frame\_height // 2, frame\_width // 2

#print("Video Reolution: ",(width, height))

#out = cv2.VideoWriter("output.avi", cv2.VideoWriter\_fourcc(\*"MJPG"), 10.0, # Uncomment to save the output video # Set the Output path for video writer

#(new\_width, new\_height))

# print("Starting the YOLO loop...")

# Create an image we reuse for each detect

darknet\_image = darknet.make\_image(new\_width, new\_height, 3) # Create image according darknet for compatibility of network

while True: # Load the input frame and write output frame.

prev\_time = time.time()

ret, frame\_read = cap.read()

# Check if frame present :: 'ret' returns True if frame present, otherwise break the loop.

if not ret:

break

frame\_rgb = cv2.cvtColor(frame\_read, cv2.COLOR\_BGR2RGB) # Convert frame into RGB from BGR and resize accordingly

frame\_resized = cv2.resize(frame\_rgb,

(new\_width, new\_height),

interpolation=cv2.INTER\_LINEAR)

darknet.copy\_image\_from\_bytes(darknet\_image,frame\_resized.tobytes()) # Copy that frame bytes to darknet\_image

detections = darknet.detect\_image(netMain, metaMain, darknet\_image, thresh=0.25) # Detection occurs at this line and return detections, for customize we can change

image = cvDrawBoxes(detections, frame\_resized) # Call the function cvDrawBoxes() for colored bounding box per class

image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

print(1/(time.time()-prev\_time)) # Prints frames per second

cv2.imshow('Demo', image) # Display Image window

cv2.waitKey(3)

#out.write(image) # Write that frame into output video

cap.release() # For releasing cap and out.

#out.release() # Uncomment to save the output video

print(":::Video Write Completed")

if \_\_name\_\_ == "\_\_main\_\_":

YOLO()

**Screenshots:**