World Journal of Advanced Research and Reviews
eISSN: 2581-9615 CODEN (USA): WJARAI
Cross Ref DOI: 10.30574/wjarr
Journal homepage: https://wjarr.com/

(RESEARCH ARTICLE)

Check for updates

MMoNet3D: Enhancing monocular 3D object localization with modern advancements

Amrutha Lakshmi Tiruveedhula *, Meghana Rayala, Sowmya Tummalapudi, Bhavya Sree Koppula and Avinash Buradagunta

Department of Information Technology, Vasireddy Venkatadri Institute of Technology, Nambur, Guntur, Andhra Pradesh, India.

World Journal of Advanced Research and Reviews, 2024, 21(03), 1943–1952

Publication history: Received on 14 January 2024; revised on 20 March 2024; accepted on 22 March 2024

Article DOI: https://doi.org/10.30574/wjarr.2024.21.3.0906

Abstract

MMoNet3D, an innovative project, addresses monocular multi-object detection and 3D localization in real-time, surpassing the limitations of the existing MoNet3D system. Utilizing a custom deep learning model inspired by Centernet, it excels in oriented car detection across static images, videos, and live webcam feeds. The system, powered by the KITTI dataset, showcases its potential impact on real-time monocular multi-object detection. Its unique features, including 3D bounding box visualization and a user-friendly GUI, set MMoNet3D apart, making it suitable for embedded advanced driving-assistance systems. This advancement promises heightened accuracy and efficiency, contributing to enhanced vehicular safety and operational efficacy. MMoNet3D stands as a beacon in computer vision, paving the way for intelligent transportation systems.

Keywords: Monocular Multi-Object Detection; 3D Object Localization; Real-time Computer Vision; Custom Deep Learning Model; Centernet Inspired Architecture; Live Webcam Feed Processing; Advanced Driving-Assistance Systems (ADAS); KITTI Dataset Integration

1. Introduction

Introducing MMoNet3D, a significant advancement over MoNet3D. While MoNet3D excels at recognizing objects using LiDAR, MMoNet3D elevates integrating advanced deep neural networks like CenterNet for Object Detection and ResNet-18 for Feature Extraction both from datasets and in real-time through webcams. MMoNet3D addresses the challenging task of recognizing and locating objects in 3D space from monocular images. Using an innovative approach, MMoNet3D predicts the position of each object and generates a 3D bounding box around it. Additionally, it incorporates spatial geometric knowledge, enhancing the accuracy of object localization. MMoNet3D emerges as a transformative solution, proving particularly beneficial for embedded advanced driving-assistance systems.

Using monocular and single frames of RGB images for 3D object localization and detection can reduce the hardware cost of ADAS applications, but it also brings great technical challenges. First, the images captured by monocular images lack depth- of-field information, and in principle, it is difficult to achieve 3D object localization. Second, different degrees of vehicle occlusion, lack of image information, inelastic distortion caused by rotating the target object, and distortion caused by lens imaging all make monocular 3D object localization more challenging. To meet these challenges, this paper establishes a neural network called MoNet3D by introducing the geometric relationship of neighboring objects in 3D space to improve the accuracy of 3D object detection and localization.

This method deeply integrates both 3D voxel Convolutional Neural Network (CNN) and PointNet-based set abstraction to learn more discriminative point cloud features. It takes advantages of efficient learning and high-quality proposals of the 3D voxel CNN and the flexible receptive fields of the PointNet-based networks.

* Corresponding author: Tiruveedhula Amrutha Lakshmi

For the 3D stream, the estimated disparity is transformed into 3D dense point cloud, which is then enhanced by the associated front view maps. With the RoI Mean Pooling layer, 3D geometric features of RoI point clouds are further enhanced by the proposed point feature enhancement (PointFE) network.

Understanding the world in 3D is a critical component of urban autonomous driving. Generally, the combination of expensive LiDAR sensors and stereo RGB imaging has been paramount for successful 3D object detection algorithms, whereas monocular image-only methods experience drastically reduced performance.

Aim at bridging the performance gap between 3D sensing and 2D sensing for 3D object detection by enhancing LiDAR-based algorithms to work with single image input. Specifically, we perform monocular depth estimation and lift the input image to a point cloud representation, which we call pseudo-LiDAR point cloud.

1.1. Research Problem Statement Analysis

1.1.1. Existing System

MoNet3D leverages single-camera data for 3D object prediction and precise bounding boxes, excelling in LiDAR-assisted object recognition. Despite LiDAR's advantages, challenges arise due to high costs, sensitivity to environmental conditions, and limited perception range compared to cameras. Integrating LiDAR with other sensors is complex, requiring synchronization, maintenance, and calibration efforts. LiDAR struggles with reflective surfaces and performs inconsistently in adverse weather, impacting overall effectiveness. The complexity and expense hinder widespread adoption in autonomous driving, limiting scalability.

