Detecting Self-Inflicted Violence In High-Raised Area

Priykrit Varma
Computer Science And Engineering
International Institute of
Information Technology
Naya Raipur, India
priykrit21100@iiitnr.edu.in

***Abstract*—The goal of the research is to create a system that uses computer vision methods to identify suicidal behaviour in high-rise and bridge regions. The technology uses real-time video monitoring to To detect whether an individual is exhibiting self-harm or standing in a hazardous location. The system analyses the video feed to find those who may be at risk using a combination of deep learning algorithms and computer vision techniques. When a suspected Self-Inflicted Violence attempt is detected, the system notifies the appropriate authorities, such as emergency medical services or law enforcement, so they can take action and stop it. The suggested approach might drastically lower the frequency of Self-Inflicted Violence in high-risk areas, offering a vital resource to aid in tragedy prevention and lifesaving.**

I. Introduction

A. Background and Motivation

With over 700,000 suicide deaths per year worldwide, it is a severe global public health concern. Jumping off a high structure, like a bridge or a tall building, is one of the most popular ways to commit suicide. Suicide in such places not only poses a major hazard to the person attempting it, but also causes mental distress for onlookers and emergency personnel.

Physical obstacles or heightened surveillance are just two examples of traditional Self-Inflicted Violence prevention measures that may or may not work in these places. In order to promptly identify people displaying suicidal behaviour and notify the appropriate authorities to intervene, innovative and efficient Self-Inflicted Violence detection systems are required. Recent years have seen encouraging outcomes in the use of deep learning algorithms and computer vision techniques. Real-time video can be used to identify human behaviour.[1] The goal of this study is to use these methods to create a system for identifying suicidal behaviour near bridges.

B. Problem Statement

One of the most popular ways to commit Self-Inflicted Violence is by leaping from a height, which is a problem for global public health. Witnesses and first responders may suffer serious mental stress as a result of this conduct, which frequently takes place in dangerous locations like bridges or tall buildings.[2]

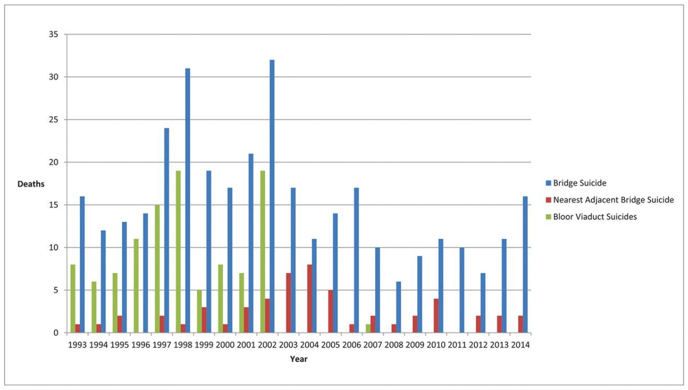


Fig. 1. suicide cases in past

Physical barriers and greater surveillance are two common traditional Self-Inflicted Violence prevention methods, although they have drawbacks and aren't always successful at stopping Self-Inflicted Violence attempts. As a result, there is a need for creative and efficient Self-Inflicted Violence detection systems that can immediately spot people acting suicidally and notify the proper authorities to take action.[2]

C. Objectives

To create a system for real-time video surveillance that can identify Self-Inflicted Violence behaviour near high raise places. [1]

To use deep learning algorithms and computer vision techniques to analyse people's activities and movements in order to spot those who could be at risk of Self-Inflicted Violence. [2]

To provide a method for alerting and contacting relevant authorities, like emergency medical services or law enforcement, in order to stop probable suicide attempts.[3]

To make suggestions for further study and the creation of Self-Inflicted Violence detection systems in high-risk regions. The major goals of the suggested system are to stop possible

Self-Inflicted Violence attempts, lessen the emotional toll on witnesses and first responders, and aid in preventing monetary losses brought on by Self-Inflicted Violence in public places.[1]

D. Research Methodology

Literature Review

The first step of this research will be to conduct a comprehensive literature review of existing research on suicide prevention strategies in bridge areas, computer vision techniques, and deep learning algorithms to detect whether an individual is exhibiting suicidal behaviour or standing in a hazardous location. The literature review will provide a foundation for identifying suitable techniques and algorithms for this project and help to address research gaps.[1]

Development of Systems

The Python programming language and the TensorFlow machine learning framework will be used to create the suggested system. The system will examine people's movements and actions in order to identify those who could be at risk of Self-Inflicted Violence. It will do this by using computer vision techniques and deep learning algorithms. When a suspected Self-Inflicted Violence attempt is detected, the system will notify the appropriate authorities, who can then take action to stop it.[3]

