TANGENCE
Number 138, 2026

ISSN: 1710-0305
PISSN: 1189-4563

Real-Time Object Detection for Rover and Drone: A Comparative Study from R-CNN to YOLO

Jitendra P Chaudhari*, Axat V Patel, Riddhi Kansagara , Sarvesh Halvadia
Charusat Space Research and Technology Center
CSPIT, Charotar University of Science and Technology, Changa, Gujarat

ABSTRACT
Real-time object detection is a fundamental requirement for autonomous navigation and surveillance systems. This work presents a comparative study of R-CNN, DCNN, and YOLO-based object detection frameworks implemented on an ADAS-enabled rover and a drone surveillance platform. Performance is evaluated in terms of inference latency, detection stability, and visual clarity under real-time operating conditions. Experimental results demonstrate that YOLO provides superior real-time performance while maintaining reliable detection accuracy, making it highly suitable for time-critical autonomous applications.

KEYWORDS
R-CNN, ADAS, YOLO, D-CNN, Object-detection

1. INTRODUCTION

In recent years, real-time object detection has emerged as a critical component of intelligent systems, particularly in the domains of autonomous vehicles, robotics, aerial surveillance, and smart city infrastructure. Camera-based perception systems play a vital role in enabling machines to interpret their surroundings by identifying objects, estimating their locations, and making informed decisions based on visual input. In applications such as Advanced Driver Assistance Systems (ADAS) and drone surveillance, timely and accurate object detection is essential to ensure safety, situational awareness, and operational reliability.

Traditional computer vision techniques relied heavily on handcrafted features and classical machine learning algorithms. Although these methods were computationally lightweight, they lacked robustness when exposed to complex environments, illumination changes, occlusions, and dynamic scenes. The introduction of deep learning, particularly convolutional neural networks (CNNs), significantly transformed object detection by enabling automatic feature extraction and improved generalization capability. As a result, deep learning–based object detection frameworks have become the dominant approach in modern vision-based systems.

Early deep learning–based object detection methods, such as Region-based Convolutional Neural Networks (R-CNN), demonstrated substantial improvements in detection accuracy and localization performance. These approaches decomposed the detection task into multiple stages, including region proposal generation, feature extraction, and classification. While R-CNN-based methods achieved high accuracy in controlled and static environments, their multi-stage architecture introduced significant computational overhead and inference latency, limiting their applicability for real-time video processing [1],[2],[3].

To address the efficiency limitations of region-based detectors, Deep Convolutional Neural Network (DCNN)-based object detection approaches were introduced. These methods aimed to reduce redundant computations by processing entire image frames and sharing convolutional feature maps across the detection task. DCNN-based models improved inference speed compared to R-CNN; however, aggressive downsampling and pooling operations often resulted in loss of spatial details. This led to blurred detection outputs and reduced localization accuracy, particularly during live video processing and dynamic scene analysis [4], [5].

More recently, single-stage object detection frameworks such as YOLO (You Only Look Once) have gained significant attention due to their ability to perform end-to-end object detection in a single forward pass. By reformulating object detection as a regression problem, YOLO eliminates the need for region proposal stages, thereby achieving substantially lower inference latency. Several studies have demonstrated that YOLO-based models are capable of maintaining high frame rates while delivering reliable detection accuracy, making them well suited for real-time applications [6],[7],[8],[9].

Motivated by these developments, this research focuses on the design and implementation of a real-time object detection system using camera-based perception for two practical platforms: an ADAS-enabled autonomous rover and a drone surveillance system. I conducted a comparative experimental study using R-CNN, DCNN, and YOLO-based deep learning models to evaluate their performance under real-time operating conditions. The evaluation emphasizes critical performance metrics such as inference latency, detection stability, visual clarity, and suitability for continuous live video processing.

The primary objective of this work is to identify an object detection framework that offers an optimal balance between detection accuracy, computational efficiency, and real-time stability. In addition to evaluating algorithmic performance, this research also examines practical deployment challenges, including lighting sensitivity, motion effects, and hardware constraints. By systematically analyzing the strengths and limitations of each approach, this study aims to provide practical insights into selecting an appropriate object detection algorithm for real-world autonomous navigation and aerial surveillance applications.

