IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

AIR CANVAS USING OPENCV AND YOLOv5 ALGORITHM

[1]CHELLURI SAI KUSUMA KEERTHI, [2]Y. SOWMYA, [3]L. SATWIKA, [4]K. SIREESHA

[1]Student, [2]Assistant Professor, [3]Student, [4]Student
[1]Computer Science Department,
[1]GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN, VISAKHAPATNAM, INDIA

Abstract: Despite the latest advancements in the domain of image processing and deep learning, the combination of object detection and monitoring mechanisms for drawing and visible artwork remained underexplored. One such captivating study region is writing in the air. The undertaking takes gain of this concept and specializes in developing a motion-to-textual content converter that could function software program for intelligent wearable gadgets for writing from the air. This project is a reporter of random gestures, a real-time video-based pointing approach that allows for the sketching and writing of English textual content in front of a camera over the air. The YOLO algorithm will be used to detect and track the movements of the finger. The generated textual content may be used for several purposes, which include sending messages, emails, etc. It could be an effective way of communique for the deaf and also can be an excellent motivation for the dustless lecture rooms for the scholars to study in. It is a powerful communique technique that reduces human-system interplay, putting off the need to write.

Index Terms **- image processing, deep learning, YOLO**

INTRODUCTION

How easy would our lives be if all we have to do is just wave our hands in the air to write the text? Minimal effort and maximum work done. Isn't that the kind of comfort we all seek in our tasks? This could make important development in the way classes are taught, how the deaf communicates with other people, how it minimizes the human-system interaction, and many more. Air Canvas is an application that traces the movements of the desired object and projects them on an interface as the textual content we want to display or convey. This is done using object tracking. It is built using the OpenCV module in Python programming language along with a real-time object detecting algorithm. The task of detecting targets is vital in this project. With the increase in GPU computing power and neural networks research in recent years, it has become a hotspot in global research. There are two main methods for target detection. One is the traditional method using HOG + SVM. The other is to use Deep learning algorithms. Right from R-CNN, the performance is getting better and better with more new emerging algorithms. Since the simulated result is to be used on the real-time object, the application parameters and network structure were adjusted based on YOLOv5s. YOLOV5 is built with Pytorch Classifier in deep learning, and after object detection, the OpenCV module is used to feed the algorithm real-time or file format video input, as well as track the item recognized in the output, making the system efficient.

RELATED WORK

2.1 Using LSTM and CNN

Md. Shahinur Alamat al. 2020[4] implemented a trajectory-based writing system using three-dimensional trajectories collected by a depth camera that tracks the fingertip. They employed LSTM and CNN as a recognizer. They also put the model to the test with a 6D motion gesture alphanumeric character dataset, which it passed with 99.32% accuracy, which is the highest to date. Four sections make up the full procedure. The four sections include fingertip detection, data gathering, normalization, and network creation. The experiment was carried out with the help of an IntelSense SR300 camera and a computer. C# and Python programming languages were used for interfacing and these networks were implemented in Keras high-level API for the Tensorflow backend. However, each character's ending frame was different due to the different writing patterns.

2.2 Using OCR

Pavithra Ramasamy et al. 2016[3] proposed a finger motion tracking system that identifies the English character written by the finger over the air. The web camera is used to capture the LED-fitted finger movements. From this, the movement is tracked by only using the LED. The entire video is divided into a sequence of frames. The tracking color is white, with a black backdrop.

To create a single black and white frame, join the following black and white frames together. The image is then cropped to fit the ROI. The character is identified and displayed on the screen using optical character recognition (OCR). The image which has the maximum correlation with the alphabet image is shown as the output. The alphabets A, C, D, J, K, L, O, P, R, S, T, U, V, and Z are more accurately identified than the alphabets E, F, G, H, I, M, N, Y. Few alphabets like B, Q, W, X tend to have the least accuracy.

2.3 Using SSD and Faster-RCNN

Nishta Dua et al. 2021[2] developed a project, a motion-text-converter. They proposed a system where the number of gestures used for controlling the system is equal to the number of functionalities involved. The basic functionalities included are Writing mode, Color mode, and Backspace. The data input is the video clip of the hand movement. Two-second videos of the hand are captured in different environments. These videos are then broken into 30 separate images. In total there were about 2000 images. The dataset is divided into train and dev sets (85%-15%). They used SSD and Faster-RCNN pre-trained models. Faster RCNN yielded much better results.

