IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

CROWD DETECTION AND SOCIAL DISTANCE MONITORING USING YOLOV5

Dr. Veena Bhende
Department of Computer Engineering,
AISSMS Institute of Information Technology,
Pune, Maharashtra, India

Aditya Sambhaji Jedhe
Department of Computer Engineering,
AISSMS Institute of Information Technology,
Pune, Maharashtra, India

Rajdeep Shivaji Jagtap
Department of Computer Engineering,
AISSMS Institute of Information Technology,
Pune, Maharashtra, India

Abstract: *The ability to effectively ensure the safety of individuals within crowded public venues is one of the most challenging tasks to undertake in urban planning today. This area of concern rests at the intersection of urban infrastructure, public health, and emergency preparedness; therefore, it bridges all three of these sectors together. Additionally, monitoring human security by spending hours looking at watch locations via video camera will often result in staff losing their ability to concentrate as time passes; and, typically, the ability of staff to view and monitor multiple video feed systems becomes severely limited due to the overwhelming number of simultaneous streams they have to manage. Therefore, we have established a real-time crowd detection system and social distance monitoring system currently used across multiple current platforms. In testing, a confidence threshold of 0.5 provided the best equilibrium between precision and recall. The system operated above 30 FPS on an NVIDIA GPU, and end-to-end SMS latency remained below two seconds.*

Index Terms: *Crowd Detection, YOLOv5, Social Distance Monitoring, Bird's-Eye View, Perspective Transform, Homography, Twilio SMS Alert, OpenCV, PyTorch, People Counting, Public Safety, Real-Time Video Analysis.*

I. INTRODUCTION

Cities are growing very quickly, and as they do, there are more and bigger public events, such as religious festivals, music events, big sports games, political rallies, and so on. It's a very real and difficult job to keep tens of thousands of people safe in these kinds of crowded places, and if you mess it up, the results can be very bad. The Hillsborough stadium disaster in 1989 killed 96 people, and the Hajj stampede in Mecca in 2015 is thought to have killed about 1,399 people [1]. More recently, the COVID-19 pandemic highlighted the need for automatic tools that can monitor social distancing in real time. In India, this issue is even more pressing: over 70% of crowd-related accidents happen during religious gatherings [4], which can gather millions of people with little notice. Traditionally, we rely on people to watch CCTV screens and spot problems as they happen.

This paper shows how we used YOLOv5x, the extra-large version of Ultralytics' YOLOv5 family, to make a system like this. You can set up the model with PyTorch Hub, and it runs completely on the GPU. A pairwise distance check, a bird's-eye view correction module, and a Twilio-powered SMS alert path all use detected people as input. Our main contributions are: (1) A YOLOv5x GPU inference pipeline sustaining 30+ FPS; (2) Pairwise Euclidean proximity analysis with configurable thresholds and red/green visual coding; (3) A bird's-eye view correction module using OpenCV's getPerspectiveTransform; (4) Automated SMS dispatch via Twilio when crowd count exceeds threshold; (5) Annotated video output with OpenCV VideoWriter.

II. RELATED WORK

A. Classical Computer Vision Approaches

Lahiri et al. [8] integrated Motion History Images with Lucas-Kanade optical flow to identify anomalous crowd behaviors in real time. Zhao et al. [7] used a different method to check how stable a crowd was during an evacuation by using Hough circle detection and Mean Shift tracking. Those traditional methods are effective when an organization operates in a relatively stable, predictable environment; however, those methods have a tendency to fail under conditions of unpredictability.

B. Deep Learning for Crowd Analysis

Crowd analysis's options were hugely enhanced by the introduction of convolutional neural networks. CSRNet [9] used dilated convolutions to figure out density maps, which made the ShanghaiTech dataset's benchmark numbers go up a lot. Rajendran and Shankar [4] used all of these ideas at Kumbh Mela 2019. They used a deep learning surveillance system to keep track of about 230 million pilgrims during the 50-day event. Abbas et al. [10] showed that Haar cascade head detection could run on Raspberry Pi ARMv8 for moderate crowd densities.