System Assessment

Accuracy, sensitivity, specificity, and false positive rates will all be taken into account when assessing the performance of the suggested system. Data gathered and testing conducted in actual bridge regions will be used in the evaluation. The effectiveness of the proposed system will also be evaluated in comparison to that of current Self-Inflicted Violence detection systems to determine its benefits and shortcomings.[5]

II. LITERATURE REVIEW

A. Self-Inflicted Violence Prevention Strategies

Video surveillance: Cameras can be installed in key locations to monitor activity and detect any signs of distress or danger. Machine learning algorithms can be used to analyse video to detect whether an individual is exhibiting suicidal behaviour or standing in a hazardous location

Crisis intervention teams: Crisis intervention teams composed of mental health professionals and law enforcement officers can be deployed to respond to Self-Inflicted Violence attempts or incidents of suicidal behaviour. [4]These teams can provide immediate support and connect individuals with appropriate resources.

Automatic alerts: When a Self-Inflicted Violence attempt is suspected, the Self-Inflicted Violence detection system can send an automated alarm with the location and incident information to local law enforcement or emergency medical services.[4]

B. Self-Inflicted Violence Detection Systems

Systems for detecting Self-Inflicted Violence are created to spot impending Self-Inflicted Violences or Violence attempts and notify the appropriate authorities. There are various Self-Inflicted Violence detection technologies that can be applied to a bridge project, such as Using cameras to monitor the bridge area and spot odd or suspicious activity are video-based systems. The video feed can be analysed using machine learning techniques to spot actions that suggest suicidal intent.[1]

The Self-Inflicted Violence detection system can warn local authorities or launch a predetermined response plan after a probable Self-Inflicted Violence has been identified. Sending crisis intervention teams or emergency medical services to the site of the probable Self-Inflicted Violence attempt may be part of the response strategy.

C. Computer Vision Techniques

Algorithms for object detection can be used to find people on bridges or in other high-risk areas. These people's movements can be analysed using machine learning algorithms to spot actions or patterns that might indicate an impending Self-Inflicted Violence attempt.

Algorithms for facial recognition can be used to spot people who might be at risk of committing Self-Inflicted Violence. Pose estimating algorithms can be used to examine how people move and position themselves on a bridge or in other high-risk areas. Algorithms for machine learning can be used to spot actions or trends that point to a Self-Inflicted Violence attempt.[3]

Unusual or aberrant actions on a bridge or in another high-risk region can be found using algorithms for anomaly identification. Machine learning algorithms can be trained on typical behavioural patterns and then used to spot abnormalities that can be signs of an impending Self-Inflicted Violence attempt. These are only a few instances of computer vision methods that might be applied in a project to detect Self-Inflicted Violence.

D. Deep Learning Algorithms

In order to find patterns and behaviours that might point to suicidal intent, deep learning algorithms can be used to evaluate data from a range of sources, such as video The following are some instances of deep learning algorithms that could be applied in a project to identify Self-Inflicted Violence:

Convolutional Neural Networks (CNNs) are capable of detecting persons and particular position, such as pacing or standing on the brink of a bridge, in image and video analysis. Long Short-Term Memory Networks (LSTMs) are capable of evaluating data sequences, such as movement patterns or vital sign measures, to spot alterations that would point to suicidal intent. Non-Maxima Suppression (NMS) can remove minor probability instances of detected object [5]

deep sort (Deep Simple Online Realtime Tracking) is an object tracking algorithm that can be used for motion tracking in behavior analysis, including Self-Inflicted Violence detection. It is a state-of-the-art algorithm that combines deep learning with classical tracking methods to achieve high accuracy and robustness in tracking.

These are just a few illustrations of deep learning algorithms that might be employed in a project to identify Self-Inflicted Violence. The nature and complexity of the data being examined, as well as the resources available for training and deploying the models, will determine the specific algorithms that are employed.

E. Relevant Case Studies

The Korea Advanced Institute of Science and Technology (KAIST) research from 2019 is one pertinent case study for Self-Inflicted Violence detection in bridges. In this study, the researchers created a deep learning-based Self-Inflicted Violence detection system that use security cameras to keep an eye on bridges and other high-risk locations.