This model has shown an accuracy of 99%. Since the generated images were from the same video, the model couldn't work for discrete backgrounds.

METHODOLOGY OF YOLOV5

Object detection is a method of detecting objects from a video. One of the most adaptable and well-known object detection models is YOLO, which stands for "You Only Look Once." The YOLOv5 made the transition from the darknet to PyTorch, achieving 140 frames per second in the Tesla P100, compared to 50 frames per second in the YOLOv4. YOLOV5 offers the same benefits as YOLOV4 and has a nearly identical architecture. In comparison to YOLOv4, YOLOv5 makes it easier to learn and recognize items. YOLO algorithm divides the given input image into the SXS grid system. Each grid is responsible for object detection where the grid cells predict the boundary boxes for the detected object. For every box, there are five main attributes: x and y for coordinates, w and h for width and height of the object, and a confidence score for the probability that the box contains the object. To predict the boxes, the YOLO algorithm predicts bounding boxes as deviations from a list of anchor box dimensions.

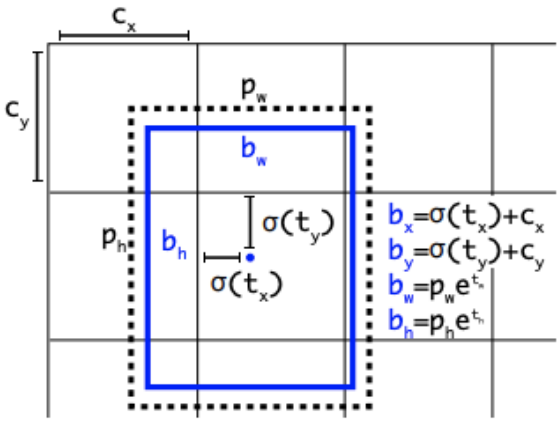


Fig.1. Boundary boxes of the detected object

For the model, YOLOv5 offers four distinct scales: S, M, L, and X, which stand for Small, Medium, XLarge, and Large, respectively. Each of these scales applies a different multiplier to the depth and width of the model, which means the overall structure of the model remains constant, but the size and complexity of each model are scaled.

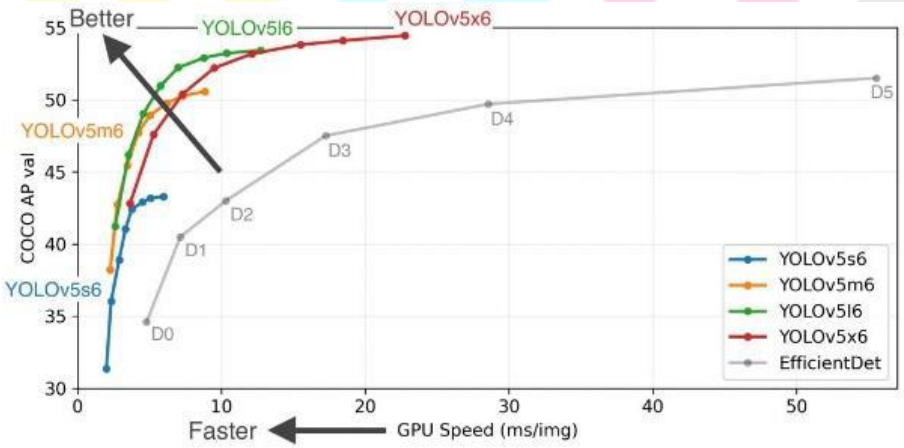


Fig.2. Performance of different scales of YOLO

It consists of three parts: (1) Backbone: CSPDarknet, (2) Neck: PANet, and (3) Head: YOLO Layer. The data is provided through CSPDarknet for feature extraction before being loaded into PANet for feature fusion. Finally, the detection results are output by YOLO Layer (class, score, location, size).

