Enhanced Real-Time Crowd Counting UsingYOLOv8 and DeepSORT Optimization.

Prof. Abhijeet More
Dept. Of Computer Application Pillai HOC College of Engineering and Technology (Mumbai University) Rasayani Maharshtra,India
abhijeetmore@mes.ac.in

1st Sanika Keni Dept. Masters of Computer Application Pillai HOC College of Engineering and Technology (Autonomous) Rasyani, Maharashtra
sanikakeni875@gmail.com

2nd Sumedh Jadhav Dept. Masters of Computer Application Pillai HOC College of Engineering and Technology (Autonomous) Rasyani, Maharashtra
sumedhjadhav08232@gmailcom

3rd Aditya Thorve Dept. Masters of Computer Application Pillai HOC College of Engineering and Technology (Autonomous) Rasyani, Maharashtra
adityathorve100@gmail.com

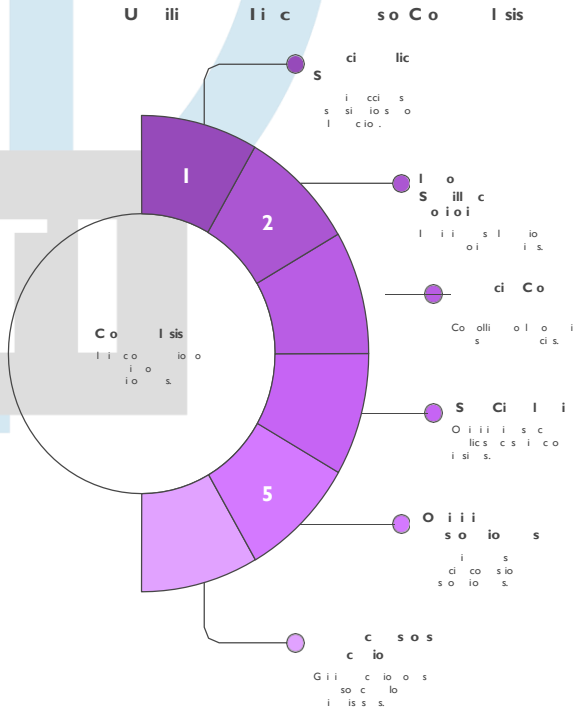
***Abstract*— Real-time detection of crowds in urban and public spaces has become increasingly important for public safety, security, and traffic movement due to the large number of people congregating at transport hubs, shopping malls, stadiums, and public gatherings. This paper proposes a Crowd Detection and Density Estimation System based on state-of-the-art deep learning and computer vision techniques that can detect, analyze and quantify the density of crowds using live CCTV camera footage. The proposed approach uses a YOLO based object detection algorithm for real-time detection of persons in the scene, due to its high accuracy and low latency performance. In combination with the proposed frame extraction and preprocessing techniques, the proposed approach is highly effective in recognizing people in real-time in highly dynamic, cluttered and complex environments. The system can identify people, calculate crowd density levels, and classify them into low, moderate, and high density levels. It then provides automated alerts, event logs, and a live monitoring dashboard to help authorities to take timely preventive actions. The combination of smart automation with surveillance infrastructure reduces the need for humans, thus saving time and improving the quality of decisions.**

Keywords*— *Crowd Detection, Computer Vision, YOLO, Surveillance System, Real-time Monitoring

1. INTRODUCTION

Crowd Monitoring various public places like airports, metros, malls, and event venues for crowd levels has become a priority in order to ensure safety and prevent overcrowding[2]. As urban areas become more and more densely populated, it is not only inefficient but also unreliable to rely on human land command alone.

Besides, the execution of manual monitoring may bring delays and inconsistencies, especially in large facilities. With the help of computer vision and deep learning, automated crowd detection systems are a faster, more robust solution that can reduce human error and increase situation awareness. This project presents a real, time Crowd Detection System that is based on the YOLO deep



learning model to recognize people from live camera footage and figure out the crowd density in general [5]. The system remains unswerving under different light sources and camera positions, and it also produces higher accuracy results at low latency. Furthermore, it has a web viewing interface that displays the live feed, total person counts, and the density level, thus, giving the security teams a clear sight with which they can make quick decisions.

This project combines the use of automatic detection and smart processing methods, which may lead to better public safety services, preservation of the operational efficiency of intelligent surveillance systems, and optimization of the crowd management decision, making processes.

2. BACKGROUND

People are hard to track when they are moving around in the world. This is because people move in ways and it is tough to predict what they will do. Old computer systems used to try to figure out where people were by looking at the edges of things and the background. They would also try to see how things were moving from one frame to the next.. These systems had a lot of trouble when there were a lot of people in one place. This is because people would get in the way of each other and it would be hard to tell who was who. The lights would also be different. It would be hard to see. So these systems were not very good at keeping track of people in time. Then artificial intelligence and deep learning came along. Changed everything. These new systems could look at pictures. Learn from them without being told what to do. They could figure out what was important and what was not. New systems like Faster R-CNN and SSD were better at finding people in places. But they needed a lot of power to work so they were not very good at working in time.