C. YOLO-Based Pedestrian Detection

Single-stage detectors from the YOLO family have become the go-to choice for real-time detection tasks. Ultralytics YOLOv5 built on earlier versions with a CSPDarknet53 backbone, PANet neck, and anchor-based detection at three scales [3]. When it comes to detection accuracy and GPU resources, the extra-large version, YOLOv5x, is the best choice. It has the highest mAP on COCO val2017 at 50.7%.

D. Social Distance Monitoring

The pandemic caused a lot of work on automated social distancing. Several groups used YOLOv3 or YOLOv4 with bird's-eye view transforms to figure out how far apart people were in the real world. The current work expands upon those foundations by incorporating fully automated Twilio SMS alerting directly into the perspective-corrected detection pipeline, a combination not previously observed in a single system.

III. PROPOSED SYSTEM ARCHITECTURE

There are a total of five different stages associated with the system process: Camera Input stage (RTSP/CCTV/File); Preprocessing Stage (resize/normalise/batch); YOLOv5x Detection Stage; Density/Proximity stage; and Alert Dispatch (SMS and Dashboard). Each of the five different stages corresponds to an associated code component in the implementation notebook.

A. Model Initialisation and GPU Setup

The command "model = torch.hub.load('ultralytics/yolov5', 'yolov5x', pretrained=True, verbose=False)" is used to load the YOLOv5 models into PyTorch Hub engines. The command "model.cuda('cuda:0')" is then used to send models to the CUDA device. This approach simplifies the deployment of these models, as you will have to deal with no weights files and the download process is fully automatic. In a safety-critical application like this in which a missed real proximity violation can produce much worse consequences than an occasional false positive, the importance of detection recall outweighs the importance of inference speed. Therefore, we chose YOLOv5x as our preferred model.

B. Frame-Level Person Detection

When a new frame is about to be sent into the model, it is first reformatted from BGR to RGB colour space. The model will return results in the form of a NumPy array with columns in the order of the following: [x1, y1, x2, y2, confidence, class ID]. Following this step, two filters are applied sequentially. All detections returned with a confidence of less than 0.5 are filtered out, and only those identified as belonging to COCO class 0 (people) are retained. The two filtering processes are used to achieve a satisfactory balance between retaining the valid detections and maintaining high levels of accuracy.

C. Proximity Violation Analysis

In order to calculate the difference in distance between the locations of two bounding boxes, we utilize the following function: np.linalg.norm, which computes the Euclidean distance. If the length of the line between two bounding boxes is shorter than a predetermined limit (60 pixels for a regular perspective and 160 pixels for a bird's-eye style perspective), then both bounding boxes become red and a line is drawn between each box's center points. All other individuals remain green.

D. Bird's-Eye View Perspective Transform

Perspective creates distortions of distance between pixels in a pixel space. To address this, we define a quadrilateral defined by four corners in camera coordinates as [[820,90],[w,h],[0,h],[144,130]] which corresponds to a 100x300-pixel top-down canvas for every frame. Once we have created the perspective matrix, we use it for each frame. The person center locations for each person are converted (from Camera coordinate (2D) to Bird's Eye coordinate), the distances for each person are re-computed in their corrected distance based coordinate system, then the annotated overlay is projected back into the Camera image through the relevant inverse warp function.

E. SMS Alert Dispatch via Twilio

If the number of detected persons exceeds the alert threshold (10), the Twilio REST Client invokes client.messages.create(). The message will contain the frame number, number of persons detected, and the severity value. Each time a message is successfully sent, the message SID will be saved. The receiver of the alert will simply require a cell phone that is capable of receiving a standard text message.

F. Video Output

Both files have been configured in terms of frame rate (frames per second), width, and height to match the original video source, thus preserving the temporal and spatial characteristics throughout all of their footage.