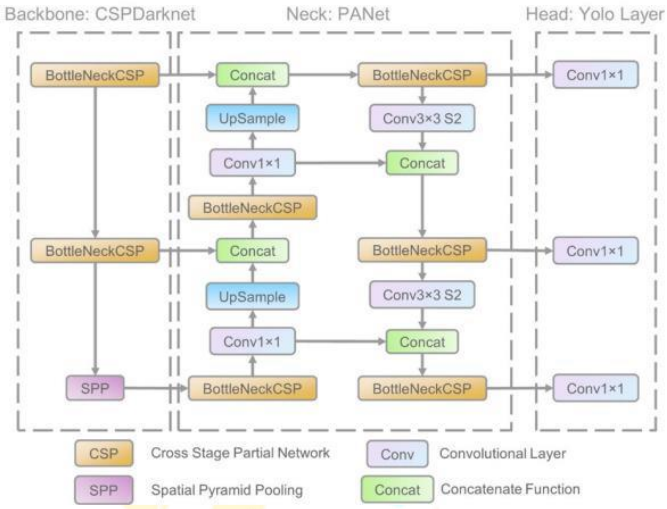


Fig.3. Architecture of YOLOv5

In the current work, YOLOv5 "s" is used. In the "s" variant, Model Backbone is primarily used to extract important features from the given input image. The CSP — Cross Stage Partial Networks — backbone is utilized in YOLOv5 to extract rich informational characteristics from the input image. With deeper networks, CSPNet has shown a significant improvement in processing time. Model Neck is mostly employed in the creation of feature pyramids. Feature pyramids aid in the generalization of models across object scales. It aids in the detection of the same object in various sizes and scales. Feature pyramids are quite beneficial in helping models operate successfully while dealing with overlooked data. Other models employ feature pyramid approaches like FPN, BiFPN, PANet, and others. PANet is used as a neck in YOLOv5 to generate feature pyramids.
The final detection is mostly performed by the model head. It creates final output vectors with class probabilities, bounding boxes, and objectness scores by applying anchor boxes to features.

3.1 Activation Function

In any deep neural network, the choice of activation functions is critical. Recently lots of activation functions have been introduced like Leaky ReLU, mish, swish, etc. YOLOv5 goes with the Leaky ReLU and Sigmoid activation function.

3.2 Optimization Function

For the optimization function in YOLOv5, we have two options

1. SGD
2. Adam

In YOLOv5, the default optimization function for training is SGD. However, you can change it to Adam by using the " --adam" command-line argument.

3.3 Cost Functions or Loss Functions

A compound loss is calculated in the YOLO family based on the objectness score, class probability score, and bounding box regression score. For the loss computation of class probability and object score, Ultralytics employed PyTorch's Binary Cross-Entropy with Logits Loss function. We also have the option of utilizing the Focal Loss tool to calculate the loss. Using the fl_gamma hyper-parameter, you can select to train with Focal Loss.

Table.1. Functional metrics of YOLOv5

Model	AP^{val}	AP^{test}	AP_{50}	$Speed_{GPU}$	FPS_{GPU}	params	FLOPS
YOLOv5s	36.6	36.6	55.8	**2.1ms**	**476**	7.5M	13.2B
YOLOv5m	43.4	43.4	62.4	3.0ms	333	21.8M	39.4B
YOLOv5l	46.6	46.7	65.4	3.9ms	256	47.8M	88.1B
YOLOv5x	**48.4**	**48.4**	**66.9**	6.1ms	164	89.0M	166.4B

3.4 Model Summary

Model Summary: 191 layers, 7.46816e+06 parameters, 7.46816e+06 gradients

IV. IMPLEMENTATION

The system begins with the home page of the project. After optioning air canvas, the live camera feed is given as the input to the object detection system. For detection, the input data is divided into frames. The YOLOV5 uses the Model or dataset to detect the item in the input data during detection. After identifying objects, the model classifies and represents the observed items by utilizing labels and bounding boxes around the identified objects, and then it is processed into output.

4.1 Input

Using OpenCV, we can access the camera module and add video files in various formats. OpenCV is used to obtain the real-time video frame from camera lenses. Opencv is an open-source library that includes image processing, camera access, and other computer vision components. GPU support has now been added to the Opencv module, which is a critical component of PyTorch. Opencv technology has progressed quickly and now supports a wide range of algorithms, particularly in the field of image processing. Numpy, matplot, use, and other Python libraries are supported by OpenCV.Neural Network.

The input data is categorized into frames using the camera or video input, and each frame is transmitted to the YOLO detection algorithm with the model specified by the user. The object is bound with boxes and labels and sent to the output phase, where the detected frames are collected and compressed into the output format, once the detection is complete. The detected frames are utilized for tracking with OpenCV before merging.

