Object Detection Using OpenCV: A Comparative Study of Deep Learning and Traditional Methods

Kartikey Kumar Pal[1], Surya Kant Pal[1]*, Saloni Srivastava[1], Hari Shankar Shyam[2], Sachin Singh[1]

[1]Department of Mathematics & Data Science, SSES, Sharda University, Greater Noida-201310, UP, India

[2]Sharda School of Business Studies, Sharda University, Greater Noida, Uttar Pradesh, India

(e-mail Id: palkartikey188@gmail.com; suryakantpal6676@gmail.com; salonisrivastava007@gmail.com; harishankar.shyam@sharda.ac.in; singhat619@gmail.com)

*Corresponding Author: suryakantpal6676@gmail.com

Abstract

This study investigates the usage of OpenCV, an Open-Source computer vision toolkit, in Python for object detection. In this paper we use a deep learning-based strategies like YOLO (You Only Look Once) and SSD (Single Shot MultiBox Detector) as well as more classical models like Haar cascades. The effectiveness, precision and real-time performance is assessed on various approaches using benchmark datasets. The results of experiments are how well OpenCV's deep learning integration and inbuilt feature works for obtaining reliable object detection. The methods for current and future real time object identification systems are explored in this study with the goal of suggesting improvements to the development of real time object identification systems.

Keywords: OpenCV, YOLO, SSD, Deep-learning, Haar Cascade

1 Introduction

OpenCV (Open-Source Computer Vision Library) is a very useful open-source tool for real time image processing and analysis. OpenCV has a wide scope of tools for various computer vision tasks like object detection, face recognition, motion tracking, image enhancement, etc. A speed or efficiency characteristic common to robotics, AI, medical imaging, and surveillance, it has become popular.

R. Nagariya et al. (eds.), *Proceedings of the International Conference on Sustainable Business Practices and Innovative Models (ICSBPIM-2025)*, Advances in Economics, Business and Management Research 349,
https://doi.org/10.2991/978-94-6463-872-1_62

OpenCV was developed or rather worked as a research project that Intel worked on for pushing the limits of computer vision back in 1999 and it became open source in 2000. Since then, it has been improved even more by an ever-growing global developer and researcher community. Today OpenCV is open source and is maintained by the non-for-profit organization — anyone interested in computer vision will look to this for resources today. It works with multiple programming languages, like Python, C++, Java and so on. So, it is acceptable to any sort of developer be it at the beginner level or a professional level graduating from college.

The practice of object detection has now become one of the corner stones of modern computer vison with applicability in the domains of autonomous driving, surveillance systems, healthcare diagnostics, robotics, etc. A high accuracy and efficiency object identification capability is required for these applications to work at all. Although there has been progress in object detection, this problem is difficult because of appearance, scale, and environment of the object variations. An object detection system can be powered by using the tools and algorithms they provide from an open-source computer vision and machine learning library, open CV, that provides a vital toolkit of tools and algorithms they use to create and run powerful variants of object detection systems.

Ultimately, our goal is to understand and assess the effectiveness of different object detection schemes that are written using OpenCV. The objective of this study is to evaluate the efficiency of these techniques in what regards accuracy, speed, and computational efficiency. We attempt to exploit pre trained model and custom trained classes so that we can improve object detection capability and provide real world implementation guidelines.

The work described here implements and compares three prominent object detection techniques which are Haar Cascades, YOLO (You Only Look Once), and SSD (Single Shot MultiBox Detector). Preprocessing steps, model selection, training processes, and performance test on datasets are also included in the study's scope. There exists a set of experiments for both to take place within the controlled environment at the same time. There is a comparison of results in respect of strengths and weaknesses. When those involved in the development of object detection systems based upon OpenCV see the results of this research, they will learn a thing or two. YOLO is an object detection algorithm with high speed and great accuracy. Shortness also making it efficient in

operation because unlike some traditional methods which run over image many times in order to achieve satisfactory detection, YOLO applies a novel approach--at one time all entire images are processed together through convolutional neural networks.

This makes it incredibly fast and ideal for real-time applications like autonomous vehicles, surveillance, and robotics.