Then the YOLO system came along. It was a big deal. It could find people. Figure out what they were doing all at the same time. It was also very fast. Could work on regular computers. The YOLO system got better and better with versions like YOLOv5, YOLOv7, YOLOv8 and YOLO11. These new versions could find people accurately and work faster. They could also work on types of devices. The YOLO system is very good at finding people and counting crowds. Just finding people is not enough. We also need to be able to track them as they move around. This is where Multi-Object Tracking algorithms come in. These algorithms can keep track of people as they move around and make sure we know who is who. They do this by looking at how people move and what they look like. Systems that use both detection and tracking are more reliable in crowded places.

Recently people have been working on making systems that can handle a lot of people and work in time. They want to be able to analyze how people are moving around and use this information to make cities smarter. This is what our project is about. We are using the YOLO system to detect people and count crowds. We are also working on making the system more efficient. This will help us make cities better and safer. The YOLO system is a part of this project. We are building on the YOLO system to make it even better. The YOLO system is very important, for crowd counting and density levels.

3. LITERATURE REVIEW

Y. Gai and their team did some work in 2021 [1] where they came up with a system to track people. They used YOLOv5 along with DeepSORT. Of using old methods they detected people one frame at a time using YOLOv5. Once they found the people they used Kalman filtering to guess where they would move next. Kalman filtering is a way to make guesses from messy data. Then they used the algorithm to match their guesses to the actual positions of people. This helped them keep track of who was who. The YOLOv5 and DeepSORT setup worked better than using YOLOv3 with DeepSORT. It had ID changes too. Although it made detection more reliable and movements smoother it can still have trouble when there are conditions, like heavy clutter or when people are fully covered. Also when things get tough the processing load goes up so it needs hardware to work well.

M Abdou and Erradi wrote a paper called "Crowd Counting: A Survey of Machine Learning Approaches" in 2020 [2] They looked at how machines can count people using methods, both old and new. The authors did not just list tools instead they examined approaches like making predictions step by step estimating crowd density and using deep learning networks designed for counting people. One section of the paper improves a known network used for monitoring people at bus stops. Although the paper identifies weaknesses in existing research particularly when crowds are very dense it lacks evidence from large-scale tests, across data sets.

P. Sivaprakash and team tried something new in 2023 [3]. They used learning to count crowds. P. Sivaprakash and team did not do what others do. They made a system that uses neural networks to figure out how many people are in a picture. This system looks at every part of the picture and makes a detailed map of where people are. It does not just look at the edges or shapes of people. It looks at every pixel. Says how many people are there. To count people you just need to add up all these numbers. When there are a lot of people in a space it is hard to see what is going on. P. Sivaprakash and team found that when people are close together it is hard to get a picture of what is happening. To make this system work you need a lot of pictures to teach it. You also need a lot of power to make it work fast. This makes it not very good, for systems that need to work like live systems.

In the paper "Vehicle Detection and Multi-Target Tracking System in Intelligent Transportation using YOLOv8 and Adaptive Deep Sorting Algorithm" by Y. Yu and others that came out in 2025 [4] they talked about a system that uses YOLOv8 to detect things and Adaptive DeepSORT to track them. This system is special because it updates the Kalman

filter in a way and it can change how important appearance features are on the fly. When there are not cars, on the road this Vehicle Detection and Multi-Target Tracking System works really well. It can find cars very accurately and track them nicely. When the road gets busy the Vehicle Detection and Multi-Target Tracking System does not work as well because it gets confused and has to slow down the video to keep up.

N. Wojke and his team made some changes in 2017.[5] They found a smart way to track objects online. Now it works a lot better. They added a way to link how things look. This means the computer does not have to guess much. It uses something called Kalman filters to figure out where things are going. Before anything starts the computer gets ready by learning some codes. These codes help the computer match faces from one picture to the next. It can do this even if the faces are not in the same spot. This is really cool because it means the computer can keep track of people even when they move around. For example the number of times the computer got the user ID wrong went down by half. It was 45 percent less. The computer was still fast though. There is still a problem. The computer needs to be really good, at finding objects in the place. If there are a lot of things packed tightly together the computer can get confused & Make mistakes.

I read about this research called "An Online DeepSORT-Based Vehicle Tracking Method for Supporting Realtime Vehicle Surveillance" by Y. Sun and Sun from 2023. [6] They tried out methods like SORT. Deepsort to track vehicles in real time. They used YOLOv8s to detect vehicles and a special metric to associate them which they trained on the VeRi dataset. They did not just see how it worked step by step. They also checked how well it worked in situations. They used something called MOTA metrics to see the results. The method worked fine when there was traffic.. It had problems when vehicles overlapped or when the boxes, around them were not detected correctly. This happened a lot in areas where the vehicle IDs changed suddenly. The vehicle tracking method had trouble with this.