The system analyses video data to find people who could be at danger of Self-Inflicted Violence using a combination of computer vision techniques and deep learning algorithms. The technology can identify when someone is leaning over a bridge or engaging in other actions that would indicate suicidal intent, such pacing back and forth. When a person is deemed to be at risk, the system notifies a nearby control centre, which can then send emergency personnel to the area. The device can also offer real-time feedback to the person in danger by playing pre-recorded messages over a loudspeaker to persuade them to ask for assistance or think again about their course of action. [1]

The KAIST project's high level of at-risk person detection accuracy was tested on a bridge in Seoul, South Korea. During the trial period, the system was also proven to be successful in lowering the number of Self-Inflicted Violence attempts on the bridge.

With Self-Inflicted Violence detection systems for bridges and other high-risk sites, this case study illustrates the potential efficacy of utilising computer vision and deep learning approaches. It is crucial to remember that additional study and testing are required to make sure that these systems are precise, dependable, and efficient in real-world settings.[1]

III. SYSTEM DESIGN

A. Data used

Video data: It is possible to identify people who could be at danger of Self-Inflicted Violence by analysing video data from security cameras or other sources. To find pertinent patterns or characteristics in video data, computer vision and machine learning algorithms may be used.[1] We are analysing camera footage to determine whether the individual is standing in a railing to commit Self-Inflicted Violence or is in a dangerous position in a high elevated region.

The appropriate aspects in the video data, such as movement patterns, body language that could be a sign of suicidal conduct, can be found using computer vision techniques. For instance, computer vision algorithms can be trained to recognise sudden stops, pacing, or other odd movements that might be signs of suicidal ideation.

B. Flow of frame feeds

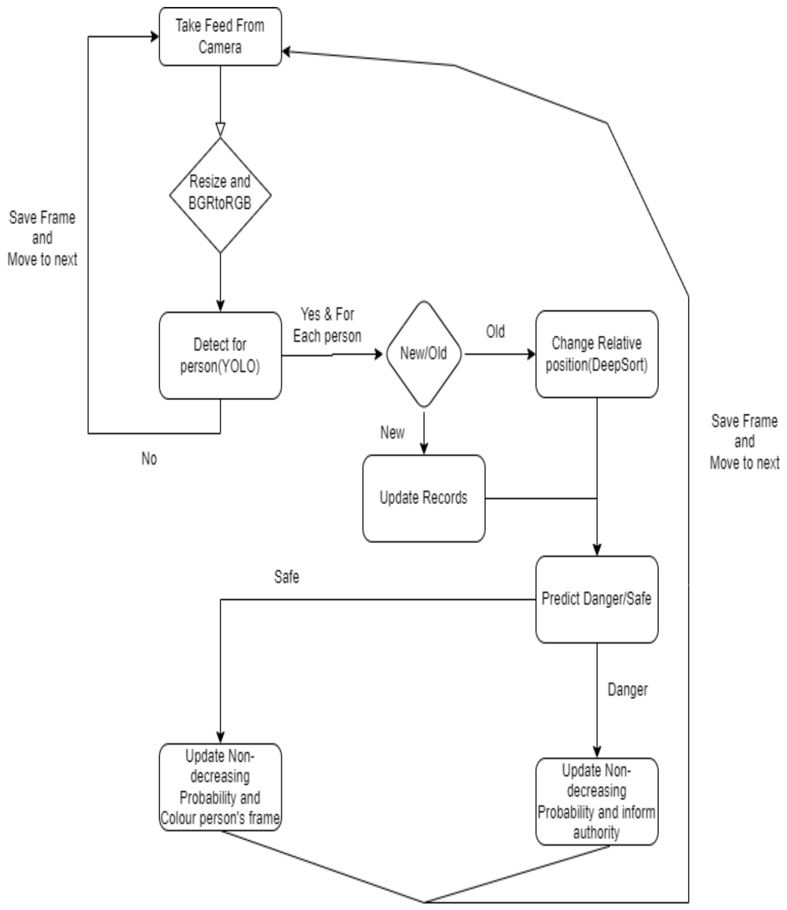

Take Feed From Camera
Resize and BGRtoRGB
Save Frame and Move to next
Detect for person(YOLO)
Yes & For Each person
New/Old
Old
Change Relative position(DeepSort)
No
New
Update Records
Save Frame and Move to next
Safe
Predict Danger/Safe
Danger
Update Non-decreasing Probability and Colour person's frame
Update Non-decreasing Probability and inform authority

Fig. 2. Flow of frame feeds

C. Video Processing Techniques

The analysis of video data from security cameras using video processing techniques can be utilised to spot people who might be at danger of Self-Inflicted Violence.[2] In efforts aimed at preventing Self-Inflicted Violence, the following typical video processing methods are employed:

Object detection is a method for locating particular individuals or objects in a video feed. Object detection can be utilised in Self-Inflicted Violence prevention initiatives to spot individuals who are perched precariously on a bridge or exhibiting other risky behaviours and also Detecting coordinates of body parts in a given frame and draw conclusion.

