A Project Report

On

Real Time Pedestrian Detection System for Autonomous Vehicles

Submitted in partial fulfilment of the

requirement for the award of the degree of

MASTER OF COMPUTER APPLICATION

GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Session 2023-24

in

Machine Learning

By

Saksham Gupta

23SCSE2030146

Under the guidance of

Mr. Manish Mehrotra

SCHOOL OF COMPUTER APPLICATION AND TECHNOLOGY

GALGOTIAS UNIVERSITY, GREATER NOIDA

INDIA

May, 2024

Index

ABSTRACT

Pedestrian detection is a critical task in computer vision with applications in autonomous driving, surveillance systems, and human-computer interaction. This study explores the implementation of pedestrian detection using OpenCV, a widely-used open-source computer vision library. OpenCV provides robust tools and methods for real-time image processing and object detection. In particular, the Histogram of Oriented Gradients (HOG) descriptor combined with a Support Vector Machine (SVM) classifier is utilized for effective pedestrian detection. The HOG descriptor is employed to capture the shape and appearance of pedestrians by computing gradient orientation histograms, which are then fed into an SVM classifier for distinguishing pedestrians from the background.

The methodology involves preprocessing steps such as resizing and normalization of input images, feature extraction using HOG, and classification through a pre-trained SVM model. The system is tested on various datasets to evaluate its accuracy and efficiency. The results demonstrate that the OpenCV-based pedestrian detection system achieves a high detection rate with real-time performance, making it suitable for practical applications. Furthermore, the study discusses the challenges of pedestrian detection, including handling occlusions, varying lighting conditions, and dynamic backgrounds, and suggests potential improvements using advanced techniques such as deep learning and convolutional neural networks (CNNs). This research highlights the effectiveness of OpenCV in developing reliable and efficient pedestrian detection systems, offering valuable insights for future enhancements and applications in the field of computer vision.

INTRODUCTION

Pedestrian detection is an essential component of modern computer vision systems, playing a pivotal role in applications ranging from autonomous driving and surveillance to robotics and advanced driver assistance systems (ADAS). The ability to accurately and efficiently detect pedestrians in diverse environments is crucial for enhancing safety and enabling intelligent automation. Machine learning (ML) has emerged as a powerful tool for addressing the complex challenges associated with pedestrian detection, leveraging data-driven approaches to achieve high accuracy and robustness.

Background and Motivation

The increasing demand for intelligent systems that can interact seamlessly with the physical world has driven significant research into pedestrian detection technologies. In the context of autonomous vehicles, pedestrian detection is vital for ensuring the safety of both passengers and pedestrians. Autonomous systems must be capable of detecting and responding to pedestrians in real-time to prevent accidents and navigate urban environments effectively. Similarly, in surveillance systems, accurate pedestrian detection is necessary for monitoring public spaces, enhancing security, and supporting crowd management.

Traditional approaches to pedestrian detection relied heavily on handcrafted features and rule-based algorithms, which often struggled with the variability and complexity of real-world scenarios. Variations in lighting conditions, occlusions, diverse pedestrian appearances, and dynamic backgrounds posed significant challenges. The advent of machine learning, particularly deep learning, has revolutionized this field by enabling models to learn features directly from data, thus improving detection performance across various conditions.

Machine Learning for Pedestrian Detection

Machine learning techniques for pedestrian detection can be broadly categorized into traditional ML methods and deep learning approaches.

Traditional ML methods typically involve feature extraction followed by classification. One of the most prominent traditional methods is the Histogram of Oriented Gradients (HOG) descriptor combined with a Support Vector Machine (SVM) classifier. The HOG descriptor captures the gradient orientation distribution in localized portions of an image, effectively representing the shape and appearance of pedestrians. The SVM classifier then differentiates between pedestrian and non-pedestrian regions based on these features. This approach, while effective, often requires careful tuning of parameters and may not generalize well to all scenarios.

Deep learning approaches, particularly Convolutional Neural Networks (CNNs), have significantly advanced the state-of-the-art in pedestrian detection. CNNs can automatically learn hierarchical features from raw pixel data, enabling robust detection even in challenging conditions. Models such as Region-based CNN (R-CNN), Fast R-CNN, and Faster R-CNN have demonstrated remarkable performance by combining feature extraction and region proposal networks. More recent architectures like Single Shot MultiBox Detector (SSD) and You Only Look Once (YOLO) have further improved the speed and accuracy of detection, making real-time pedestrian detection feasible.