Imagine: YOLO is a reckless and bold algorithm that cuts images into grids, like a super-efficient pizza chef. Then, it throws a series of bounding boxes - imagine them as vibrant little boxes bouncing around - and throws class probabilities onto any object that dares to reside within each small cell. This chaotic yet brilliant method allows it to discover multiple objects in an instant while maintaining astonishing processing speed. However, this speed demon occasionally trips - it may be a bit clumsy when dealing with small or overlapping objects, unlike more cautious two-stage detectors like the meticulously precise Faster R-CNN.

Over the years, YOLO has undergone a dazzling transformation, evolving into increasingly complex versions - YOLOv3, YOLOv4, YOLOv5, and the latest, gorgeous YOLOv8, with higher precision and astonishing efficiency in each iteration. Its adaptability consolidates its position as the preferred tool in many fields, from the complex world of medical imaging to the vigilant eyes of security systems and the busy retail world.

Now, let's take a look at SSD, another deep learning miracle, a fast object detection algorithm built for real-time applications. Unlike more organized, region-based detectors like Faster R-CNN (which carefully identifies potential object regions before classification), SSD is a streamlined and powerful engine that quickly completes object localization and classification in a stunning neural network pass. This makes it incredibly fast and maintains astonishing high precision.

SSD is very similar to YOLO in that it partitions images into grids, but it directly predicts bounding boxes from multiple feature maps - treating these maps as insightful blueprints. Its key difference lies in the use of multiple convolutional layers, like having multiple well-trained eyes, with each layer focused on detecting objects of different sizes. This makes it a master at discovering the smallest details. However, in scenes filled with overlapping objects, it may not be as accurate as those two-stage perfectionists.

Due to the smart balance between its amazing speed and impressive accuracy, SSD is a popular choice in the key world of real-time video surveillance, autonomous vehicle, and the evolving field of mobile based object detection. Its efficiency makes it a top competitor for tasks that require lightning-fast processing without affecting too much accuracy.

2 Literature review

Jalled et al. [1] presented an approached to enhance Unmanned Aerial Vehicles (UAVs) with object and face detection capabilities using the Haar Cascade algorithm implemented through OpenCV-Python. The primary goal is to mitigate the risk of UAV collisions due to undetected objects and to improve surveillance functions by accurately detecting and tracking human subjects. To minimize these problems, the Viola-Jones algorithm, as well as cascade object detector function and vision training function are used in the study in order to improve the capability of human detection and tracking. The advantage of their approach is that it requires reduced processing time, which makes it applicable to real time application. The developed Python code was validated by testing against existing video and image databases over existing object detection and surveillance conducted by UAV and the results were shown to be effective.

Mehtab et al. [2] explained some of the ways of object detecting and tracking in video streams using OpenCV and Python. In this regard it discusses multiple approaches such as frame differencing, where moving regions are identified through compute of difference between consecutive frames, and Color Space Transformation which is used to improve the accuracy of the object detection. Another technique of motion detection is also implemented to separate foreground objects from a static background. The second topic of study is optical flow and the way that are analysing object motion by changes in pixel intensity and Haar cascade classifiers, which use previously trained models to locate objects on the scene, like faces, eyes, and noses. Furthermore, paper includes Canny Edge Detection algorithm for the object edge in image to enhance the detection accuracy. The research uses these methodologies to provide some insights on the implementation of object detection and tracking using OpenCV, which can be implemented in real time.

Myori et al. [3] focused on practical methods for implementing the real-time object detection through live webcam feeds using Python. The study aligns with the growing trend of utilizing lightweight, accessible technologies for real-time computer vision tasks in everyday applications.

Sharma et al. [4] used Open CV to complete those type of detection tasks. In this project the authors explore the ways in which artificial intelligence (AI), machine learning (ML) techniques facilitate object recognition and discuss why OpenCV is a valuable tool for real time object detection and tracking system beginners. For car safety applications, it is essential that tracking systems are also adaptable to moving cameras, and this is demonstrated in the project.