D. Alert and Intervention Mechanisms

Projects to prevent Self-Inflicted Violence often include alert and intervention systems. The right treatments can be started to stop harm if a person is recognised as being at risk of Self-Inflicted Violence.

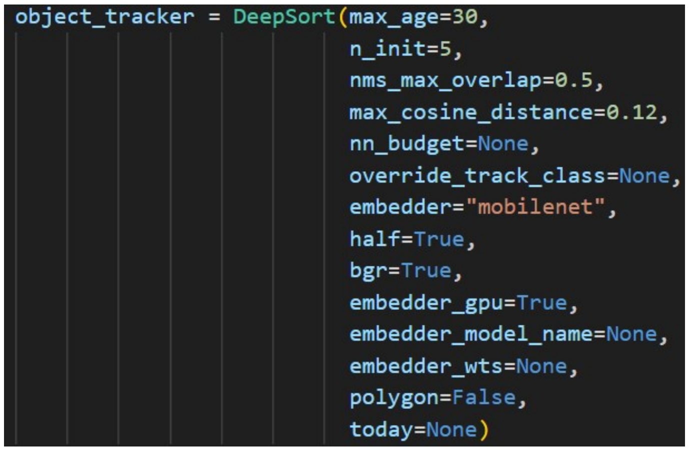
Systems of notification: When a person is suspected of being suicidal, notification systems can be set up to notify the appropriate authorities, assistance networks, or carers. These alerts may be triggered manually by operators or automatically based on the results of prediction algorithms.

Teams that respond to emergencies: Teams that respond to emergencies might be called in to offer rapid help to people who could be suicidal risk. These teams may include of trained law enforcement officers, first responders, or mental health specialists.

IV. IMPLEMENTATION AND TESTING

A. *Deep-sort*

Deep SORT (Simple Online and Realtime Tracking with a Deep Association Metric) is a technique for tracking multiple objects in video sequences using deep learning. It extracts features of object appearance and motion and associates them across frames to track objects in real-time with high accuracy and robustness, even in crowded scenes. It finds applications in various fields like surveillance, robotics, and autonomous driving. Setting up of parameters is imortant base on the distance between camera and ROI.



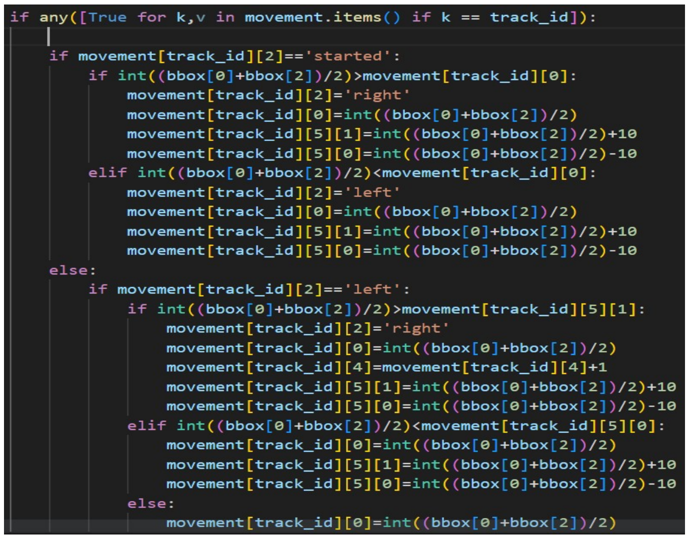
```
object_tracker = DeepSort(max_age=30,
                          n_init=5,
                          nms_max_overlap=0.5,
                          max_cosine_distance=0.12,
                          nn_budget=None,
                          override_track_class=None,
                          embedder="mobilenet",
                          half=True,
                          bgr=True,
                          embedder_gpu=True,
                          embedder_model_name=None,
                          embedder_wts=None,
                          polygon=False,
                          today=None)
```

Fig. 3. DeepSort Initialization in Project

Parameter used during the initialization of the DeepSort object tracker.