A. Abadi and team introduced a method in 2023 titled “Detection of Cyclist’s Crossing Intention Based on Posture Estimation for Autonomous Driving” [7]. Instead of relying on motion tracking alone, they combined pose prediction with direction sensing through artificial intelligence patterns. When a rider appears, software notices the presence first by mapping key joints via thermal imaging-style outputs. Following that step, guesses about which way the body is facing and turning come into play during path forecasting. Decisions here depend heavily on how angles shift across frames rather than raw speed data. While safety models often ignore individual gestures, this approach ties behavior prediction closely to subtle changes in upper limb alignment. Even though the method boosts safety in self-driving cars and works well, it uses heavy computations and struggles when objects hide or lighting changes.

A paper by W. Lu (2024) [8], titled “Safety Helmet Detection System Based on YOLOv5 Algorithm,” introduces a live helmet detection system powered by YOLOv5s to boost safety at construction sites. Instead of relying on human review, the approach uses object detection to spot whether workers wear helmets in video feeds. While it cuts down on time spent watching screens, accuracy sometimes drops when scenes get cluttered or lighting is low. Though useful, results depend heavily on environmental factors during capture.

Starting off, J. Wang and team introduced an updated version of YOLO - called Mobile-YOLO - in their 2022 paper [9]. Instead of sticking to traditional designs, they reshaped the YOLOv4 framework by cutting down on unnecessary data points. This change helps speed up processing without losing much precision in finding objects. Though the core idea stays familiar, it now runs leaner through reworked layers that keep performance steady. Built from 3,698 diverse images showing objects at different sizes, the tailored drone image set supported hands-on testing. At just over 51 frames per second, object spotting happened quickly enough to work smoothly in live operations. Speed here does not come at the cost of precision, which helps when resources are tight and space is limited. Still, adjusting settings for each dataset is needed. When scenes get messy - changes in size, bad weather - performance can dip.

Starting with raw sensor data, B. Jin in 2024 [10] introduced an approach for spotting and following objects using unprocessed visuals instead of refined image streams. Instead of first altering light signals through complex processing paths, the system leans on YOLOv8 to identify targets within scenes captured straight from cameras. Following detections, BOT-SORT takes over, linking together multiple instances across motion-filled sequences without extra delays. Because early stage cleanup steps are skipped entirely, workload drops sharply when handling large volumes of input frames. Results tested against RobotCar benchmarks showed performance matching established metrics - accuracy levels holding steady even as speed and resource needs dipped noticeably. Results show raw images can work well for live autonomous tasks. Still, how well they do hinges on sensor accuracy and setup precision. More testing needs to happen under different scenes and machine setups before relying on it steadily.

4. PROBLEM STATEMENT

Even with fast changes in the crowd's position, spotting how full an area is now can be tricky when people block views or directions shift mid-frame. Views from security cams tend to come at odd angles, making estimates harder than they first seem. Light fluctuations across scenes add another layer of uncertainty during actual operations. Most setups today still need people to watch screens all the time. This way of doing things does not work well when it matters because people get slower at making decisions.

What this work does is create a tool that can automatically find people in moving images and figure out how crowded an area is.. It does this quickly and accurately. The goal of this tool is to build a system like YOLO that can look at video clips fast like what happens at events and find people correctly. Then it sends out warnings based on how many people're, in a crowd to help communities stay safe and teams make good decisions.

5. METHODOLOGY

The way we do things is about creating a crowd detection system. This system is used to find people and count crowds and see how they behave. We use computer vision and deep learning techniques to make it work.Our crowd detection system has a steps. We need to be able to identify people and figure out how dense the crowd is.. We need to do it all in real time. We get video from cameras or streaming devices. Then we make each frame of the video better.We make it smaller. Get rid of noise and make it normal so our crowd detection system can handle it. The improved frames are then sent to the part of the crowd detection system that detects people in the crowd. Our crowd detection system uses these frames to detect people, in the crowd and count the crowd.

After we detect people we do some steps like filtering out detections that are not sure and counting how many people are there. Based on this count we say if the crowd is small, medium or big. We show the results and how dense the crowd is on a website that also shows the video, the count and any warning messages. We use Python and OpenCV to make sure everything happens fast and works well. OpenCV helps us show the frames draw boxes around people, video and send out the stream all with very little delay. Because OpenCV is not too big it helps the different parts of our system talk to each other smoothly even when the crowd is very dense or the light is not good. We always get a steady stream of frames. Crowd detection system is what we use to detect crowds. It is very important, for crowd detection.

5.1. Object Detection: YOLOv8

The first step is object detection. This is where the system finds and classifies every person in a video frame. This step is important because it sets the quality and speed for the rest of the process.