4.2 Dataset

This field is used to create a custom dataset from raw images, which may then be used to create a custom model for detection. The first stage is to build a dataset from raw images obtained from various sources. In this scenario, the images depict a person's hand carrying a blue-colored object. The objects from the dataset images must next be annotated and labeled. Annotating and labeling the objects is done using Python frameworks like "LabelImg." Each object we want the detector to see is enclosed in a box labeled with the object class that we want the detector to anticipate. Each bounding box has five predictions: x, y, w, h, and confidence.

Fig.4. Hand holding a blue object

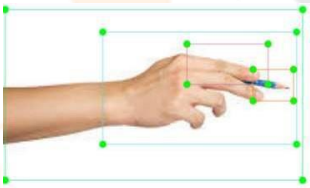


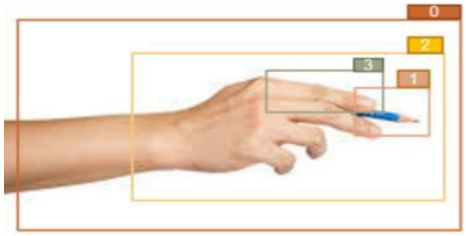
Fig.5. bounding boxes of the class labels in the image.

```
3 0.370909 0.489071 0.720000 0.628415
2 0.478182 0.464481 0.469091 0.404372
0 0.545455 0.377049 0.196364 0.142077
1 0.656364 0.450820 0.098182 0.114754
```

Fig.6. Text file containing the coordinates of the bounding boxes of the class labels.

After the dataset has been annotated and labeled, it is divided into train and test images in a percentage of 80% for train and 20% for the test, which is the usual optimal percentage for training. After that, it will be sent to the YOLO training method, which will train the dataset and create the model.

x, y, w, h, and confidence are the five predictions in each bounding box. The (x, y) coordinates represent the box's centroid in relation to the grid cell's limits. The width and height are calculated in terms of the entire image. Finally, the IOU between the projected box and any ground truth box is represented by the confidence prediction. Each grid cell additionally estimates Pr(Classi



|Object), which are C conditional class probabilities. These probabilities are based on the grid cell in which an object is located. Regardless of the number of boxes B, we only anticipate one set of class probabilities per grid cell.

Fig.7. Predicting the class labels after training.

The proposed methodology's architecture can be seen in the diagram below. This project combines OpenCV tasks with an object detection method. OpenCV is used to collect live video in order to detect the item and provide an interface that allows the user to paint the required patterns. Initially, the trackbar for changing the color's chromatic values is set up with options like hue and saturation. To display the output, a plain white paint interface is constructed. The next stage is to use the web camera for input. In the form of frames, the feed is passed on to the next phase. The standard frame rate for OpenCV is 30 frames per second. However, with the YOLO algorithm, it is enhanced to 45 frames per second. The frames are then subjected to morphological procedures such as dilation and erosion. These methods hide the contents of the frames, leaving only the principal item to be detected viewable.

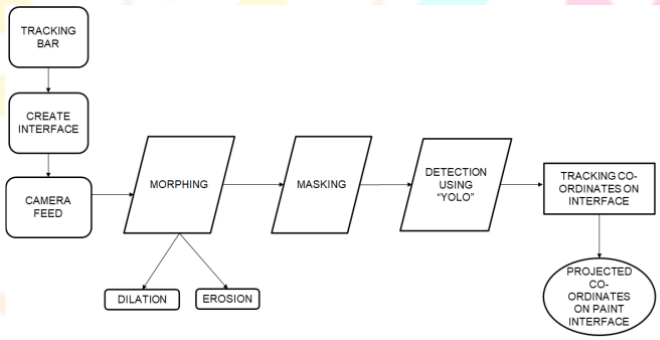


Fig.8. Content diagram of the implementation

4.3 Dilation

For binary pictures, dilation is defined mathematically as:

• Assume X is the set of Euclidean coordinates for the input binary image and K is the set of coordinates for the structuring element.

• Let Kx stand for K's translation with its origin at x.

• The dilation of X by K is the set of all locations x such that Kx traverses X in a non-empty way.

$I \oplus H \equiv \{(p+q) \mid \text{for every } p \in I, q \in H \quad (1)$

$(I \oplus H)(u,v) = \max_{(i,j) \in H} \{ I(u+i, v+j) + H(I,j)\} \quad (2)$

4.4 Erosion

For binary pictures, the mathematical definition of erosion is:

• Assume that X is the set of Euclidean coordinates corresponding to the input binary image and K is the set of coordinates for the structuring element.