- maxage: The maximum number of frames that a track can be missed before it is deleted. In other words, this parameter determines the maximum age of a track (in frames) before it is considered inactive and removed from the tracker.
- ninit: The minimum number of detections required to initiate a track. If a track does not have enough detections in its history to meet this threshold, it will not be initialized.
- nmsmaxoverlap: The maximum allowed overlap between two detections in order to perform non-maximum suppression (NMS). This parameter helps to remove redundant detections and improve the quality of the tracking results.
- maxcosinedistance: The maximum cosine distance allowed between two embeddings in order to consider them as part of the same track. This parameter helps to determine whether two detections belong to the same object or not.
- nnbudget: The maximum number of embeddings that can be stored in memory at any given time. If the number of embeddings exceeds this budget, the oldest embeddings will be discarded.
- overridetrackclass: Allows the user to override the default track class used by the tracker.
- embedder: The name of the feature extraction network to use. In this case, the ”mobilenet” architecture is used.
- half: Whether to use half-precision floating point arithmetic for the feature extraction network. This can help to speed up the computation and reduce memory usage.
- bgr: Whether to use BGR image format for input to the feature extraction network.
- embeddergpu: Whether to use GPU acceleration for the feature extraction network.
- embeddermodelname: The name of the pre-trained feature extraction network to use. If set to None, the default network for the chosen architecture will be used.
- embedderwts: The path to the weights file for the pre-trained feature extraction network.
- polygon: Whether to use polygon instead of bounding box. If polygon is enabled, the tracker will use polygon instead of bounding box for detections.
- today: The date to use as the reference date for the tracker. If set to None, the current date will be used.

B. *Record updataion*



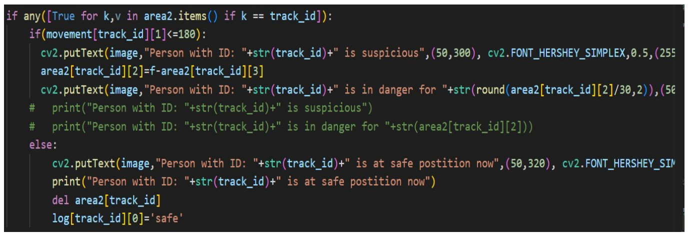
```
if any([True for k,v in movement.items() if k == track_id]):

    if movement[track_id][2]=='started':
        if int((bbox[0]+bbox[2])/2)>movement[track_id][0]:
            movement[track_id][2]='right'
            movement[track_id][0]=int((bbox[0]+bbox[2])/2)
            movement[track_id][5][1]=int((bbox[0]+bbox[2])/2)+10
            movement[track_id][5][0]=int((bbox[0]+bbox[2])/2)-10
        elif int((bbox[0]+bbox[2])/2)<movement[track_id][0]:
            movement[track_id][2]='left'
            movement[track_id][0]=int((bbox[0]+bbox[2])/2)
            movement[track_id][5][1]=int((bbox[0]+bbox[2])/2)+10
            movement[track_id][5][0]=int((bbox[0]+bbox[2])/2)-10
    else:
        if movement[track_id][2]=='left':
            if int((bbox[0]+bbox[2])/2)>movement[track_id][5][1]:
                movement[track_id][2]='right'
                movement[track_id][0]=int((bbox[0]+bbox[2])/2)
                movement[track_id][4]=movement[track_id][4]+1
                movement[track_id][5][1]=int((bbox[0]+bbox[2])/2)+10
                movement[track_id][5][0]=int((bbox[0]+bbox[2])/2)-10
            elif int((bbox[0]+bbox[2])/2)<movement[track_id][5][0]:
                movement[track_id][0]=int((bbox[0]+bbox[2])/2)
                movement[track_id][5][1]=int((bbox[0]+bbox[2])/2)+10
                movement[track_id][5][0]=int((bbox[0]+bbox[2])/2)-10
            else:
                movement[track_id][0]=int((bbox[0]+bbox[2])/2)
```

Fig. 4. Record Updation while person is in motion

The system is designed to track multiple individuals, each with a unique identifier. It records the movement of each person by tracking changes in the horizontal and vertical coordinates of their bounding boxes(Fig. 4). When either coordinate changes, the system updates the movement direction and counts the number of times the person changes direction. Additionally, it records how long they remain in frame sequences.