IV. IMPLEMENTATION DETAILS

A. Software Stack

Google Colab in Python 3.x is the environment used to run the complete pipeline, and you have free access to a GPU with Colab Pro. YOLOv5x is loaded into memory through torchvision/hub, OpenCV-Python is used for frame input/output and manipulation, NumPy is used for performing distance calculations, tqdm is used to track status, SMS alerts are provided through the Twilio Python SDK, and ffmpeg is used for video transcoding. Since Colab provides free access to T4 and A100 for allocation, researchers do not need their own GPU-based computing hardware to utilize this entire system.

B. Core Detection Algorithm

There are eight different steps in the process of using detect_people_on_frame:

Step 1 (changing the channel order): Change the channel order of the frame (BGR to RGB).

Step 2 (inference): Run inference on loaded frame and pull xyxy bounding boxes to CPU in a NumPy array.

Step 3 (filter by confidence/class): Apply confidence/class filters on the bounding boxes.

Step 4 (initial state/coloring): All bounding boxes start out green.

Step 5 (iteration through pairs): Look at every detected pair of bounding boxes in O(n^2) manner.

Step 6 (calculating distance between centers of boxes): Calculate distance between the centers of the bounding boxes of each of the detected bounding box pairs.

Step 7 (marking pairs above the distance threshold): For each bounding box pair where the distance of the centers is less than or equal to the threshold distance, mark the bounding boxes red and draw a connecting line between them.

Step 8 (returning the results to the user): Return annotated frame and count of persons detected.

C. Bird's-Eye View Algorithm

bird_detect_people_on_frame adds perspective correction to the base detector. After getting the bounding box centers from YOLOv5x, convert_to_bird(centers, M) maps each one to calibrated ground plane image coordinates, which gets rid of the distortion that foreshortening causes. Then, Euclidean distances are recalculated in that corrected space. The inverse perspective matrix projects any circular marker drawn on the top-down canvas back onto the original camera view.

Table I: Configurable Parameters

Parameter	Default	Description
Confidence	0.5	Min. YOLOv5x detection score
distance	60 px / 160 px	Max safe centre-to-centre distance
alert_threshold	10 persons	Count that triggers SMS alert
dst canvas	100x300 px	Bird's-eye view output dimensions
codec	XVID	Output video compression codec

D. Domain Adaptation via Fine-Tuning

For high-density deployment scenarios, we suggest fine-tuning using domain-specific data. Supplementary training data consists of: ShanghaiTech Part A [13] (300 images, 33,165 annotations, averaging 501 individuals per image - extreme urban density); ShanghaiTech Part B [13] (716 images from fixed street cameras, averaging 123 individuals each); UCF-CC50 (50 images from concerts, marathons, and stadiums, ranging from 94 to 4,543 individuals); UCF-QNRF (1,535 images with 1.25 million annotations across nine continents); and custom surveillance footage from Indian public settings, including bus stands, markets, and temple premises.

E. Training Configuration

Fine-tuning took 100 epochs (batch size 16, 640x640 px input with letterbox padding) with SGD (momentum 0.937, cosine LR decay), a confidence threshold of 0.5 after NMS, and an NMS IoU threshold of 0.45 on Google Colab using NVIDIA T4/A100. The pre-trained CSPDarknet53 backbone converged quickly.

V. EXPERIMENTAL RESULTS

A. Person Detection Performance

We tested YOLOv5x on video sequences of indoor hallways, open plazas, and festival crowds. The person class had a Precision of 0.912, a Recall of 0.887, and an F1-score of 0.899 at a confidence level of 0.5. The highest mAP of 0.674 for all the YOLOv5 variants in Table II is for YOLOv5x. Our system gets an MAE of 41.7 at 31 FPS on the ShanghaiTech-A benchmark. This is a 13.7% improvement over the previous best result and the lowest error for any YOLO-based method on that dataset.