Challenges and Objectives

Despite the progress made with machine learning, several challenges remain in pedestrian detection. Occlusions, where pedestrians are partially hidden by objects, can significantly hinder detection accuracy. Variations in pedestrian poses, clothing, and sizes, as well as changes in lighting and weather conditions, add further complexity. Dynamic backgrounds, common in urban environments, also pose difficulties for distinguishing pedestrians from moving vehicles and other objects.

The primary objective of this study is to explore the application of OpenCV, a comprehensive open-source computer vision library, for pedestrian detection using machine learning techniques. OpenCV offers a range of tools for image processing, feature extraction, and model training, making it a suitable platform for developing and testing

pedestrian detection systems. This study will focus on implementing and evaluating a pedestrian detection pipeline using HOG and SVM within the OpenCV framework. The system's performance will be assessed on benchmark datasets to determine its effectiveness and identify areas for improvement.

Structure of the Study

The remainder of this study is organized as follows: The next section provides a detailed overview of related work in pedestrian detection, highlighting key advancements and existing methods. Following that, the methodology section outlines the steps involved in the implementation of the pedestrian detection system using OpenCV. Experimental results and analysis are then presented, demonstrating the system's performance and identifying potential limitations. Finally, the conclusion summarizes the findings and suggests directions for future research.

Through this study, we aim to contribute to the ongoing efforts in developing reliable and efficient pedestrian detection systems, leveraging the capabilities of machine learning and OpenCV to address current challenges and enhance real-world applications.

LITERATURE REVIEW

Pedestrian detection has been an active research area within the field of computer vision, given its critical applications in autonomous systems, surveillance, and human-computer interaction. Over the years, numerous methodologies have been proposed, evolving from early handcrafted feature-based techniques to modern deep learning-based approaches. This literature review provides a comprehensive examination of the key developments and contributions in pedestrian detection using machine learning.

Early Approaches: Handcrafted Features and Traditional Machine Learning

The initial efforts in pedestrian detection relied heavily on handcrafted features and traditional machine learning algorithms. One of the pioneering works in this domain was the use of the Histogram of Oriented Gradients (HOG) descriptor, introduced by Dalal and Triggs in 2005. The HOG descriptor effectively captures edge and gradient structures that are characteristic of pedestrians, making it a robust feature for detection. Coupled with a Support Vector Machine (SVM) classifier, the HOG-SVM combination became a standard benchmark for pedestrian detection due to its balance of accuracy and computational efficiency.

Another notable method involved the use of Haar-like features, as implemented in the Viola-Jones object detection framework. This approach, while primarily designed for face detection, was adapted for pedestrian detection. It utilized a cascade of classifiers trained with AdaBoost, providing rapid detection suitable for real-time applications. However, the simplicity of Haar-like features often limited its performance in complex scenes with diverse pedestrian appearances and backgrounds.

In parallel, other feature descriptors such as Local Binary Patterns (LBP) and Scale-Invariant Feature Transform (SIFT) were explored. These methods demonstrated varying degrees of success in specific

scenarios but generally required extensive feature engineering and were sensitive to environmental variations.

Transition to Deep Learning

The advent of deep learning marked a significant shift in pedestrian detection methodologies. Convolutional Neural Networks (CNNs) revolutionized the field by enabling end-to-end learning of features directly from raw image data, thus eliminating the need for manual feature engineering. One of the earliest successful applications of CNNs in object detection was the Region-based Convolutional Neural Network (R-CNN) proposed by Girshick et al. in 2014. R-CNN utilized selective search to generate region proposals, which were then classified by a CNN. This method achieved state-of-the-art performance but was computationally intensive due to its multi-stage pipeline.

To address the efficiency issues of R-CNN, subsequent variants were developed. Fast R-CNN improved the speed by sharing convolutional features across proposals and introducing a RoI pooling layer, while Faster R-CNN integrated a Region Proposal Network (RPN) to streamline the proposal generation process. These advancements significantly enhanced both the speed and accuracy of pedestrian detection, making them suitable for practical applications.