C. Display of Message



```
if any([True for k,v in area2.items() if k == track_id]):
    if(movement[track_id][1]<=180):
      cv2.putText(image,"Person with ID: "+str(track_id)+" is suspicious",(50,300), cv2.FONT_HERSHEY_SIMPLEX,0.5,(255
      area2[track_id][2]=f-area2[track_id][3]
      cv2.putText(image,"Person with ID: "+str(track_id)+" is in danger for "+str(round(area2[track_id][2]/30,2)),(50
    #   print("Person with ID: "+str(track_id)+" is suspicious")
    #   print("Person with ID: "+str(track_id)+" is in danger for "+str(area2[track_id][2]))
    else:
        cv2.putText(image,"Person with ID: "+str(track_id)+" is at safe postition now",(50,320), cv2.FONT_HERSHEY_SIM
        print("Person with ID: "+str(track_id)+" is at safe postition now")
        del area2[track_id]
        log[track_id][0]='safe'
```

Fig. 5. Message about status

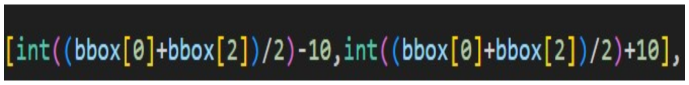
The program first verifies if the current individual being monitored has a unique identifier assigned by the tracker. Once the ID is identified, the program evaluates the person's movements to determine whether they are at risk. If the person's movements are suspicious, the program adds a message to the video frame displaying their ID and the duration of time they have been in danger(Fig. 5). Conversely, if the person's movements are safe, the program generates a message indicating their ID, updates their status in the log to ”safe”, and removes their ID from the list of tracked IDs. The log keeps track of the status of each person, which can either be ”safe” or ”danger”. Whenever a person's status is changed from ”danger” to ”safe”, the log is updated.

D. Curse of NMS

It is possible to generate different bounding boxes for the same person when it passes through the same image multiple times. This can happen due to various reasons such as changes in the person's pose, lighting conditions, occlusions, and changes in the background. So if the lighting changes NMS can suppress the original coordinates of bounding boxes and assign new ones. This would mean that person is in motion which is incorrect and the person is still stationary. To avoid this we introduced buffer regions(Fig. 6) around the points which would mean that on crossing a certain threshold only a person would be considered in motion and which greatly reduces the risk of false detection of motion and keeps the probability in check. For the same reason, the Region of Interest(ROI) had to be predefined because if trained to identify then NMS can create varying bounding boxes which could give ambiguous results. (Fig. 8)

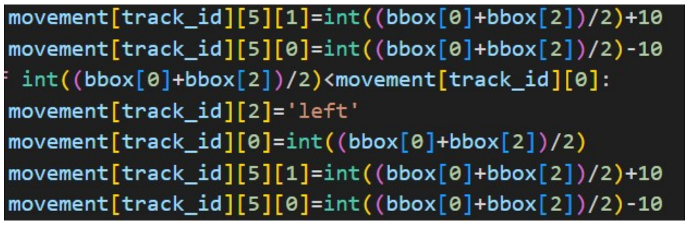
E. Updating non-decreasing Probability and Color on the Frame

The probability is calculated by distributing 85 to horizontal changes(Maximum of 10), 85 to vertical changes(Maximum of 10), and 40 to time spent(300 seconds) in the range of 0-255(reserving 40 for the future), then calculating the effective color of bounding boxes and non - decreasing probability for individual unique id.(Fig. 9)



```
[int((bbox[0]+bbox[2])/2)-10,int((bbox[0]+bbox[2])/2)+10],
```

Fig. 6. Creation buffer region with 10 units



```
movement[track_id][5][1]=int((bbox[0]+bbox[2])/2)+10
movement[track_id][5][0]=int((bbox[0]+bbox[2])/2)-10
 int((bbox[0]+bbox[2])/2)<movement[track_id][0]:
movement[track_id][2]='left'
movement[track_id][0]=int((bbox[0]+bbox[2])/2)
movement[track_id][5][1]=int((bbox[0]+bbox[2])/2)+10
movement[track_id][5][0]=int((bbox[0]+bbox[2])/2)-10
```

Fig. 7. Updating buffer region

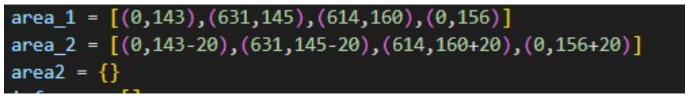
F. Hardware and Software Implementation

The hardware and software parts of the Self-Inflicted Violence detection system are installed and set up in this phase in accordance with the project requirements. The software includes computer vision algorithms, machine learning models, and other analytical tools for processing and evaluating the data, while the hardware may comprise cameras, sensors, and other devices for data collections.[3]

The system is then connected to any necessary alarm and response systems, such as notification systems, crisis hotlines, or emergency response teams.