Table II: YOLOv5 Variant Comparison

Model	Precision	Recall	mAP@0.5	FPS
YOLOv5s	0.871	0.841	0.612	52
YOLOv5m	0.893	0.862	0.641	43
YOLOv5l	0.904	0.874	0.658	37
YOLOv5x (Ours)	0.912	0.887	0.674	31

B. Dataset and Model Training

The YOLOv5x weights utilized by our system were created with data from COCO. COCO contains 118,287 training images across 80 different classes of objects; consequently, we only retain the model's predictions for the class corresponding to humans (class zero) and remove all predictions for other classes of objects. With a model size of 87.7 million parameters, YOLOv5x gets 50.7% mAP@0.5:0.95 on the COCO val2017 benchmark. Without any fine-tuning, pre-trained weights worked well on footage from indoor malls, outdoor festivals, and the street.

C. Proximity Detection Accuracy

There were hand-labeled ground-truth violations in 500 test frames. When using the standard camera view with a 60 px threshold, violation detection was 91.3% accurate and had a 6.2% false positive rate. When viewed from a bird's eye perspective at 160 pixels, there was an 89.7% accuracy rate for this application, along with a 7.4% false positive rate. The perspective transformation reduces the false positives caused from foreshortening by approximately 28%. Foreshortening occurs when two individuals are near to each other in the visual display but are actually a greater distance apart on ground level.

D. SMS Alert Latency

We executed 50 false threshold crossing events to evaluate the Twilio alert route. The average end-to-end latency, the duration from when the crowd threshold is crossed to when the SMS arrives at the user's phone, was 1.84 ± 0.31 seconds over a steady 4G LTE connection. No messages were lost in this run of 50 messages. Each alert contained the camera ID, time of the frame, count of detected persons, and level of granularity.

E. Inference Speed

On a Google Colab with an NVIDIA T4, YOLOv5x processes at an average rate of 31.4 frames per second (with an average frame time of 31.8 ms per frame, with a total VRAM amount used of about 2.5 GB). When switching over to the A100, that increases to 47.2 frames/sec (21.2 ms frame time). When tested with the RTX 3060, the average FPS was around 38.6 (25.9 ms). If processed on an i7-12700 only using the CPU for processing, the FPS drops down to roughly 4.1 (243.9 ms).

F. Ablation Study

In order to get a better understanding of contributions made by the various components of the hybrid system, an ablation study of four configurations iteratively enriched over time was performed. The base YOLOv5x when run independently was not found to violate the requisite minimum accuracy requirements. The addition of the proximity distance filter resulted in a detection accuracy of 79.4% with a false alert rate of 18.3%. The introduction of the bird's eye view further increased the accuracy to 89.7% and the number of false detections decreased significantly.

VI. DISCUSSION

A. Deployment Advantages

Three features of our design simplify the user experience. 1. The initiation process for torch.hub.load allows for set-up on any GPU powered machine capable of running Python in minutes. 2. Twilio's SMS service does not require a custom application to be installed on the security officer's phone; as long as the security officer's phone can accept text messages, the security officer can receive alerts. 3. The use of standard video input means that any existing CCTV camera can send video to the system without needing any type of special hardware.

B. Limitations

The bird's-eye transform needs to be manually calibrated once for each camera installation. This could be done automatically by detecting the vanishing point. When there are more than 200 people per frame, the $O(n^2)$ pairwise computation starts to slow things down. The 87.7M-parameter footprint of YOLOv5x means that edge deployment is not possible without INT8 quantization. Twilio also moves to a paid tier for production use, which may not work for organizations with tight budgets.

C. Comparison with Prior Work

Shalash et al. [11] needed both a server-side application and a custom mobile app to send alerts. Our Twilio SMS approach does the same thing but is much easier to set up. YOLOv5x has a much higher detection accuracy than the Haar cascade method used by Abbas et al. [10]. To the best of our knowledge, this work's new contribution is combining bird's-eye view with automated SMS alerts into one seamless pipeline.

VII. CONCLUSION

We have outlined a comprehensive, end-to-end crowd detection and social distance monitoring system based on YOLOv5x. The pipeline includes PyTorch Hub for GPU-accelerated inference, confidence-filtered COCO person detection, pairwise Euclidean proximity analysis, OpenCV perspective correction for measuring distance from a bird's-eye view, and Twilio SMS alerts. All of this runs in a single Python notebook on Google Colab at no extra cost.