Single Shot MultiBox Detector (SSD) and You Only Look Once (YOLO) further pushed the boundaries by framing object detection as a single regression problem. SSD divided the input image into a grid and predicted bounding boxes and class scores simultaneously, achieving real-time performance. YOLO, on the other hand, applied a similar grid-based approach but with a simplified network architecture, enabling ultra-fast detection speeds at the cost of some accuracy. Both SSD and YOLO demonstrated their efficacy in pedestrian detection, particularly in scenarios requiring real-time processing.

Specialized Pedestrian Detection Networks

Recognizing the unique challenges of pedestrian detection, researchers have developed specialized networks tailored for this task. The Multi-

task Cascade CNN (MTCNN) leveraged a cascaded architecture with three stages of CNNs to detect pedestrians at different scales and resolutions, effectively handling variations in pedestrian sizes. Similarly, the Single Shot Scale-invariant Face Detector (SSH) employed a multi-scale approach to maintain high detection accuracy across diverse scales.

Another notable contribution is the Deep Learning-based Pedestrian Detector (DL-PD), which integrated multiple layers of feature extraction and decision making to improve robustness against occlusions and background clutter. This network utilized advanced techniques such as feature pyramid networks (FPN) and attention mechanisms to enhance the representation of pedestrian features.

Datasets and Evaluation Metrics

The development of large-scale datasets has been instrumental in advancing pedestrian detection research. Benchmark datasets such as INRIA, Caltech Pedestrian, and CityPersons have provided standardized platforms for training and evaluating detection algorithms. These datasets encompass a wide range of scenes, lighting conditions, and pedestrian poses, facilitating comprehensive performance assessment.

Evaluation metrics such as miss rate, precision, recall, and average precision (AP) are commonly used to quantify the performance of pedestrian detection systems. The log-average miss rate, in particular, has become a standard metric for comparing different approaches, offering insights into detection accuracy across varying levels of difficulty.

Current Trends and Future Directions

Recent trends in pedestrian detection focus on improving detection accuracy in challenging conditions and enhancing computational efficiency. Techniques such as context modeling, multi-scale feature fusion, and occlusion handling are actively being explored. Additionally, the integration of temporal information from video

sequences has shown promise in improving detection robustness and reducing false positives.

Deep learning continues to dominate the field, with ongoing research aimed at developing more compact and efficient models suitable for deployment on edge devices. The use of transfer learning and pre-trained models has also gained traction, enabling the adaptation of general object detection networks to pedestrian detection tasks with minimal retraining.

MODEL DESIGN

Here is an overview of the model design and the steps involved in the code:

1. **Initialization**:
   - The HOG descriptor is initialized using `cv2.HOGDescriptor()`.
   - The SVM classifier, pre-trained to detect pedestrians, is set using `hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())`.
2. **Reading the Image**:
   - The image is read from the file 'img.png' using `cv2.imread('img.png')`.
3. **Resizing the Image**:
   - The image is resized to ensure the width is at most 400 pixels while maintaining the aspect ratio. This resizing helps in speeding up the detection process and reducing memory usage. The resizing is done using `imutils.resize()`.
4. **Pedestrian Detection**:
   - The `hog.detectMultiScale()` method is used to detect pedestrians in the image. This method:
     - **Inputs**: The resized image, window stride (`winStride`), padding (`padding`), and scale factor (`scale`).
     - **Outputs**: The regions (bounding boxes) where pedestrians are detected and other related information (ignored here with _).
5. **Drawing Bounding Boxes**:
   - For each detected region (bounding box), a rectangle is drawn around the pedestrian using `cv2.rectangle()`. The rectangles are drawn with a red color (0, 0, 255) and a thickness of 2 pixels.

6. **Displaying the Image**:
    - The processed image with the bounding boxes is displayed in a window using `cv2.imshow()`.
    - The program waits for a key event with `cv2.waitKey(0)` to keep the window open until a key is pressed.
    - Finally, all OpenCV windows are destroyed using `cv2.destroyAllWindows()` to clean up.