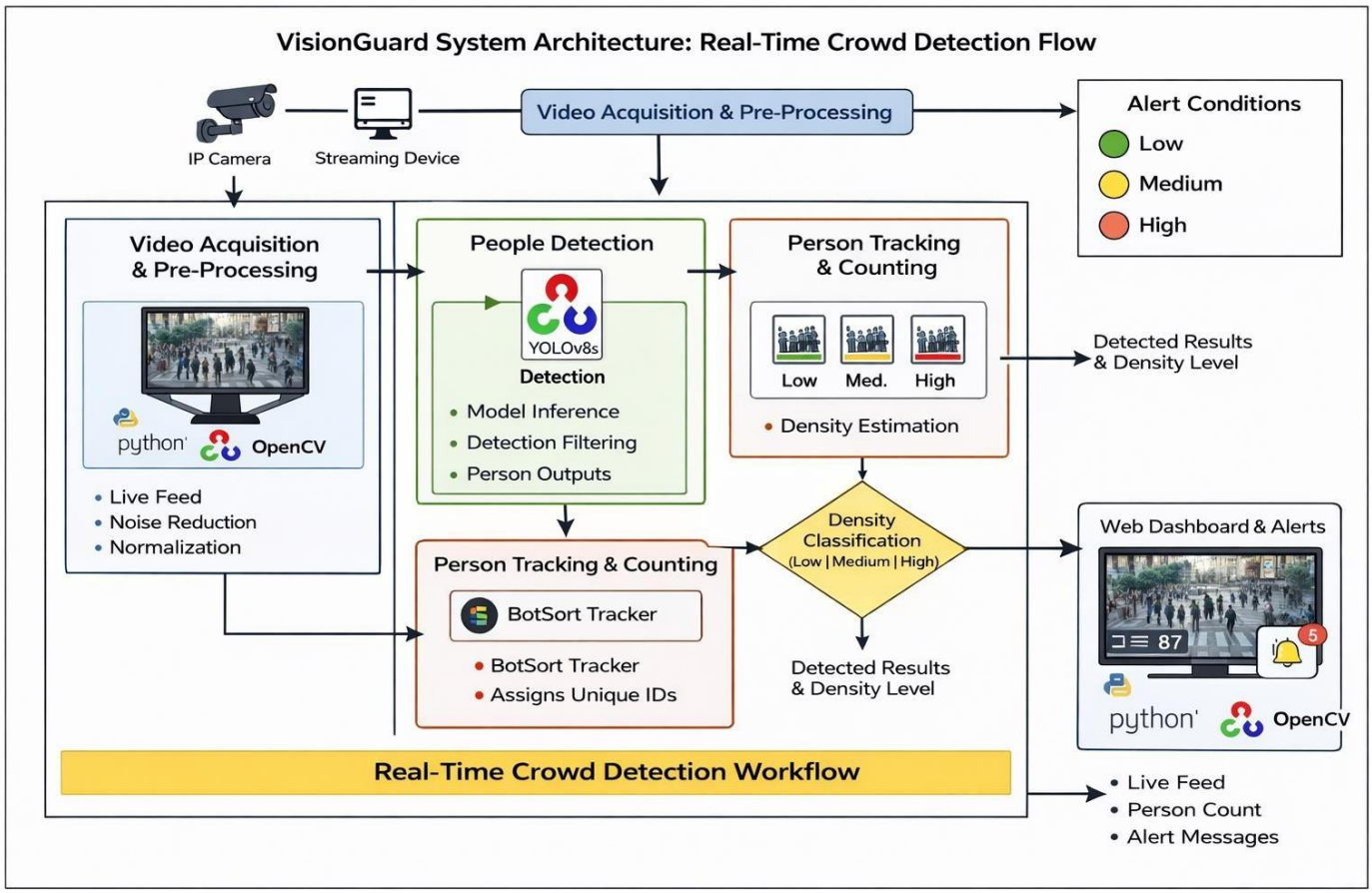
5.1.1. Model Selection and Implementation

The system uses YOLOv8. This is a state-of-the-art model that's very good at finding objects. The system uses the version of YOLOv8. This was a choice because it is fast and accurate. YOLOv8 is better than versions because it uses an anchor-free architecture. This makes the system simpler and better at finding objects. The model can predict where objects are and what they are. This makes the system faster and better for real-time applications.

1. Input and Prediction: The system looks at each video frame. It then uses the -trained neural network to make predictions. The model gives a lot of predictions, including where objects might be and what they might be.

2. Output Filtering and Refinement: After the model makes predictions the system filters them. It only keeps the predictions that're about people. The system uses the COCO dataset standard to know what is a person. Then it uses Non-Maximum Suppression. This means it picks the prediction with the confidence and removes the other predictions that are similar. This ensures that the output is a list of boxes that show where each person is in the frame. The system uses YOLOv8 to do this. YOLOv8 is good, at finding people in video frames. The system uses YOLOv8 to make sure the output is accurate.