The contributions of this research include a detailed experimental comparison of three widely used object detection paradigms, real-time implementation on ground and aerial platforms, and an in-depth analysis of performance trade-offs. The findings of this work highlight the importance of singlestage detection frameworks for time-sensitive applications and establish a foundation for future enhancements involving multi-sensor fusion, embedded optimization, and large-scale deployment.

2. OBJECTIVES

- To study and compare R-CNN, DCNN, and YOLO object detection algorithms.
- To implement real-time object detection on an ADAS rover platform.
- To deploy real-time object detection on a drone surveillance system.
- To evaluate performance in terms of latency, detection stability, and clarity.
- To identify a suitable algorithm for real-time autonomous applications.

A. System Specifications :-

TABLE 1 : System specifications for real-time object detection

Parameter	**Specification**
Input	USB Webcam
Detection Algorithms	R-CNN, DCNN, YOLO
Processing Mode	Real-time video stream
Detection Output	Bounding boxes with class labels
Application Domain	Autonomous navigation, surveillance

ISSN: 1710-0305
PISSN: 1189-4563

3. METHODOLOGY

A. System Architecture:-

The proposed real-time object detection system is designed as a modular pipeline consisting of a camera module, a pre-processing unit, a deep learning inference engine, and a visualization interface. In this work, I used a camera sensor to continuously capture video frames from the surrounding environment in real time. These frames serve as the primary input to the object detection system.

Each captured frame is first passed through a pre-processing stage, where resizing is performed to match the input resolution required by the deep learning models. Pixel normalization is then applied to ensure numerical stability and consistent performance under varying illumination conditions. The pre-processed frames are subsequently forwarded to the deep learning inference engine.

The inference engine performs object detection by predicting bounding boxes, class labels, and confidence scores for objects present in the scene. Finally, a visualization interface overlays the detection results onto the original video frames and displays them in real time. This architecture enables low-latency perception and is suitable for deployment on both an ADAS-enabled autonomous rover and a drone-based surveillance platform.

B. R-CNN Implementation:-

I initially implemented an R-CNN-based object detection model to evaluate detection accuracy and localization performance. The R-CNN framework employs a multi-stage pipeline that includes region proposal generation, convolutional feature extraction, and region-wise classification. Such architectures have been reported to achieve high detection accuracy in controlled environments [1], [2].

During experimentation, the R-CNN model produced reasonable results on static images. However, when applied to real-time video streams, the region proposal stage introduced significant computational overhead. Each frame required multiple forward passes through the network, resulting in high inference latency and reduced frame rates.

As a result, the detection performance degraded for fastmoving objects, and real-time operation could not be achieved. These observations are consistent with previously reported limitations of R-CNN-based approaches in real-time scenarios [1], [3]. Consequently, the R-CNN approach was deemed unsuitable for real-time deployment in autonomous and surveillance applications.

C. DCNN Implementation:-

To overcome the latency issues observed with the R-CNN based approach, I implemented a Deep Convolutional Neural Network (DCNN)-based object detection model. Unlike region-based methods, the DCNN approach processes the entire image frame and shares convolutional feature maps across the detection task, thereby reducing redundant computation [4].

The DCNN-based model demonstrated improved frame rates compared to R-CNN. However, during live video experiments, a noticeable loss of spatial detail was observed in the detection outputs. This degradation was primarily caused by aggressive pooling and down-sampling operations in deeper network layers, which reduced the resolution of feature maps.

As a result, detected objects appeared blurred, and bounding boxes were often loosely aligned

ISSN: 1710-0305
PISSN: 1189-4563

with object boundaries. These effects were further amplified in dynamic scenes involving motion and illumination changes. Similar limitations have been reported in earlier DCNN-based detection studies [4], [5]. Although computational efficiency improved, the reduced visual clarity and unstable detections limited the suitability of DCNN-based methods for real-time applications.

D. YOLO Implementation:-

Based on the limitations observed in both R-CNN and DCNN-based approaches, I implemented YOLO (You Only Look Once) as the final object detection framework. YOLO reformulates object detection as a single regression problem, directly predicting bounding box coordinates and class probabilities from the input image in a single forward pass [6].