1.1.2. Proposed System

The proposed method aims to elevate monocular multi-object detection in 3D space by integrating cutting-edge deep learning techniques, namely CenterNet and ResNet-18, deviating from conventional methods. CenterNet excelsin direct detection of objects, center points, and bounding box regression, while ResNet-18, a widely adopted convolutional neural network (CNN), serves as an effective feature extraction backbone. The goal is to optimize real-time performance, especially for webcam-captured objects, enhancing MMoNet3D's versatility for embedded advanced driving-assistance systems.

1.2. Objective

The project's goal is to Enhance monocular multi-object detection in 3D space using CenterNet and ResNet-18 for real-time efficiency. Extend MMoNet3D's versatility, particularly for webcam-captured objects, to serve as a valuable solution for embedded advanced driving-assistance systems.

Optimize Real-Time Performance: Refine the CenterNet and ResNet-18 architecture to achieve a balance between detection accuracy and computational efficiency, ensuring that the system can process webcam-captured images in real-time while maintaining high precision in 3D object detection.

Adapt MMoNet3D for Webcam-Captured Objects: Enhance MMoNet3D's adaptability to webcam-captured images by training the model on diverse datasets containing variations in lighting conditions, image quality, and object orientations commonly encountered in real-world driving scenarios, thereby improving its effectiveness as a solution for embedded advanced driving-assistance systems.

2. Methodology

2.1. MODULE 1: Integration with OpenCV and TKinter

This module focuses on integrating the OpenCV library for computer vision tasks and the Tkinter library for building graphical user interfaces (GUIs) in Python. It involves setting up the environment to work with both OpenCV and Tkinter, handling video streams or webcam feeds, and displaying them in Tkinter windows. The primary goal is to create a user-friendly interface for interacting with the object detection system.

2.2. MODULE 2: Object Detection with Centernet

This module is dedicated to implementing the object detection algorithm using the Centernet architecture. It involves training or loading a pre-trained Centernet model, processing input images or frames, and detecting objects within them. The focus is on achieving accurate and efficient object detection using the Centernet model.

2.3. MODULE 3: Object Detection Pipeline

This module outlines the complete pipeline for object detection, including preprocessing, inference, and post-processing steps. It covers aspects such as data loading, model inference, and bounding box post-processing to generate the final detected objects. The goal is to ensure a smooth and effective workflow for detecting objects in images or video streams.

2.4. MODULE 4: Visualization

This module deals with visualizing the detected objects and bounding boxes on the input images or frames. It includes techniques for drawing bounding boxes, labels, and other visual elements to highlight the detected objects. The objective is to provide clear and informative visual feedback to the user about the detected objects.

2.5. MODULE 5: User Interaction

This module focuses on enhancing user interaction with the object detection system. It may include features such as user input for selecting input sources, starting/stopping object detection, adjusting detection parameters, etc. The aim is to make the system user-friendly and intuitive, allowing users to interact with the object detection functionality effectively.