3. Result: The result is a data set that is now ready to be used by the tracking module in the next stage.



5.1. Person Tracking and Counting: BotSort

We need to keep track of people in videos. We use a tool called YOLOv8 to find people. Then we use another tool to give each person a name so we can tell them apart. This way we do not count the person twice. We want to know how many people are in the video and this tool helps us do that.

5.2.1. BotSort Tracking Algorithm

The VisionGuard platform uses a tool called BotSort to track people. BotSort is a tool that can follow people even when they are in a crowd. It uses ways of tracking and new ways with deep learning. This means it can recognize people even when they come back into the picture after being gone for a bit. BotSort is good, at tracking people in areas and it helps us get the right count of people. trajectories in complex, crowded environments.

1. Motion Prediction is used in crowded places.

It uses the Kalman Filter to do its job. The Kalman Filter is a tool that helps figure out where things are and how big they are in a picture. It looks at how things move. Then it tries to guess where they will be next. The Kalman Filter thinks that it can understand how things move by using some math ideas. This helps Motion Prediction make a guess about where things will be and how big they will be. Motion Prediction with the Kalman Filter is really useful. It can cover up things that block our view for a time by giving us a good idea of where they will show up again. The way it does this is by using a measurement called the Mahalanobis distance. The Mahalanobis distance measures how apart the predicted location of something is from where it is actually seen. This is helpful for Motion Prediction and the Kalman Filter. The Kalman Filter and Motion Prediction work together to make things clearer. The Kalman Filter is a part of Motion Prediction. Motion Prediction uses the Kalman Filter to make guesses, about where things are and how big they are.

1. Re-Identification (Re-ID) (OSNet-x0_25): For appearance-based matching, BotSort integrates a pre-trained deep convolutional neural network, the lightweight OSNet-x0_25_msmt17 [C9]. This network takes the pixels inside a persons box. Turns them into a special kind of

feature that can tell people apart. The network uses a version of itself and a special set of pictures called MSMT17 to make sure it works well and can handle things like different camera angles, lighting and when people are partly hidden..

2. Data Association is like the brain of the system. It makes a table that shows how similar each person it has seen before is to each new person it finds. This table looks at how people move and what they look like. By using both of these things the system can keep track of people even when they are hidden from view or when they look the same in pictures. This is important, for counting people over a long time.

5.2.1 person counting Methodology:

The project wants to find out how many different people are there. It does this by using an good way of storing data in the monitor_worker.py script.

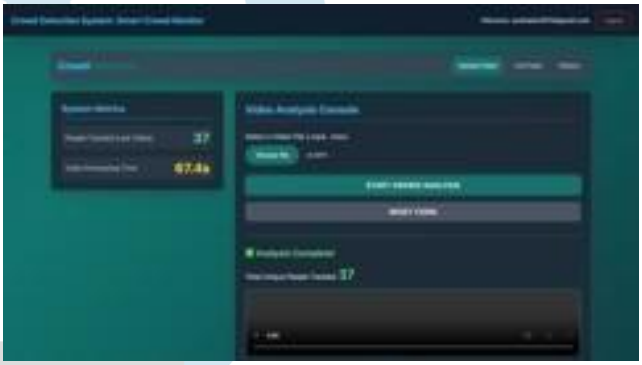
5.1 *Fig. Crowd Counting Uploaded Feed*

1. Initialization: The project starts by making a list of ids called global_unique_ids.

2. Track ID Addition: When the project looks at each frame of the video it gets a list of track ids from the BotSort tracker. It adds each of these ids to the global_unique_ids list.

3. Final Count Reliability: The list stores each id one time. So if the BotSort tracker gives the same id to the person many times it is only counted one time. At the end of the video the project counts how many ids are in the list. This gives the number of unique people found in the video. The project uses this method to make sure it gets the count of unique people that the tracker found in the whole video. The unique person counting methodology is used to count the people. The unique person counting methodology works by using the global_unique_ids list. The unique person counting methodology is important, for getting the count of unique people.

5.2 *Fig. Crowd Counting Dashboard*



5.3 *Fig. Crowd Counting Upload Video*

5.4 *Fig. Crowd Counting Video History*

5.5 *Fig. Crowd Counting Time and People Count*

6. SYSTEM IMPLEMENTATION

TThe Crowd Detection System is an application that works in two modes. It can handle jobs that need to be very accurate and it can also watch things in real time without slowing down. The Crowd Detection System is made using Python. This makes it good, at keeping things separate and handling jobs in a way. The Crowd Detection System is designed to do these things well.