• Let Kx be the translation of K with x as the origin.

• The set of all points x such that Kx is a subset of X is therefore referred to as the erosion of X by K.

$I \ominus H \equiv \{p \in Z_2 \mid (p+q) \in I, \text{for every } q \in H\} \quad (3)$

$(I \ominus H)(u,v) = \min_{(i,j) \in H} \{ I(u+i, v+j) - H(I,j)\} \quad (4)$

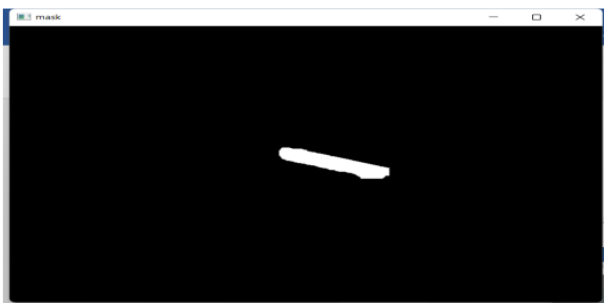


Fig.9. Camera contents after masking.

4.5 Weights and Models

A ".yaml" extension format file must be used to configure the framework's labeled dataset, which can then be used to append the text label to the algorithm. Once the YAML file is configured, PyTorch is used to train the given dataset using GPU according to the epochs specified in the algorithm for training, and with the test dataset, the trained model is tested after completion, predicting the objects in the test image. Once the objects are predicted the model is compressed into the YOLO model format which is configured using the pre-trained model that is using the dataset model. After that, the model file, along with the associated test result plotted in graphs and texts, is saved in the output folder, where the model can be evaluated and tested by employing it in a detection method.

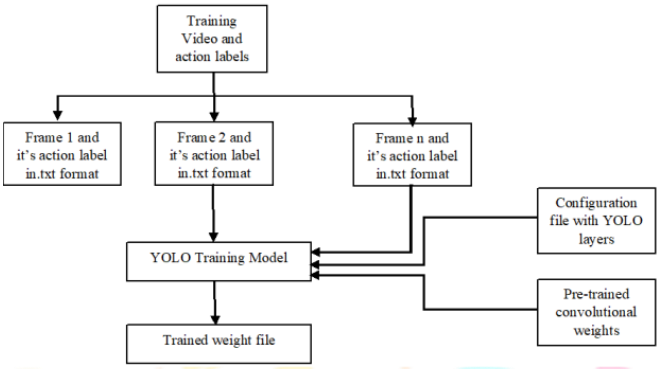


Fig.10. Content diagram of the YOLOv5 methodology

4.6 Results and Performance Analysis

The camera is accessed using the OpenCV library. The model is applied to each individual frame obtained from the live feed. The movements of the object are simultaneously tracked and projected onto the paint interface.

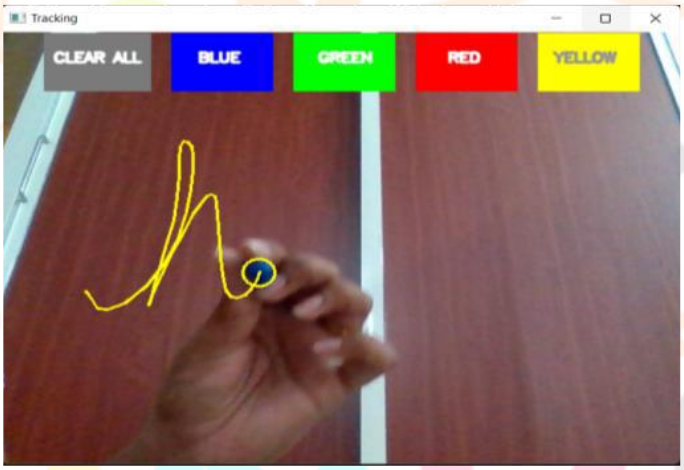


Fig.11 Tracking of the blue object

Fig.12. Traced output on the paint interface.

mAP, Precision, and Recall are the three terms used to evaluate YOLO's performance. The mAP is a metric that combines recall and precision to detect object correctness. It is derived using the average precision value for recall values ranging from 0 to 1 and IOU (intersection over union) values ranging from 0.5 to 0.95.