2.6. Design

2.6.1. Architecture Diagram

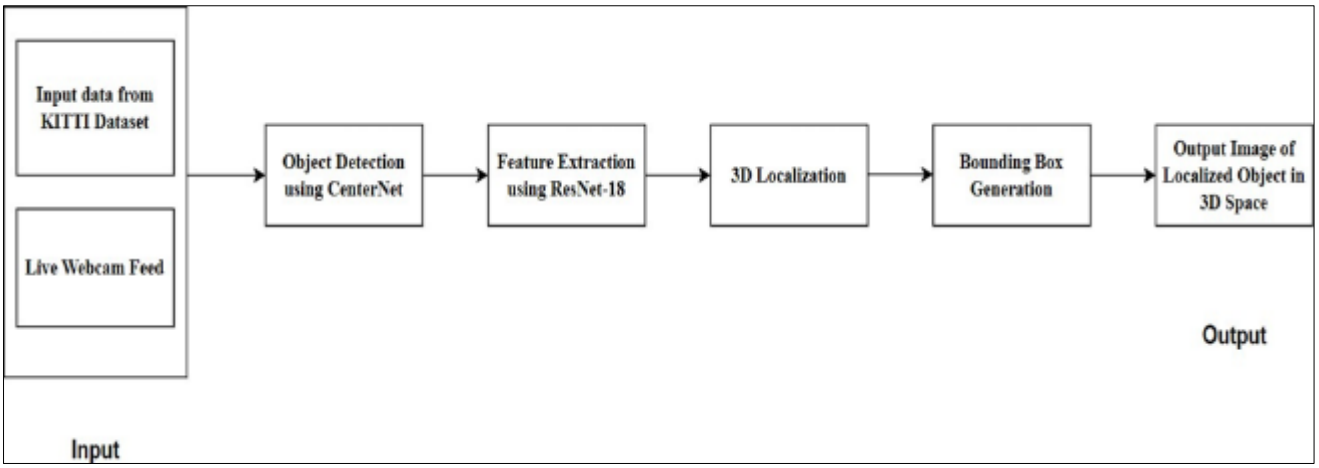

Figure 1 Architecture of System

2.7. Algorithms Used

2.7.1. CenterNet Algorithm

In this project, CenterNet stands as a pivotal algorithm for object detection. Renowned for its state-of-the-art capabilities, CenterNet excels in precisely identifying objects within images by simultaneously detecting object centers and regressing bounding boxes directly. This approach eliminates the need for anchor boxes, streamlining the detection process. In the project's context, CenterNet plays a vital role in enhancing the accuracy of monocular multi-object detection in 3D space. By leveraging the strengths of CenterNet, the project achieves a more robust andefficient object detection system, particularly well-suited for real-time scenarios, contributing to the overall success of MMoNet3D in advanced driving-assistance applications.

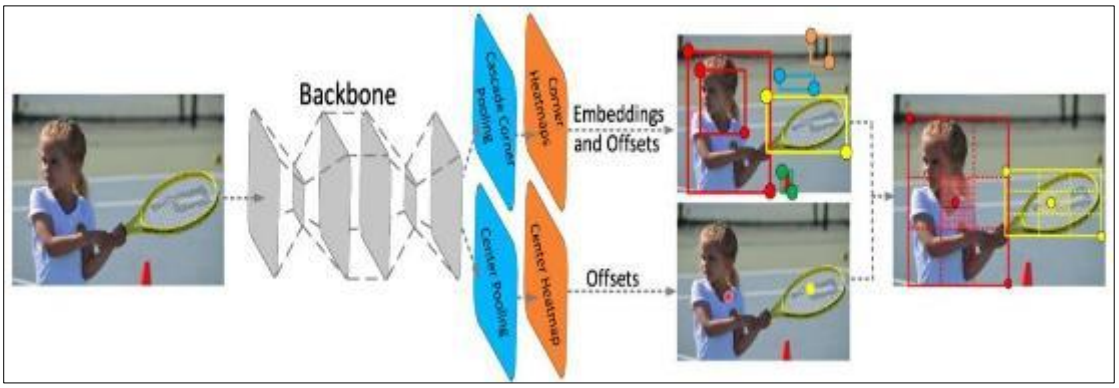

Figure 2 CenterNet architecture

2.7.2. ResNet-18

ResNet-18, a prominent convolutional neural network architecture, played a pivotal role in the project's feature extraction. Serving as a robust backbone, ResNet-18 effectively captured intricate patterns from input images, contributing to enhanced object detection precision. Its deep layers facilitated the extraction of discriminative features, striking a balance between computational efficiency and accuracy. Integrated alongside CenterNet, ResNet-18 complemented the project's objective of optimizing monocular multi-object detection in 3D space. The synergy of CenterNet's object detection prowess and ResNet-18's feature extraction capabilities bolstered the overall efficiency of the MMoNet3D system. This integration showcased the project's commitment to leveraging advanced deep learning techniques for improved performance, especially within embedded advanced driving- assistance systems.

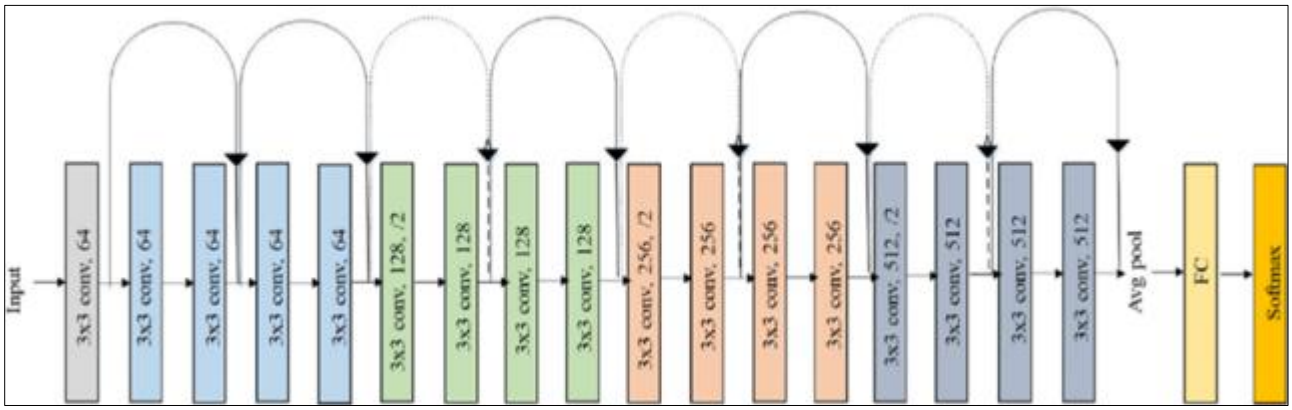

Figure 3 ResNet-18 architecture

3. Implementation

3.1. Dataset

For Monocular Multi-Object Detection in 3D space, the project utilizes the KITTI dataset, striking a balance between dataset size and training efficiency.