Step	Component	Description
1	File Ingestion (server.py – /upload_video)	Receives the uploaded video file and stores it securely in the tmp/uploads directory for further processing.
2	Task Creation (server.py)	Generates a unique UUID (task_id), initializes task status as PENDING, and stores metadata in a global thread-safe tasks dictionary.
3	Subprocess Launch (server.py)	Uses Python's subprocess module to execute monitor_worker.py in the background. Input path, output path, and log path are passed as command-line arguments.
4	Core Processing (monitor_worker.py)	Initializes YOLOv8s and BotSort (with OSNet Re-ID). Processes video frames sequentially, performs detection, assigns unique tracking IDs, and draws bounding boxes on frames.
5	Counting & Output Generation (monitor_worker.py)	Maintains a global set of unique IDs to compute total unique count. Saves processed video to static/output and writes final count to a JSON log file in tmp/tasks.
6	Status Retrieval (server.py – /status/<task_id>)	Allows frontend to check task progress. Access to the tasks dictionary is protected using threading.Lock() for thread safety.
7	Result Delivery (server.py)	Reads final count from JSON log file, calculates execution duration, updates status to COMPLETED, and sends results to frontend dashboard.
8	Cleanup (server.py)	Deletes temporary JSON log files after successful result retrieval to maintain storage efficiency.

6.1 Technical Stack and Development Environment The system uses a modern libraries and technologies.

1. We mainly use Python 3.x for our project.

2. For building the web application we use Flask. Flask handles all the routing serves the frontend and also manages our API endpoints.

3. The frontend is built with HTML and JavaScript and styled with Tailwind CSS to make it look modern and work well on different devices.

4. For computer vision tasks we rely on OpenCV, which is used for capturing frames and handling video input/output.

5. Deep learning is done using Ultralytics YOLO, which's great for detecting objects.

6. To track objects and handle Re-ID we use BoxMOT, which implements BotSort.

7. For tasks, in parallel we use Pythons built-in threading and subprocess modules.

6.2 Offline Video Analysis flow

This detailed workflow is handled by server.py. Run by monitor_worker.py. It is designed to count unique people in videos.

6.2 Live Feed Monitoring (Real-Time Stream)

The live feed is taken care of by the Flask server at the time so that the live feed is seen right away when it is needed for real-time applications. The live feed is handled by the Flask server in a way that reduces the time it takes to show the feed. This is important, for the feed because it has to be seen in real-time. The Flask server is used to handle the feed so that people can see what is happening in real-time.

6. *Fig. Croud Counting Live Feed*

1. When we start the stream in the server.py file it begins by setting up a new instance of the YOLOv8 model and a special tracker called BotSort that we use just for the live stream. This tracker stays active for long as the stream is running.

2. The generate frames function is what gets the video going. It uses OpenCV to capture frames from the webcam.

3. As the video plays we show how many people are in the frame now. Because the stream never stops we do not keep a count of all the unique people who have been in the frame. Instead the live stream focuses on showing us how many people are in the frame, at this moment using the YOLOv8 model and the BotSort tracker to make it happen.

3.2. Model Configuration and Tuning

The Crowd Detection System works well because of the choice and setup of the deep learning models used for detection and tracking. This is not about using the default settings. It is, about making the models work best for the parts of crowd monitoring. The Crowd Detection System has to deal with a lot of objects in a space things getting in the way of each other and the need to get answers quickly in real time. The Crowd Detection System needs all these things to work together to be effective.

3.2.1. Detection and Tracking Initialization

The initial setup of the detection and tracking parts is very important for it to work well. For detecting things we chose the YOLOv8s.pt model weight because it is a mix of being fast and accurate which is needed for processing lots of data quickly in real time. This detection model works with a Confidence Threshold of 0.25, which's the default setting for Ultralytics models. This setting helps the model detect people properly even when a few people are close together. For tracking we chose BotSort with BoxMOT tracker type. BotSort uses both the movement and appearance of objects to keep track of identities. This is possible because of a Re-Identification Weight, the osnet_x0_25_msmt17.pt model. The osnet_x0_25_msmt17.pt model is light and good at telling people because it is trained on a big dataset, for this purpose.

Finally the internal Re-ID Threshold can be. It decides how similar features should be to confirm that what is being detected is already being tracked, thus stopping the identity from being switched.

3.2.2. Frontend Interaction and Status Polling

The user experience depends on how the frontend and the server tasks that run in the background talk to each other.

1. Task Polling: When a user uploads a video the JavaScript code in the visionguard_dashboard.html file starts checking the server every seconds. It does this by sending HTTP GET requests to the /status/<task_id> route to see what is going on with the task.

2. State Management: The server has a list of all the tasks and their status like if they're still waiting to start, currently running or already finished. This list is like a database that the server uses to keep track of things. The server also has a lock that prevents problems when many users are checking on their tasks at the time.

3. Dynamic Rendering: When the task is finally done the frontend gets the results like how many people were in the video and how long it took to process. Then it updates the system metrics panel with the numbers, like the total count of people and how long the video took to process. It also changes the video player to show the video so the user can see the result right away without having to reload the page. The video just shows up. The user can watch it. This makes the user experience better because the video is available to watch. The user does not have to wait for the page to reload or do anything. The final processed video is there ready to watch.