$$AP=\Sigma PrRr=1R \quad (5)$$

$$mAP=1N\Sigma APk \quad (6)$$

Precision refers to the accuracy with which objects are predicted. Precision refers to how well the model predicts the positive class.

$$\text{Precision}= TP \,/\, (\, TP + FP \,) \quad (7)$$

The recall determines how well the objects or classes are recognized. In other words, recall or sensitivity determines the percentage of true positives that are accurately identified.

$$\text{Recall} = TP/\, (\, TP + FN \,) \quad (8)$$

Intersection Over Union is the term for IoU. IoU is used to determine the distance between two picture boundaries in order to see if they have overlapped and, if so, at what pace. In order to determine if the detection was truly positive or false positive, we will set an IoU threshold (say, 0.5 in our context).

$$\text{IoU} = Area\ of\ Overlap \,/\, Area\ of\ Union \quad (9)$$

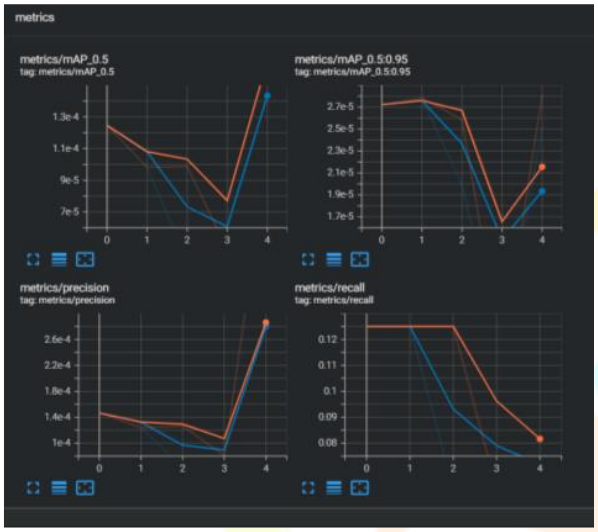


Fig.13. Performance graphs of the model

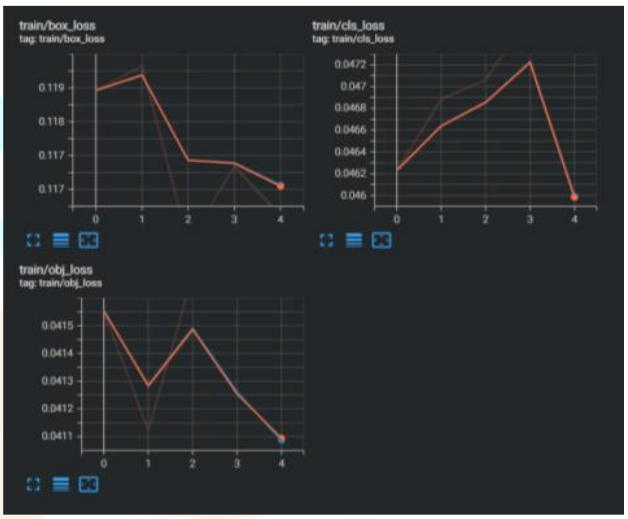


Fig.14. Performance graphs of the model (contd.)

REFERENCES

[1] Baig, Fasial& Khan, Muhammad & Beg, Saira. (2013). Text writing in the air. Journal of Information Display. 14. 10.1080/15980316.2013.860928.

[2] Saoji, Saurabh &Dua, & Vidyapeeth, Bharati & Choudhary, Akash & Phogat, Bharat. (2021). AIR CANVAS APPLICATION USING OPENCV AND NUMPY IN PYTHON. International Journal of Research in Engineering and Technology. 8. 2395-0056.

[3] P. Ramasamy, G. Prabhu and R. Srinivasan, "An economical air writing system converting finger movements to text using web camera," 2016 International Conference on Recent Trends in Information Technology (ICRTIT), 2016, pp. 1-6, doi: 10.1109/ICRTIT.2016.7569563.

[4] M. S. Alam, K. -C. Kwon and N. Kim, "Trajectory-Based Air-Writing Character Recognition Using Convolutional Neural Network," 2019 4th International Conference on Control, Robotics and Cybernetics (CRC), 2019, pp. 86-90, doi: 10.1109/CRC.2019.00026.