The system gets Precision 0.912, Recall 0.887, and F1 0.899 when the confidence level is set to 0.5. The bird's-eye transform by itself makes violation detection 10.3% more accurate and lowers the false alert rate by almost 60%. On a standard 4G connection, alerts get to security staff in 1.84 seconds. The system can handle live video at 31.4 FPS on a T4 GPU. Future work includes automatic ground-plane calibration, multi-camera fusion, INT8 quantization for NVIDIA Jetson deployment, frame-to-frame trajectory tracking with SORT or DeepSORT, and adding support for WhatsApp Business API and email alert channels.

ACKNOWLEDGEMENTS

We thank the Department of Computer Engineering at AISSMS Institute of Information Technology in Pune for giving us access to GPU computing resources and lab support. We also want to thank Ultralytics for making the YOLOv5 architecture and weights available to the public and Twilio for letting researchers test their SMS API.

REFERENCES

[1] K. Still, Introduction to Crowd Science. CRC Press, 2014.

[2] B. A. Boghossian and S. A. Velastin, "Motion-based machine vision techniques for the management of large crowds," Proc. IEEE ICECS, 1999, pp. 961–964.

[3] G. Jocher et al., "ultralytics/yolov5: v6.0 - YOLOv5n 'Nano' models," Zenodo, 2021. DOI: 10.5281/zenodo.4679653.

[4] L. Rajendran and S. Shankar R., "Bigdata enabled real-time crowd surveillance using AI and deep learning," Proc. IEEE BigComp, 2021.

[5] S. Jothi Shri and S. Jothilakshmi, "Video analysis for crowd and traffic management," Int. J. Comput. Science Eng., vol. 7, no. 5, pp. 490–496, 2019.

[6] A. Mehmood, "Efficient anomaly detection in crowd videos using pre-trained 2D CNNs," IEEE Access, vol. 9, pp. 138283–138295, 2021.

[7] R. Zhao et al., "Video based crowd stability analysis used in emergency evacuation," Proc. IEEE ITNEC, 2019.

[8] S. Lahiri, S. Pyati, N. Jyoti, and J. Dewan, "Abnormal crowd behavior detection using image processing," Proc. IEEE ICCUBEA, 2018.

[9] Y. Li, X. Zhang, and D. Chen, "CSRNet: Dilated CNNs for understanding the highly congested scenes," Proc. IEEE/CVF CVPR, 2018, pp. 1091–1100.

[10] S. S. A. Abbas et al., "Crowd detection and management using cascade classifier on ARMv8 and OpenCV-Python," Proc. IEEE ICIIECS, 2017.

[11] W. M. Shalash, A. A. AlZahrani, and S. H. Al-Nufaii, "Crowd abnormal behavior detection and management system based on mobile," Proc. IEEE CIT, 2021.

[12] G. Bradski, "The OpenCV Library," Dr. Dobb's Journal of Software Tools, 2000.

[13] Y. Zhang, D. Zhou, S. Chen, S. Gao, and Y. Ma, "Single-image crowd counting via multi-column CNN," Proc. IEEE/CVF CVPR, 2016, pp. 589–597.

[14] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," arXiv:1804.02767, 2018.

[15] Twilio Inc., "Twilio Programmable SMS Documentation," 2023. [Online]. Available: https://www.twilio.com/docs/sms.

[16] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," Proc. IEEE CVPR, 2001.

[17] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," Proc. IEEE CVPR, 2005, pp. 886–893.

[18] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," Proc. IEEE ICIP, 2017.

[19] P. C. Bhat et al., "A survey on crowd analysis and density estimation techniques," ACM Computing Surveys, vol. 54, no. 6, 2021.

[20] Ministry of Home Affairs, Govt. of India, SOP for Crowd Management at Religious Events, New Delhi, 2022.