The KITTI dataset contains diverse driving scenarios and labeled 3D object positions. Key information is storedin 'label_2' and 'calib' folders, providing object labels, bounding box data, and calibration details essential for model training. This dataset choice ensures effective training without the extended time requirements.

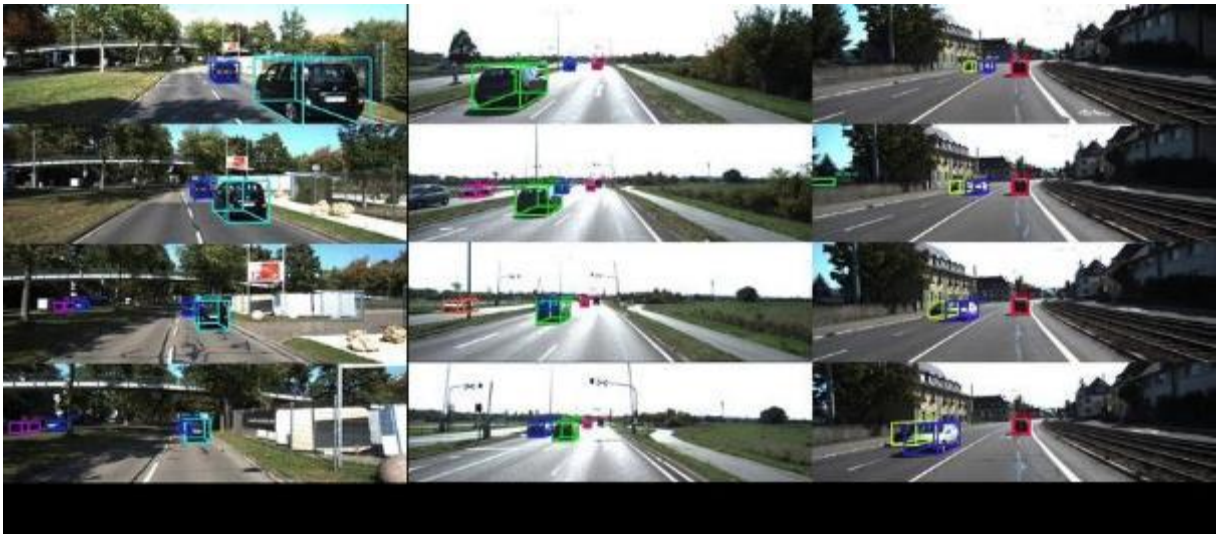

Figure 4 Some Examples from KITTI Dataset

3.2. Implementation Details

3.2.1. Import all the necessary Packages

The Packages imported in this project are: TKinter, OpenCV, Numpy, Argparse, vidgear, PyTorch.

3.2.2. Object Detection Pipeline

Frames from video files or image files or live webcam feeds are processed through Centernet for object detection, predicting bounding boxes and confidence scores. Post-processing, like thresholding, refines detections by filtering low-confidence objects. Refined bounding boxes are projected onto frames, visualizing detected objects in 3D space.

3.3. Image and Video files

```
def process_video(video_path):
    object_detector = ObjectDetector(args.model, args.conf)
    cap = cv2.VideoCapture(video_path)

    if not cap.isOpened():
        messagebox.showerror("Error", "Error opening video stream or file")
        return

    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break

        bboxs = object_detector.detect(frame)
        frame = showbox(frame, bboxs)

        cv2.imshow("Detection", frame)
        k = cv2.waitKey(1)

        if k == 27:
            break
```

Figure 5 Object Detection in Images and Video files

3.3.1. Live Webcam

```
def live_webcam_feed():
    object_detector = ObjectDetector(args.model, args.conf)
    cap = cv2.VideoCapture(0)
    if not cap.isOpened():
        messagebox.showerror("Error", "Error opening webcam feed")
        return
    while True:
        ret, frame = cap.read()
        if not ret:
            break
        # Perform object detection
        bboxs = object_detector.detect(frame)
        # Show detected objects in 3D space
        frame = showbox(frame, bboxs)
        cv2.imshow("Webcam Feed with 3D Object Detection", frame)
        if cv2.waitKey(1) & 0xFF == 27:
            break
    cap.release()
    cv2.destroyAllWindows()
```