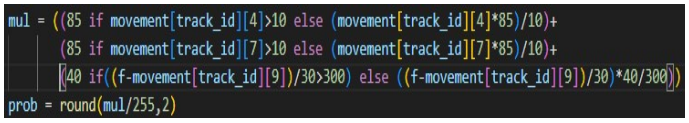
G. Assessment and Testing

The system must be tested and reviewed after installation and configuration to make sure it is operating as intended and achieving the project's objectives. Setting up of parameters is imortant base on the distance between camera and ROI. This could entail different kinds of testing, like: Functional testing: Checking the system's functioning to make sure it accurately and consistently detects and alerts on Self-Inflicted Violence conduct. Performance testing: Analysing how well a system performs under various circumstances, such as varying lighting or weather, to make sure it operates consistently. User acceptability testing: Verifying that the system's user interface and overall user experience are intuitive and simple to use. Testing for ethical and legal compliance: Making sure the system complies with all applicable ethical and legal requirements, such as those pertaining to data protection and privacy. Feedback from users and stakeholders, such as mental health experts, first responders, and persons who could be at risk of Self-Inflicted Violence, may also be gathered as part of the evaluation of the system's success. The system can be improved and made more efficient with the help of this feedback.[1] A Self-Inflicted Violence detection project's testing and deployment phases are crucial to ensure that the system is efficient, dependable, and ethical in its operation.



```
area_1 = [(0,143),(631,145),(614,160),(0,156)]
area_2 = [(0,143-20),(631,145-20),(614,160+20),(0,156+20)]
area2 = {}
```

Fig. 8. Coordinates of ROI



```
mul = ((85 if movement[track_id][4]>10 else (movement[track_id][4]*85)/10)+
      (85 if movement[track_id][7]>10 else (movement[track_id][7]*85)/10)+
      (40 if((f-movement[track_id][9])/30>300) else ((f-movement[track_id][9])/30)*40/300))
prob = round(mul/255,2)
```

Fig. 9. Non-decreasing Probability calculation

To make sure that the system achieves its intended aims and objectives, careful planning, attention to detail, and coordination among stakeholders are needed.

V. RESULT AND DISCUSSION

A. System Performance Evaluation

At the moment, a frame is started by two models, YOLO object detection and Deepsort object tracking, which requires a lot of computation and causes delays. Further improvements can/will be made in our project to gain better optimisation and produce better results.

B. Limitations and Future Work

One or more of the Self-Inflicted Violence detection project's potential drawbacks could be:

- Limited data: The system's accuracy and dependability may be impacted by the system's restricted data availability.
- Ethics: Using cameras and other surveillance technologies, especially in public places, may raise ethical questions.
- Cost: For some firms, the system's implementation and maintenance costs may be a hurdle.

The following are some prospective areas for future development in Self-Inflicted Violence detection projects:

- Algorithm improvement: In order to decrease false positives and boost accuracy, machine learning algorithms can be enhanced and improved Several data sources can be used to increase accuracy and dependability. Examples of such additional data sources are social media and audio recordings.
- Collaboration with mental health specialists: Partnering with mental health specialists can offer extra perspectives and knowledge to increase the system's efficacy.

VI. CONCLUSION AND FUTURE DIRECTION

A. Summary of Findings

Because it has the potential to save lives and avert tragedies, Self-Inflicted Violence detection is an important area of research. In this project, our goal was to create a model that can correctly predict a person's risk of Self-Inflicted Violence. Based on our results, we can conclude that it is possible to develop a reliable Self-Inflicted Violence detection model using Computer Vision techniques and Machine learning algorithms. However, there are still many challenges in this field, such as the need for more extensive and diverse field.

B. Recommendations for Future Work

Accuracy improvements: Self-Inflicted Violence detection algorithms can benefit from ongoing accuracy improvements. This can be done by incorporating extra features and data sources and by improving algorithms through continuous testing and evaluation. Increase the number of data sources: social media, online discussion boards, and medical records are just a few of the sources that can help algorithms that detect Self-Inflicted Violence. Future research may examine the use of additional data sources to increase precision and recognize people who are at risk.[4] Create interventions: Effective interventions, such as crisis counseling and mental health resources, should be used in conjunction with Self-Inflicted Violence detection algorithms. The development and evaluation of interventions that can be delivered via digital platforms may be the main focus of future work.