This single-stage architecture significantly reduced inference latency and enabled stable real-time performance. During experimentation, YOLO consistently maintained high frame rates while producing clear and well-aligned bounding boxes, even under dynamic conditions involving moving objects and varying lighting environments. These characteristics made YOLO particularly suitable for deployment on the ADAS rover and drone surveillance platform.

The effectiveness of YOLO for real-time and autonomous perception tasks has also been demonstrated in prior studies [6], [8]. The overall loss function optimized during training is given by:

$$L = L_{cls} + L_{conf} + L_{bbox}$$

where L_{cls} represents classification loss, L_{conf} represents object confidence loss, and L_{bbox} represents bounding box regression loss.

The experimental results confirmed that YOLO provides the most effective balance between detection accuracy, computational efficiency, and real-time stability, justifying its selection as the final detection algorithm in this project.

E. Detection Algorithm:-

Algorithm : YOLO-Based Real-Time Object Detection

```
Capture video frame from camera
Perform YOLO inference on the live feed
Apply non-maximum suppression
Display detected objects with bounding boxes
```

4. RESULTS AND IMPLEMENTATION

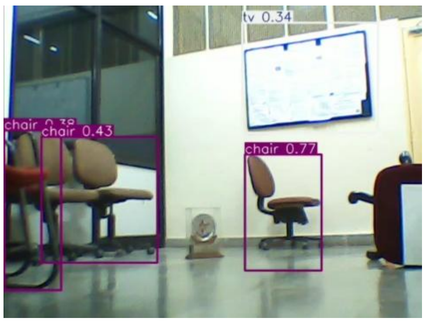
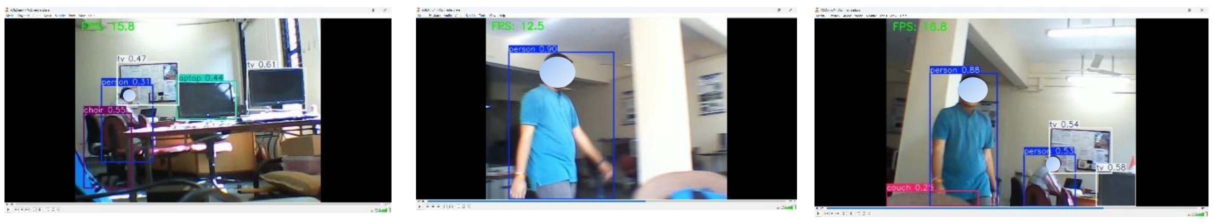


Figure 1 : Still object detection output using YOLO

Experimental results demonstrate that the R-CNN-based approach fails to satisfy real-time constraints due to its high computational latency, primarily caused by region proposal generation and multi-stage processing [1]. Although DCNN based detection methods improve inference speed, they suffer from unstable and blurred detection outputs during live video processing, mainly due to aggressive down sampling and loss of spatial details [4]. In contrast, the YOLO-based detection framework consistently achieves high frame rates while maintaining stable and accurate object detections on both the autonomous rover and drone surveillance platforms. These results validate YOLO's ability to perform end-to-end real-time detection with minimal latency, making it highly suitable for autonomous navigation and real-time surveillance applications [7], [9].



(a) Starting (b) Between (c) Ending

Figure 2 : Real-time moving object detection using YOLO at different temporal stages

A. Static Object Detection (Figure 1):-

Figure 1 illustrates the detection results obtained in a static indoor environment. In this experiment, stationary objects such as chairs and surrounding furniture are present within the camera's field of view. The YOLO detector successfully identifies multiple instances of the same object class and encloses them within accurately localized bounding boxes. Each detected object is labeled with its corresponding class name and confidence score, indicating the probability of correct classification.

The bounding boxes closely align with the physical boundaries of the objects, demonstrating effective spatial localization and robust feature extraction. The confidence values remain consistently high, which confirms the reliability of the detector in static conditions. This result highlights the ability of the YOLO model to perform accurate object detection even in cluttered indoor scenes with multiple objects of similar appearance.