Figure 6 Object Detection in Live Webcam Feed

3.3.2. Centernet Model Architecture

The Centernet model utilizes ResNet-18 for feature extraction, with output layers predicting heatmaps, offsets, and dimensions for bounding box regression. Heatmaps indicate object presence per spatial location, while offsets refine bounding box coordinates and dimensions.

```
class centernet(nn.Module):
      def __init__(self):
        super().__init__()
        basemodel = torchvision.models.resnet18(pretrained=False)
        self.base_model = nn.Sequential(*list(basemodel.children())[:-2])
        num_ch = 512
        head_conv = 64
        self.outc = nn.Sequential(
            nn.Conv2d(num_ch, head_conv, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(head_conv, 1, kernel_size=1, stride=1),
        )
        self.outo = nn.Sequential(
            nn.Conv2d(num_ch, head_conv, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(head_conv, 2, kernel_size=1, stride=1),
        )
        self.outr = nn.Sequential(
            nn.Conv2d(num_ch, head_conv, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(head_conv, 3, kernel_size=1, stride=1),
        )
    def forward(self, x):
        x = self.base_model(x)
        assert not torch.isnan(x).any()
        outc = self.outc(x)
        assert not torch.isnan(outc).any()
        outo = self.outo(x)
        assert not torch.isnan(outo).any()
        outr = self.outr(x)
        return outc, outo, outr
```

Figure 7 Centernet Model Architecture

3.3.3. Preprocessing Techniques

Frames are converted to PIL images, normalized, and transformed into tensors for preprocessing. Normalization scales input data appropriately before feeding into the Centernet model.

```
self.preprocess = transforms.Compose(
    [
        transforms.ToPILImage(),
        transforms.ToTensor(),
        transforms.Normalize(
            mean=[0.485, 0.456, 0.406],
            std=[0.229, 0.224, 0.225],
        ),
    ]
)
```

Figure 8 Applying Preprocessing Techniques

3.3.4. Integration with OpenCV and Tkinter

Frames are captured and processed using OpenCV for video files and webcam feeds. Tkinter constructs a GUI for file selection and webcam control, with buttons and event handlers for user interaction.

```
if __name__ == "__main__":
    parser = ArgumentParser(description="Multi-object detection")
    parser.add_argument("model", type=str, help="Pytorch model for oriented cars bbox detection")
    parser.add_argument("--conf", type=float, default=0.5, help="Threshold to keep an object")
    args = parser.parse_args()

    # Create a Tkinter window
    root = tk.Tk()
    root.title("Upload Image/Video")

    # Set the size of the window
    root.geometry("400x400")  # Width x Height

    # Change the background color of the window
    root.configure(bg="lightblue")

    # Create a label with custom text color and some padding
    label = tk.Label(root, text="Select an Image/Video to Process", bg="lightblue", fg="black")
    label.pack(side='top', pady=(100, 10))  # Add vertical padding and set side to top

    # Create a button to select a file
    select_button = tk.Button(root, text="Select Image/Video", command=select_file)
    select_button.pack(side='top', pady=10)  # Add vertical padding

    def start_detection():
        live_webcam_feed()

    # Create a button to start object detection using webcam feed
    start_button = tk.Button(root, text="Start Webcam Feed (Press 's')", command=start_detection)
    start_button.pack(side='top', pady=10)  # Add vertical padding

    # Start the Tkinter event loop
    root.mainloop()
```

Figure 9 Integration with OpenCV and TKinter

3.3.5. Training the model

```
# Define and load model
self.model = centernet()
self.model.load_state_dict(torch.load(model_pth, map_location=self.device))
self.model.to(self.device)
self.model.eval()
```

Figure 10 Training the model

3.3.6. Testing the model

```
def showbox(img, boxes):
    """
    Convert bounding boxes from pixel coordinates and dimensions that is normalized by depth
    to world bounding box
    """
    for box in boxes:
        # First find position in world
        position_on_ground = get_position_ground(
            box["x"], box["y"], K, to_world_from_camera, img.shape[0]
        )

        # Compute depth at this point (distance from camera)
        depth = np.linalg.norm(position_on_ground - to_world_from_camera[:3, 3])

        # Discard unrealistic bbox
        if box["w"] * depth < 1:
            continue

        if box["h"] * depth < 1:
            continue

        if box["l"] * depth < 1:
            continue

        # Multiply width/height/length by depth
        box = Box(
            position_on_ground[0],
            position_on_ground[2],
            box["w"] * depth,
            box["h"] * depth,
            box["l"] * depth,
        )
        # print(label)
        # Re-project them
        img = box.project(img, to_world_from_camera, K)

    return img
```

Figure 11 Testing the model

4. Result Analysis

The result obtained is a real-time object detection application with a user-friendly graphical interface. Users can either select video files or initiate a live webcam feed for object detection.