[5] S. R. Balaji and S. Karthikeyan, "A survey on moving object tracking using image processing," 2017 11th International Conference on Intelligent Systems and Control (ISCO), 2017, pp. 469-474, doi: 10.1109/ISCO.2017.7856037.

[6] T. -H. Tsai and J. -W. Hsieh, "Air-writing recognition using reverse time ordered stroke context," 2017 IEEE International Conference on Image Processing (ICIP), 2017, pp. 4137-4141, doi: 10.1109/ICIP.2017.8297061.

[7] Handalage, Upulie & Kuganandamurthy, Lakshini. (2021). Real-Time Object Detection Using YOLO: A Review. 10.13140/RG.2.2.24367.66723.

[8] Gupta, Akshara & Verma, Aditya & Yadav, Aditya & M, Arvidhan. (2021). YOLO OBJECT DETECTION USING OPENCV. International Journal of Engineering Applied Sciences and Technology. 5. 10.33564/IJEAST.2021.v05i10.036.

[9] Kurdthongmee, Wattanapong. (2020). Accelerate the Detection Frame Rate of YOLO Object Detection Algorithm. 10.1007/978-3-030-19861-9_14.

[10] Wu, Jian-Da & Chen, Bo-Yuan & Shyr, Wen-Jye & Shih, Fan-Yu. (2021). Vehicle Classification and Counting System Using YOLO Object Detection Technology. Traitement du Signal. 38. 1087- 1093. 10.18280/ts.380419.

[11] Balamurugan, Sudhir & Santhanam, Sanjay & Billa, Anudeep & Aggarwal, Rahul & Alluri, Nayan. (2021). Model Proposal for a Yolo Objection Detection Algorithm-based Social Distancing Detection System. 1-4. 10.1109/ICCICA52458.2021.9697212.

[12] Ejiyi, Chukwuebuka & Bamisile, Olusola & Nneji, Grace & Zhen, Qin & Ilakoze, Ndalahwa & Chikwendu, Ijeoma. (2021). Systematic Advancement of Yolo Object Detector For Real-Time Detection of Objects. 279-284. 10.1109/ICCWAMTIP53232.2021.9674163.

[13] N. Dardagan, A. Brđanin, D. Džigal and A. Akagic, "Multiple Object Trackers in OpenCV: A Benchmark," 2021 IEEE 30th International Symposium on Industrial Electronics (ISIE), 2021, pp. 1-6, doi: 10.1109/ISIE45552.2021.9576367.

[14] A. Brdjanin, N. Dardagan, D. Dzigal and A. Akagic, "Single Object Trackers in OpenCV: A Benchmark," 2020 International Conference on INnovations in Intelligent SysTems and Applications (INISTA), 2020, pp. 1-6, doi: 10.1109/INISTA49547.2020.9194647.

[15] Q. Cai, S. Zhong and K. Chen, "Target Detection in Rural River Image Based on Yolo V5 Model with the Cross-Stitch Unit," 2021 6th International Conference on Image, Vision and Computing (ICIVC), 2021, pp. 36-42, doi: 10.1109/ICIVC52351.2021.9527005.

[16] M. Karthi, V. Muthulakshmi, R. Priscilla, P. Praveen and K. Vanisri, "Evolution of YOLO-V5 Algorithm for Object Detection: Automated Detection of Library Books and Performace validation of Dataset," 2021 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES), 2021, pp. 1-6, doi: 10.1109/ICSES52305.2021.9633834.

[17] I. I. Lychkov, A. N. Alfimtsev and S. A. Sakulin, "Tracking of Moving Objects With Regeneration of Object Feature Points," 2018 Global Smart Industry Conference (GloSIC), 2018, pp. 1-6, doi: 10.1109/GloSIC.2018.8570061.

[18] H. Kim and K. Sim, "A particular object tracking in an environment of multiple moving objects," ICCAS 2010, 2010, pp. 1053-1056, doi: 10.1109/ICCAS.2010.5669674.

[19] Q. Guo and B. Qiao, "Research on the detection and tracking technology of moving object in video images," 2015 IEEE International Conference on Mechatronics and Automation (ICMA), 2015, pp. 469-473, doi: 10.1109/ICMA.2015.7237531.

[20] F. Yan and Y. Xu, "Improved Target Detection Algorithm Based on YOLO," 2021 4th International Conference on Robotics, Control and Automation Engineering (RCAE), 2021, pp. 21-25, doi: 10.1109/RCAE53607.2021.9638930