REFERENCES

[1] https://www.researchgate.net/publication/358610395$_{R}eal$ – $Time_{H}uman_{P}ose_{D}etection_{a}nd_{R}ecognition_{U}sing_{M}ediaPipe$
[2] https://ieeexplore.ieee.org/document/8899360
[3] https://www.cityofpasadena.net/public-works/wp-content/uploads/sites/29/2017-07-19-Public-Safety-Committee-Suicide-Mitigation-Proposals.pdf
[4] https://ieeexplore.ieee.org/document/8310087
[5] https://www.mdpi.com/2075-5309/11/9/409

VII. OUTPUTS OF PROJECT

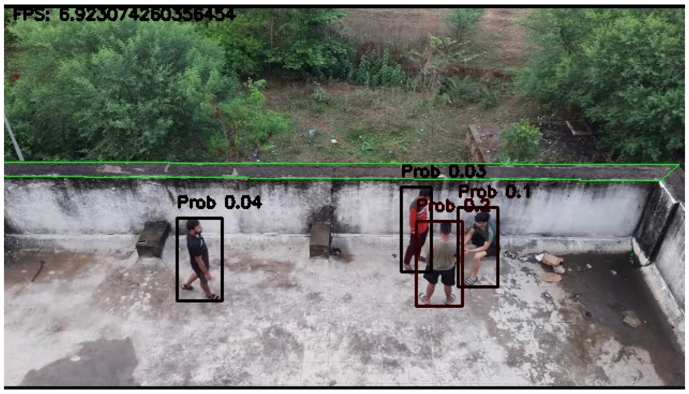


Fig. 10.

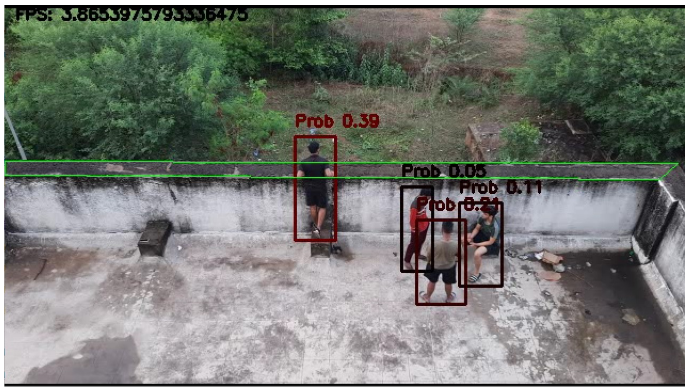
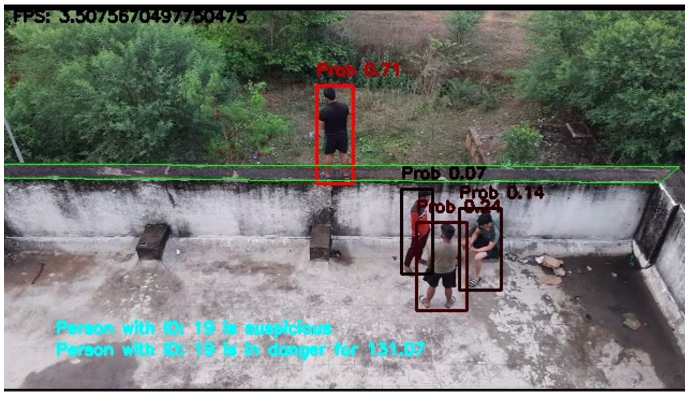
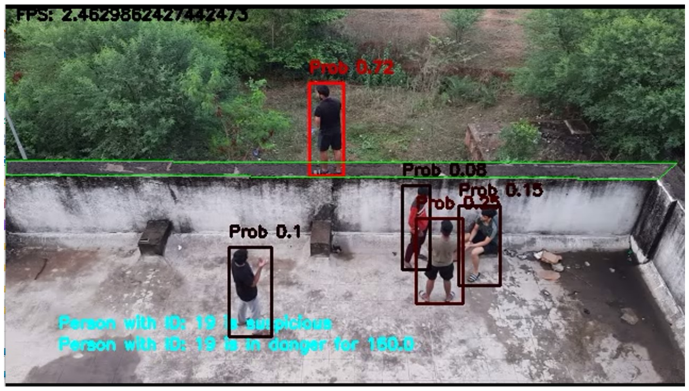


Fig. 11.



```
Person with ID: 19 is suspicious
Person with ID: 19 is in danger for 131.07
```

Fig. 12.



```
Person with ID: 19 is suspicious
Person with ID: 19 is in danger for 150.0
```

Fig. 13.