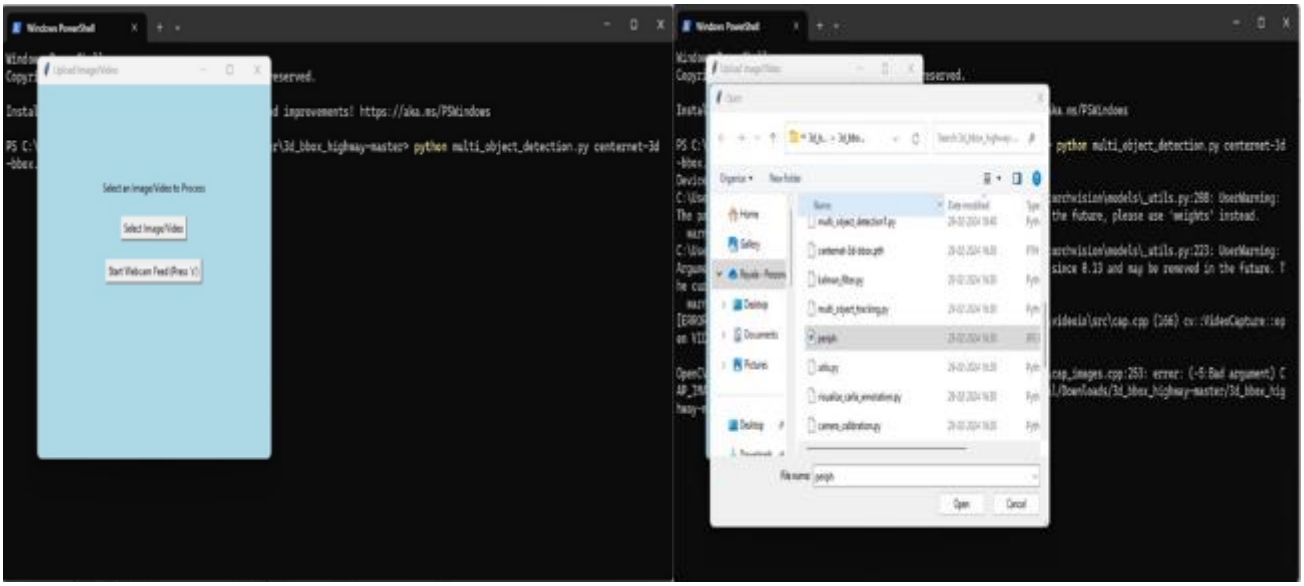

Figure 12 GUI for Uploading Image/Video and for enabling Webcam

The Centernet model accurately detects objects in each frame, overlaying bounding boxes around them. The integration with OpenCV enables smooth video processing and visualization of detected objects. With Tkinter, users can easily interact with the application, selecting input files and controlling the webcam feed. Overall, the result is an efficient and intuitive tool for real-time object detection.