The contribution of the study emphasizes how much AI computation could help improve object recognition and how can we mix global positioning systems when connected to moving cameras. For applications such as vehicle safety, where real-time object detection and tracking are critical, this is particularly relevant as it is an adaptability, and real time is of importance. Finally, the project is concluded as a convenient reference to those who are interested in computer vision and object detection and can be used by them to incorporate OpenCV and Python to develop an efficient and flexible object detection system.

Sharma et al. [5] reflected a border trend in AI research where the convergence of powerful neural network models with an efficient processing frameworks like OpenCV is driving rapid progress in areas such as object detection, facial recognition, and real time video analysis.

Rakesh et al. [6] explored efficient strategies for real-time object recognition using classical image processing techniques, with a strong emphasis on practical, real-world implementations.

Hasan et al. [7] offered the detail exploration of how classical computer vision techniques can be effectively used for face detection and recognition with OpenCV. It is framed within the border context of the rising demand for automated biometric systems in areas such as security, authentication, and human computer interaction.

Chandan et al. [8] explored the integration of deep learning models with OpenCV for creating efficient, real-time object detection and tracking systems. Their work reflects an important step in the evolution of computer vision, merging classical computer vision tools with modern neural network-based models to achieve both speed and accuracy.

Howse et al. [9] provided an extensive and project-driven exploration of computer vision techniques using OpenCV and Python. The book is structured around practical, hands-on projects that guide readers through the implementation of real-world applications, from basic image processing to complex vision-based systems.

3 Methodology

This survey uses side-by-side comparison to evaluate OpenCV's object localization, comparing cutting-edge deep learning with old-fashioned computer vision methods. We are using a standardized benchmark dataset - COCO - to compete fairly. This dataset is a true hodgepodge of images, each carefully labelled with its inhabitants. For training and testing, we'll choose a representative slice – a meticulously chosen sample with a variety of item kinds, chameleon-envious backdrops, and illumination conditions ranging from brilliant sunlight to pitch-black darkness. Before diving into the main event, we will start by getting our images ready. That means resizing them to ensure consistency, normalising pixel values so everything is on the same scale, and giving them a creative boost through data augmentation - think rotating the images or adjusting the contrast for fun, almost like giving them a playful transformation. Finally, we will format the annotations so they smoothly integrate with popular tools like OpenCV and the deep learning powerhouses TensorFlow and PyTorch

This study explores two major approaches to object detection.

The first is traditional computer vision, using tools like OpenCV. These methods include edge detection with reliable Canny filters, Contour-based object search, surprisingly effective template matching, and the ever-trusted Haar cascade classifiers, All of these rely on carefully handcrafted features.

Then came the deep learning revolution, where intelligent neural networks took over the tasks of feature extraction and classification. In this study, we compare leading models in this space:

YOLO – known for its lightning-fast real-time detection,

Faster R-CNN - a refined two-staged model that excels in precision,

SSD – offering an excellent trade-off between speed and accuracy in a single shot. The models are implemented using OpenCV, TensorFlow, PyTorch and scikit-learn. Deep Learning methods run on GPU-enabled systems, while traditional techniques use CPUs. Key hyperparameters like learning rate, batch size and number of epochs are optimized during training..

Performance evaluation employs accuracy, recall, precision, and F1-score to evaluate detection reliability. Additionally, inference time and computational complexity are analysed to compare efficiency and resource consumption. The experimental procedure involves training and testing deep learning models, implementing traditional object detection techniques, evaluating performance metrics, and analysing trade-offs between accuracy, speed, and computational demands. In order to guide future research paths, the structured methodology offers a thorough comparison of object identification techniques using OpenCV, highlighting each technique's unique benefits and drawbacks.

Dataset

A large-scale dataset for object detection, segmentation, and image captioning is used known as Common Objects in Context (COCO). It sets a standard for computer vision research with its 330,000+ images, 200,000 labelled images, and 80 object categories, which makes COCO a benchmark for computer vision research. Dataset's bounding boxes, segmentation masks, and captions makes the models to understand contextual associations between items.