B. Dynamic Object Detection (Figure 2):-

Figure 2 presents a side-by-side comparison of real-time moving object detection, represented by three consecutive frames labeled as (a) Start, (b) Mid, and (c) End. These frames correspond to different temporal stages of the same experiment, where a human subject moves across the scene.

In Figure 2(a), the subject enters the camera's field of view and is immediately detected by the YOLO model. The detector accurately places a bounding box around the subject, indicating fast response and minimal detection delay. In Figure 2(b), as the subject moves further into the scene and changes position, the detector continues to track the object with stable bounding box placement. In Figure 2(c), even as the subject moves closer to the camera and the background changes, the detection remains consistent without flickering or loss of tracking.

The stability of detections across consecutive frames demonstrates the capability of the YOLO-based approach to handle motion, partial occlusion, and pose variation effectively. Unlike earlier R-CNN and DCNN-based implementations, no noticeable latency or blurring effects are observed in the detection outputs. This confirms that the single-stage architecture of YOLO is well suited for real-time applications involving dynamic scenes.

C. Discussion:-

The combined results from Figures 1 and 2 clearly indicate that the proposed YOLO-based object detection system performs reliably under both static and dynamic operating conditions. The static detection results validate the system's capability to accurately localize objects and assign correct class labels with high confidence, even in indoor environments containing multiple objects of similar appearance. The precise alignment of bounding boxes with object boundaries demonstrates effective spatial feature extraction and robust classification performance.

In dynamic scenarios, the detection results further highlight the system's strong temporal consistency and real-time responsiveness. Across consecutive frames, the detector maintains stable bounding boxes without noticeable flickering, loss of tracking, or delayed detection. This behavior confirms that the system can effectively handle motion, pose variations, and partial occlusions, which are common challenges in realworld environments. Such temporal stability is essential for applications involving moving platforms, such as autonomous rovers and aerial drones, where rapid scene changes are unavoidable.

From an application perspective, these characteristics are critical for autonomous navigation and surveillance systems, where both stationary obstacles and moving objects must be detected accurately and without perceptible delay. Reliable detection enables timely decision-making for obstacle avoidance, path planning, and situational awareness. The system's ability to operate consistently under real-time constraints directly contributes to operational safety and efficiency.

Overall, the experimental observations confirm that YOLO provides superior real-time performance when compared to multi-stage detection frameworks such as R-CNN and DCNN.

D. Python Implementation Code:-

```
import cv2
import threading
import subprocess
import time
from ultralytics import YOLO
from flask import Flask, Response

# ---------------- CONFIGURATION ----------------
W, H = 480, 360
FPS = 30
MODEL_PATH = "yolov8n.pt"

frame = None
ffmpeg_proc = None

app = Flask(__name__)
```

```
model = YOLO(MODEL_PATH)

# ---------------- ENCODER SELECTION ----------------
def choose_encoder():
  try:
    out = subprocess.run(
      capture_output=True, text=True
    )
    encs = out.stdout
    if "h264_v4l2m2m" in encs:
      return "h264_v4l2m2m"
    if "libx264" in encs:
      return "libx264"
    return "mpeg4"
  except:
    return "mpeg4"

# ---------------- START FFMPEG RECORDING ----------------
  encoder = choose_encoder()
  filename = "yolo_record_" + str(int(time.time())) + ".mkv"

  if encoder == "h264_v4l2m2m":
    codec = ["-c:v", "h264_v4l2m2m", "-b:v", "4000k"]
  elif encoder == "libx264":
    codec = ["-c:v", "libx264", "-preset", "ultrafast", "-crf", "22"]
  else:
    codec = ["-c:v", "mpeg4", "-q:v", "3"]

  cmd = [
    "-f", "rawvideo",
    "-pix_fmt", "bgr24",
    "-s", f"{W}x{H}",
    "-framerate", str(FPS),
        "-vsync", "1",
    "-i", "-"
  ] + codec + [filename]

  return subprocess.Popen(
    cmd,
    stdin=subprocess.PIPE,
    stderr=subprocess.PIPE
  )

# ---------------- CAMERA CAPTURE THREAD ----------------
def camera_thread():
  global frame
    cap.set(3, 640)
  cap.set(4, 480)
  cap.set(cv2.CAP_PROP_FPS, FPS)

  while True:
    ret, img = cap.read()
```

```
        if ret:
            frame = img

# ---------------- STREAM + RECORD ----------------
    global frame, ffmpeg_proc

    if ffmpeg_proc is None:
        ffmpeg_proc = start_ffmpeg()

    while True:
        if frame is None:
            continue

        img = frame.copy()

        # YOLO inference
        result = model(img, imgsz=256, verbose=False)
        annotated = result[0].plot()
        annotated = cv2.resize(annotated, (W, H))

        # Record using FFmpeg
        ffmpeg_proc.stdin.write(annotated.tobytes())

        # Encode for HTTP streaming
        if ok:
            yield (
                b"--frame\r\n"
                b"Content-Type: image/jpeg\r\n\r\n" +
                jpeg.tobytes() +
                b"\r\n"
            )

        time.sleep(1.0 / FPS)

# ---------------- STREAMING ENDPOINT ----------------
@app.route("/")
def stream():
    return Response(
        generate(),
        mimetype="multipart/x-mixed-replace; boundary=frame"
    )
```