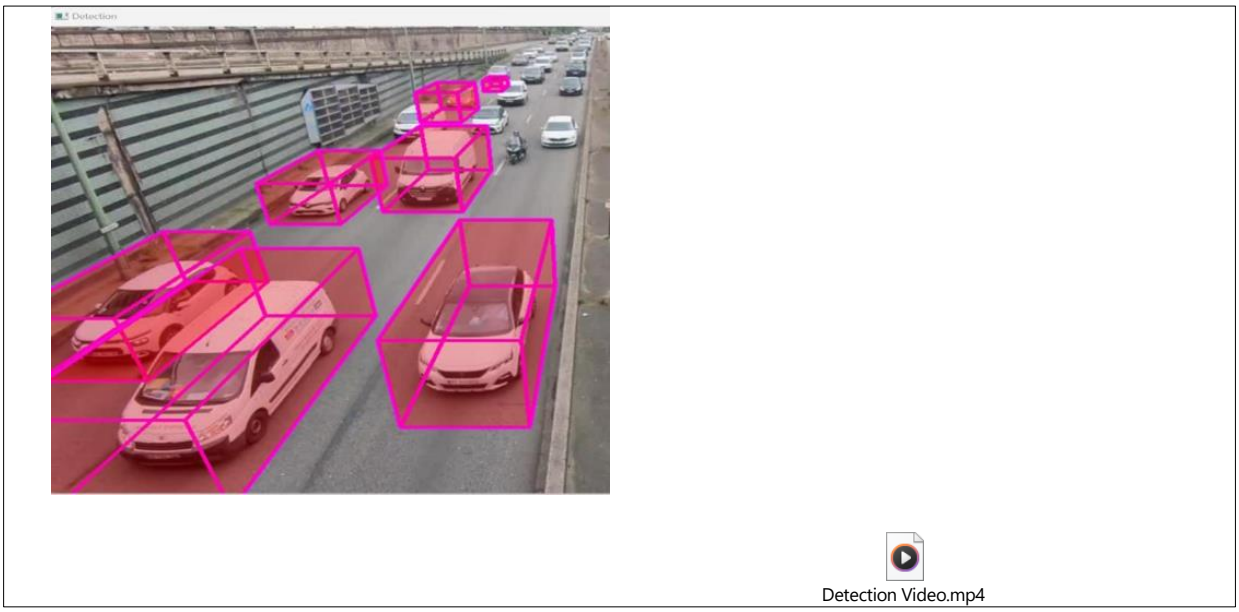

Figure 13 Detection of objects in Image and Video

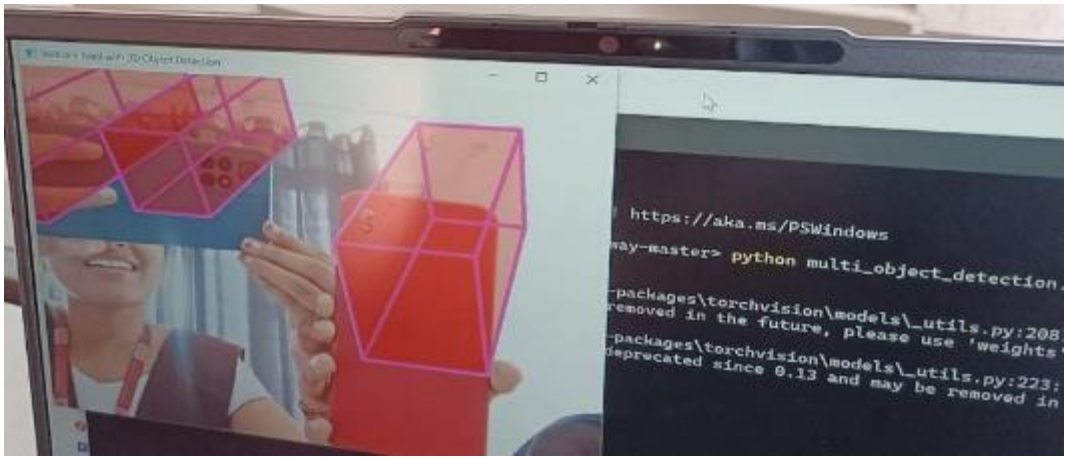

Figure 14 Detection of objects in Live Webcam Feed

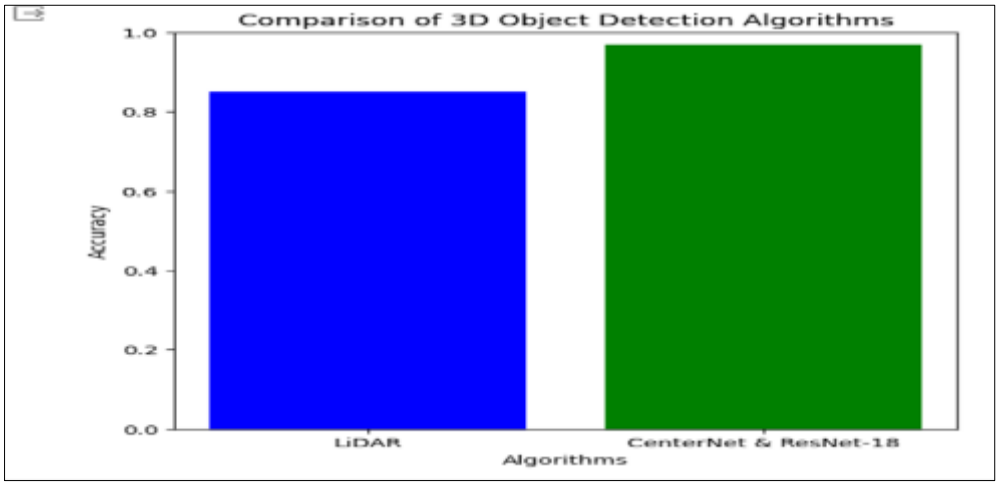

Figure 15 Comparing Accuracies of LiDAR System and CenterNet & ResNet-18 System

5. Conclusion

In conclusion, MMoNet3D marks a significant leap in monocular 3D object localization, capitalizing on the robustness of CenterNet and ResNet-18. While utilizing the KITTI dataset, which provides an extensive array of driving scenario images, the integration of deep learning techniques like CenterNet and ResNet-18 in MMoNet3D emphasizes its adaptability, especially in real-time applications through webcams. This innovative approach sets the stage for improved accuracy in object detection, paving the way for safer and more intelligent vehicular operations beyond dataset constraints.