4 Results and Outcomes

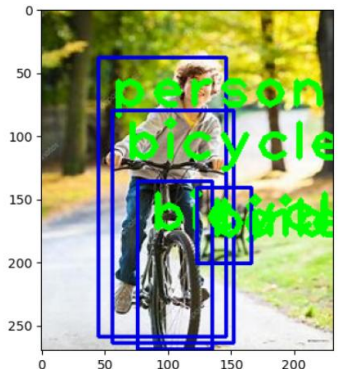
The findings from study's conclusions demonstrate that deep learning- based object detention models outperform conventional methods in terms of performance and efficiency. However, the Haar Cascade classifier performed well on simple objects, but it performed poorly when trying to detect complicated objects surrounded by complicated

backdrops or different lighting situations. YOLO and SSD both provided superior performance, with SSD providing better performance but relying on real-time object identification to try and utilize its high processing speed, which in turn allows YOLO to outperform SSD in real time object identification. SSD was an embedded systems and resource constrained applications because balanced precision and computing efficiency. The FPS research said that YOLO was the best to get frame rates and was great for real time applications. Even so, neither deep learning model was able to deal with obstructed objects, but occasionally made false positives. In general, OpenCV and Python proved to be useful tools to create object detection, and deep learning-based solutions appeared to be overall better performing and more adaptable.

```
font_scale = 3
font = cv2.FONT_HERSHEY_PLAIN
for ClassInd, conf, boxes in zip(ClassIndex.flatten(), confidece.flatten(), bbox):
    cv2.rectangle(img, boxes, (225,0,0),2)
    cv2.putText(img, classLabels[ClassInd-1], (boxes[0]+10, boxes[1]+40),font, fontScale = font_scale, color=(0,255,0), thickness=3)
```

```
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x2be7968bd90>
```



5 Conclusion

In this study, we researched object recognition in Python using OpenCV, which is effective for recognizing and categorizing things in pictures as well as in the video streams. We went over the numerous object identification strategies from the deep learning system like YOLO and SSD to the conventional one like Haar cascades. Real

time object detection in OpenCV comes with the accuracy and the oriented performance optimization its large library of image processing offers.

Though having some benefits, it does not solve all the problems with occlusion, changes in lighting, and computing requests. The future research can be to combine OpenCV with deep learning frameworks to improve detection accuracy and resilience. Additionally, improvements to models for edge devices can not only improve object detection, but also increase the practical uses of object detection in the industrial automation, security monitoring, and driverless car industries.

Overall, OpenCV gives us a good basis for object detection, and it facilitates object detection research and development for developers and researchers.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

[1] Jalled, Fares, Ilia Voronkov, Object detection using image processing. arXiv preprint arXiv:1611.07791 (2016).

[2] Mehtab, Sidra, Object Detection and Tracking Using OpenCV in Python. March 2020 (2020).

[3] Myori, Dwiprima Elvanny, Object detection with a webcam using the Python programming language. *J. Appl. Eng. Technol. Sci*. 2(2) (2021): 103-111.

[4] Sharma, Ayushi, Jyotsna Pathak, Muskan Prakash, J. N. Singh, Object detection using OpenCV and python. *In: 2021 3rd int. conf. adv. Compu., commun. Con. Netw. (ICAC3N).* 501-505. IEEE, 2021

[5] Sharma, Ujjwal, Tanya Goel, Dr Jagbeer Singh, Real-Time image processing using deep learning with opencv and python. *J. Pharm. Negat Results (*2023): 1905-1908.

[6] Rakesh, Vasa, Prathima Chilukuri, P. Vaishnavi, P. Sreekaran, P. Sujala, D. Rama Krishna Yadav, Real time object recognition using opencv and numpy in python. *In: 2023 Int. Conf. Inno. Data Commun. Technol. Appl. (ICIDCA).* 421-426. IEEE, 2023.

[7] Hasan, Ramadan TH, Amira Bibo Sallow, Face detection and recognition using opencv, *J. of Soft Comput. Data Mining.* 2(2) (2021): 86-97.

[8] Chandan, G., Ayush Jain, and Harsh Jain, Real time object detection and tracking

using Deep Learning and OpenCV. In: *2018 Int. Conf. inventive res. Comput. Appl. (ICIRCA)*. 1305-1308. IEEE, 2018.

[9] Howse, Joseph, Prateek Joshi, Michael Beyeler. Opencv: computer vision projects with python. Packt Publishing Ltd, 2016.