Fig. Tracking and counting using DeepSort(scenario 1)

Diagram is needed

6.1 *Fig. Tracking and counting using DeepSort (scenario 2)*

6.2 *Fig. Tracking and counting using DeepSort (scenario 3)*

6.3 *Fig. Tracking and counting using DeepSort (scenario 4)*

7. RESULTS AND DISCUSSION

The Crowd Detection System is really good at what it does. I looked at how the Crowd Detection System works by comparing it to systems. We took a look at four Multi-Object Tracking algorithms. BotSort, ByteTrack, DeepSORT and StrongSORT. The Crowd Detection System was used with the YOLOv8 detection model and these algorithms We wanted to see how well the Crowd Detection System and these algorithms could count people correctly. This is important because it helps us understand if the Crowd Detection System and these algorithms can keep track of people properly and tell them apart. The Crowd Detection System and these Multi-Object Tracking algorithms are used for things like counting people in a crowd. The Crowd Detection System is good, for this because it can keep track of people.

7.1. Comparative Performance Metrics

The performance of each algorithm was compared to the data from three video samples. This was done to see how well each algorithm could count things compared to the count.

TEST DATA	Metric	Bot Sort	Byte Track	DeepSort
Sample 1 (High Density)	Original Count	34	34	34
	Calculated Count	28	58	31
	Accuracy (%)	82.0	58.6	91.2
	Dcount	6	24	3
Sample 2 (Variable Occlusion)	Original Count	32	32	32
	Calculated Count	23	69	29
	Accuracy (%)	71.8	46.3	90.6
	Dcount	9	37	3
Sample 3 (Low Density)	Original Count	11	11	11
	Calculated Count	13	20	16
	Accuracy (%)	84.6	55.0	68.7
	Dcount	3	9	5

7.2. Discussion of Findings

The results of the experiment show that some algorithms are better than others at tracking things. They also show how well these algorithms can handle problems that happen in crowded areas:

StrongSORT is the best: StrongSORT was always the most accurate. It even got a score in one of the video samples and was very close to perfect in another. This shows that its combination of features and ways of associating data is the best way to keep track of things in crowded areas.

DeepSORT is strong: DeepSORT did a job of tracking things even when there were a lot of people in the area. This shows that using a filter to predict movement and combining it with special features is a good way to track things.

BotSort is okay: BotSort did a job of tracking things but it tended to undercount. This means that when people are moving quickly or are blocked from view it might lose track of them and not be able to find them

ByteTrack has problems: ByteTrack made a lot of mistakes when counting things. It would often count many people. This is because it would connect detections that were not very confident which would create tracks. This makes it not suitable, for counting people in crowded areas.

Tracker	Average Accurasy (%)	Visual Representation
StrongSORT	98.5%	
DeepSort	90.9%	
BotSort	76.9%	
ByteTrack	52.4%	

7.3. Formal Definition of Evaluation Metrics

1. Difference Count (Dcount):

The difference between the number of people the algorithm thinks it sees and the real number of people is called Dcount. Dcount is a measure of how wrong the algorithm's when it counts people. It shows if the algorithm is undercounting or overcounting people.

A small Dcount value means the algorithm is very good at counting people and we can trust it. This is especially important when there are a lot of people in a space or when people are blocking each other from view.

Dcount is a way to compare how well different algorithms work in situations. Because Dcount measures how far, off the algorithm is, we can use it to see which algorithms are the consistent and stable. We can use Dcount to compare algorithms when people are moving around or when the environment is changing. Dcount helps us understand how well the algorithm is working in these situations. This is really important when we are talking about Dcount and how it works with different algorithms. We can look at Dcount to see how well the algorithm is working and this is where Dcount is really useful it helps us understand Dcount and the algorithm.

2. Accuracy:

A The accuracy of an algorithm is the number of things it gets right compared to the number of things it says it finds. This gives us a percentage that shows how well the algorithm really works. If the algorithm misses some things that're really there that is a problem. The algorithm can also get confused it thinks one thing is things that is a problem too. The accuracy metric takes into account both of these problems with the algorithm. So the accuracy metric gives us a picture of how the algorithm

$$\text{Accuracy}(\%) = \left(1 - \frac{\text{Dcount}}{\text{Original Count}}\right) \times 100$$

can keep track of things from one frame to the next. This is especially important when we are watching things in time because even small mistakes with the algorithm can add up over time and cause problems, with the algorithm.

7.4. Discussion: Speed vs. Accuracy Trade-off

1. StrongSORT's Advantage: Its superior performance comes at a higher computational cost due to more complex association and filtering stages.