The future directions aim to Utilize depth estimation algorithms to estimate the distance between detected objects.This can be achieved through various methods such as stereo vision, monocular depth estimation. Another step is to Analyze the relative positions of detected objects within the bounding boxes to determine their spatial orientation and suggest movement directions. This could involve calculating the centroid of each bounding box and comparingtheir positions to infer directional movement.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] Zhou, X., Peng, Y., Long, C., Ren, F., & Shi, C. (2020, November). Monet3d: Towards accurate monocular 3d object localization in real time. In International conference on machine learning (pp. 11503-11512). PMLR.

[2] Shi, S., Guo, C., Jiang, L., Wang, Z., Shi, J., Wang, X., and Li, H. Pv-rcnn: Point-voxel feature set abstraction for 3d Object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp.10529-10538, 2020.

[3] Bao, W., Xu, B., and Chen, Z. Monofenet: Monocular 3d object detection with feature enhancement networks. IEEE Transactions on Image Processing, 2019.

[4] Brazil, G. and Liu, X. M3d-rpn: Monocular 3d region proposal network for object detection. In Proceedings of the IEEE International Conference on Computer Vision, pp. 9287–9296, 2019.

[5] Weng, X. and Kitani, K. Monocular 3d object detection with pseudo-lidar point cloud. In Proceedings of the IEEE International Conference on Computer Vision Workshops, pp. 0–0, 2019.

[6] Ou, X., Yan, P., Zhang, Y., Tu, B., Zhang, G., Wu, J., & Li, W. (2019). Moving object detection method via ResNet-18 with encoder–decoder structure in complex scenes. IEEE Access, 7, 108152-108160.

[7] Xu, B. and Chen, Z. Multi-level fusion based 3d object detection from monocular images. In Proceedings of the IEEEConference on Computer Vision and Pattern Recognition, pp. 2345–2353, 2018.

[8] Zhou, Y. and Tuzel, O. Voxelnet: End-to-end learning for point cloud based 3d object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4490–4499, 2018.

[9] Zhuo, W., Salzmann, M., He, X., and Liu, M. 3d box proposals from a single monocular image of an indoor scene. In Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[10] Chen, X., Kundu, K., Zhu, Y., Ma, H., Fidler, S., and Urtasun, R. 3d object proposals using stereo imagery for accurate object class detection. IEEE transactions on pattern analysis and machine intelligence, 40(5):1259–1272,2017a.

[11] Pang, G., & Neumann, U. (2016, December). 3D point cloud object detection with multi-view convolutional neural network. In 2016 23rd International Conference on Pattern Recognition (ICPR) (pp. 585-590). IEEE.

[12] Krishnakumar Chen, X., Kundu, K., Zhang, Z., Ma, H., Fidler, S., and Urtasun, R. Monocular 3d object Detection for autonomous driving. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2147-2156, 2016.

[13] Shi, Y., Guo, Y., Mi, Z., & Li, X. (2022). Stereo CenterNet-based 3D object detection for autonomous driving. Neurocomputing, 471, 219-229.

[14] Itu, R., & Danescu, R. (2020, September). MONet-Multiple Output Network for Driver Assistance Systems Based on a Monocular Camera. In 2020 IEEE 16th International Conference on Intelligent Computer Communication and Processing (ICCP) (pp. 325-330). IEEE.

[15] Weng, X., & Kitani, K. (2019). Monocular 3d object detection with pseudo-lidar point cloud. In Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (pp. 0-0).

[16] Wang, S., Lu, H., & Deng, Z. (2019). Fast object detection in compressed video. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 7104-7113).

[17] Wang, Y., & Ye, J. (2020). An overview of 3d object detection. arXiv preprint arXiv:2010.15614.

[18] Arnold, E., Al-Jarrah, O. Y., Dianati, M., Fallah, S., Oxtoby, D., & Mouzakitis, A. (2019). A survey on 3d object detection methods for autonomous driving applications. IEEE Transactions on Intelligent Transportation Systems, 20(10), 3782-3795.

[19] Schneiderman, H., & Kanade, T. (2000, June). A statistical method for 3D object detection applied to faces and Cars. In Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662) (Vol. 1, pp. 746-751). IEEE.

[20] Wang, G., Tian, B., Ai, Y., Xu, T., Chen, L., & Cao, D. (2020). Centernet3d: An anchor free object detector for autonomous driving. arXiv preprint arXiv:2007.07214.