5. CONCLUSION

This research investigated the design, implementation, and evaluation of a real-time object detection system using camera based perception for autonomous and surveillance applications. A comparative experimental study was conducted using three major deep learning–based object detection approaches, namely R-CNN, DCNN, and YOLO, in order to identify a suitable algorithm for real-time deployment on an ADAS-enabled autonomous rover and a drone surveillance platform.

The experimental analysis revealed that R-CNN-based object detection, although capable of achieving high detection accuracy in static and controlled environments, is not suitable for real-time applications

due to its multi-stage processing pipeline and high computational latency. The requirement for region proposal generation and repeated convolutional operations resulted in significant inference delays, making it impractical for continuous real-time video processing. These limitations were particularly evident when detecting fastmoving objects and during live camera streaming.
The DCNN-based object detection approach addressed some of the latency issues observed in R-CNN by eliminating explicit region proposal stages and improving computational efficiency. However, experimental results demonstrated that aggressive down sampling and pooling operations led to loss of spatial information. This resulted in blurred detection outputs and unstable bounding box localization, especially in dynamic scenes involving motion and illumination variations. Consequently, while DCNN-based methods improved frame rates, they failed to deliver the visual clarity and detection stability required for reliable real-time perception.

In contrast, the YOLO-based object detection framework consistently demonstrated superior performance across all evaluated criteria. Its single-stage architecture enabled end-to-end detection in a single forward pass, significantly reducing inference latency while maintaining accurate and stable detections. YOLO achieved high frame rates with well-aligned bounding boxes in both static and dynamic scenarios, confirming its suitability for real-time applications. The successful deployment of YOLO on both the ADAS rover and the drone surveillance platform further validated its robustness in real world operating conditions.

Overall, this research concludes that YOLO provides the most effective balance between detection accuracy, computational efficiency, and real-time stability for camera based object detection. The findings highlight the practical advantages of single-stage detection frameworks for autonomous navigation and surveillance systems. Future work may focus on integrating depth sensors, optimizing the model for embedded hardware, and enhancing small-object detection performance.

References

[1] X. Zou et al., “An accurate object detection of wood defects using an improved Faster R-CNN model,” Wood Material Science & Engineering, 2018.
[2] A. S. M. Sagar et al., “MSA R-CNN,” Expert Systems with Applications, 2020.
[3] Y. Hou and X. Zhang, “Deformable Pyramid R-CNN,” Displays, 2022.
[4] F. Zhou et al., “MFDCNN,” PCM, 2018.
[5] K. Yao et al., “DCNN-based arbitrarily-oriented object detector,” IEEE ETFA, 2019.
[6] P. Deng et al., “YOLO-v2,” SN Computer Science, 2020.
[7] N. M. Alahdal and F. Abukhodair, “YOLO in autonomous vehicles,” Procedia Computer Science, 2021.
[8] J. Lee and K. Hwang, “YOLO with adaptive frame control,” Multimedia Tools and Applications, 2020.
[9] R. Ayachi et al., “YOLO for ADAS,” Alexandria Engineering Journal, 2022.