2. BotSort's BotSort's Advantage (Deployment Focus): BotSort is optimized for faster inference speed. For a web-based application, a minor compromise on peak accuracy is often accepted to ensure a higher rate. This choice is critical for two reasons:

1. Asynchronous Batch Processing: It ensures the monitor_worker.py completes long video tasks quickly, maintaining rapid turnaround times.

2. Live Feed Responsiveness: It guarantees the server.py can maintain a low-latency, high-throughput stream necessary for a smooth real-time user experience.

7.1 *Fig. Sample Tracking Scenario 1*

7.2 *Fig. Sample Tracking Scenario 2*

7.3 *Fig. Sample Tracking Scenario 3*

7.4 *Fig. Sample Tracking Scenario 4*

8. CONCLUSION

This project made a Crowd Detection System that works in two ways. The Crowd Detection System uses YOLOv8 to find objects and a special tracking system to count people. This Crowd Detection System is ready to use. Has a simple design. The Crowd Detection System has a backend that uses Flask and can handle video processing tasks without slowing down the frontend of the Crowd Detection System. We tried four Multi-Object Tracking algorithms to see which one is the best. StrongSORT was very accurate when we tested it.. When we thought about using it in the real world we chose BotSort because BotSort is fast and accurate. This makes BotSort good for watching things all the time and for processing batches of data. The Crowd Detection System is what we are talking about here. The Crowd Detection System uses models like YOLOv8s.pt and OSNet-x0_25. This makes the Crowd Detection System work better and be more reliable. The Crowd Detection System is a solution for monitoring crowds. The Crowd Detection System helps keep people safe and makes cities smarter. The Crowd Detection System is a starting point for making crowd monitoring better in the future. The Crowd Detection System is useful, for safety and intelligent surveillance of the Crowd Detection System.

9. FUTURE WORK

To make the Crowd Detection System work better and handle people we can look at a few things in the future. We should compare StrongSORT. Botsort to see which one is better by looking at things like Dcount, Accuracy and FPS when they are used on the same hardware. If they are much the same we can use StrongSORT to get a more accurate count of people when we are not online.

We can also try using YOLOv8 and BotSort on devices like the NVIDIA Jetson by making them smaller, like

ONNX or TensorRT so the server does not get too busy and things happen faster in real time. If we use a database like SQLite instead of storing things in memory the Crowd Detection System will be more reliable and can start again if it stops working suddenly. Also if we use something called Zone-of-Interest counting we can see how people are moving in and out of a place, which gives us an idea of how crowds are moving and that is more useful, than just knowing how many people are there.

REFERENCES

[1] Y. Gai, W. He and Z. Zhou, "Pedestrian Target Tracking Based On DeepSORT With YOLOv5," *ICCEIC*, 2021.

[2] M. Abdou and A. Erradi, "Crowd Counting: A Survey of Machine Learning Approaches," *ICIoT*, 2020.

[3] P. Sivaprakash, M. Sankar, R. Chithambaramani and D. Marichamy, "A Convolutional Neural Network Approach for Crowd Counting," *ICOSEC*, 2023.

[4] Y. Yu, J. Liu, J. Wang, Q. Tan and Q. Zheng, "Vehicle Detection and Multi-Target Tracking System in Intelligent Transportation using YOLOv8 and Adaptive Deep Sorting Algorithm," *ICDCECE*, 2025.

[5] N. Wojke, A. Bewley and D. Paulus, "Simple Online and Realtime Tracking with a Deep Association Metric," *ICIP*, 2017.

[6] Y. Sun and P. Sun, "An Online DeepSORT-Based Vehicle Tracking Method for Supporting Realtime Vehicle Surveillance," *INSAI*, 2023.

[7] A. D. Abadi, Y. Gu, I. Goncharenko and S. Kamijo, "Detection of Cyclist's Crossing Intention Based on Posture Estimation for Autonomous Driving," *IEEE Sensors Journal*, 2023.

[8] W. Lu, "Safety Helmet Detection System Based on YOLOv5 Algorithm," *AIIM*, 2024.

[9] J. Wang, W. Hongjun, J. Liu, R. Zhou, C. Chen and C. Liu, "Fast and Accurate Detection of UAV Objects Based on Mobile-YOLO Network," *WCSP*, 2022.

[10] B. Jin, "Unlocking the Potential of Raw Images for Object Detection with YOLOv8 and BOT-SORT Techniques," *ICMLCA*, 2024.

[11] G. Tang, Z. Peng, B. Yin, J. Zhu, J. Dong and X. Bao, "Tracking Players in Volleyball Matches using Vol-Bot-SORT," *ICCEA*, 2024.

[12] "Crowd Density Estimation Based on Rich Features and Random Projection Forest," *IEEE Conference Publication*, 2016.

[13] "A Novel Multiple Hypothesis Testing (MHT) Scheme for Tracking of Dim Objects," *IEEE Conference Publication*, 2015.

[14] "Bayesian Poisson Regression for Crowd Counting," *IEEE Conference Publication*, 2009.