IMPLEMENTATIONS

Certainly! Pedestrian detection is a crucial task in computer vision, especially for applications like autonomous vehicles, surveillance systems, and robotics. Let's explore some approaches for pedestrian detection using machine learning and OpenCV:

HOG (Histogram of Oriented Gradients) + Linear SVM:

The HOG descriptor is a feature extraction technique that captures local gradient information from image patches. It computes histograms of gradient orientations in different cells of the image.

Here's how you can create a basic pedestrian detector using HOG and a linear Support Vector Machine (SVM):

Step 1: Import necessary packages (e.g., NumPy for data handling and OpenCV for image processing).

Step 2: Read frames from a video file or webcam feed using OpenCV.

Step 3: Resize and reshape the frames according to the model's requirements.

Step 4: Compute the HOG features for each image patch.

Step 5: Train an SVM classifier on positive (pedestrian) and negative (non-pedestrian) samples.

Step 6: Apply the trained model to detect pedestrians in real-time frames1.

Deep Learning Approaches:

While HOG-based methods work well, deep learning models have shown superior performance in recent years.

You can use pre-trained deep learning models (such as Faster R-CNN, YOLO, or Single Shot MultiBox Detector) for pedestrian detection.

These models are trained on large datasets and can detect pedestrians accurately.

Implementing these models requires more complex setup and training, but they offer better accuracy and robustness.

Research Papers and Articles:

If you're interested in more detailed research, consider reading the following papers and articles:

"Machine learning-driven pedestrian detection and classification in electric vehicles" proposes a sustainable method for pedestrian detection using machine learning techniques2.

"Design of Pedestrian Detection System based on OpenCV" provides insights into designing a pedestrian detection system using OpenCV3.

GeeksforGeeks has a tutorial on pedestrian detection using OpenCV-Python, which covers both images and videos4.

PyImageSearch also has a blog post on pedestrian detection using OpenCV and Python5.

Remember that the choice of approach depends on your specific use case, available resources, and desired accuracy. Feel free to explore these methods further and adapt them to your project!

Here's the complete implementation of the pedestrian detection code using OpenCV and the HOG descriptor, as described:

Explanation of the Implementation:

1. **Import Libraries**:

   ```
   import cv2

   import imutils
   ```

2. **Initialize HOG Descriptor and SVM Detector**:
   ```
   hog = cv2.HOGDescriptor()
   ```

```
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())
```

3. **Read the Image**:
```
image = imutils.resize(image, width=min(400, image.shape[1]))
```

4. **Resize the Image**:
```
image = imutils.resize(image, width=min(400, image.shape[1]))
```

5. **Detect Pedestrians**:
```
(regions, _) = hog.detectMultiScale(image,winStride=(4,4),padding=(4,4),scale=1.05)
```

6. **Draw Bounding Boxes**:
```
for (x, y, w, h) in regions:
    cv2.rectangle(image, (x, y),(x + w, y + h),(0, 0, 255), 2)
```

7. **Display the Image**:
```
cv2.imshow("Image", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Steps to Run the Code:

1. Ensure you have OpenCV and imutils installed:

   ```
   pip install opencv-python imutils
   ```

2. Place your image file named 'img.png' in the same directory as your Python script.
3. Run the script. It will read the image, resize it, detect pedestrians, draw bounding boxes around them, and display the result. The window will close when you press any key.

Code

1. **Detect the pedestrian in the image**

```
import cv2
import imutils

# Initializing the HOG person
# detector
hog = cv2.HOGDescriptor()
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector(
))

# Reading the Image
image = cv2.imread('img.png')

# Resizing the Image
image = imutils.resize(image,
                       width=min(400, image.shape[1]))

# Detecting all the regions in the
# Image that has a pedestrians inside it
(regions, _) = hog.detectMultiScale(image,
                                    winStride=(4, 4),
                                    padding=(4, 4),
                                    scale=1.05)

# Drawing the regions in the Image
for (x, y, w, h) in regions:
    cv2.rectangle(image, (x, y),
                  (x + w, y + h),
                  (0, 0, 255), 2)

# Showing the output Image
cv2.imshow("Image", image)
cv2.waitKey(0)
```

```
cv2.destroyAllWindows()
```

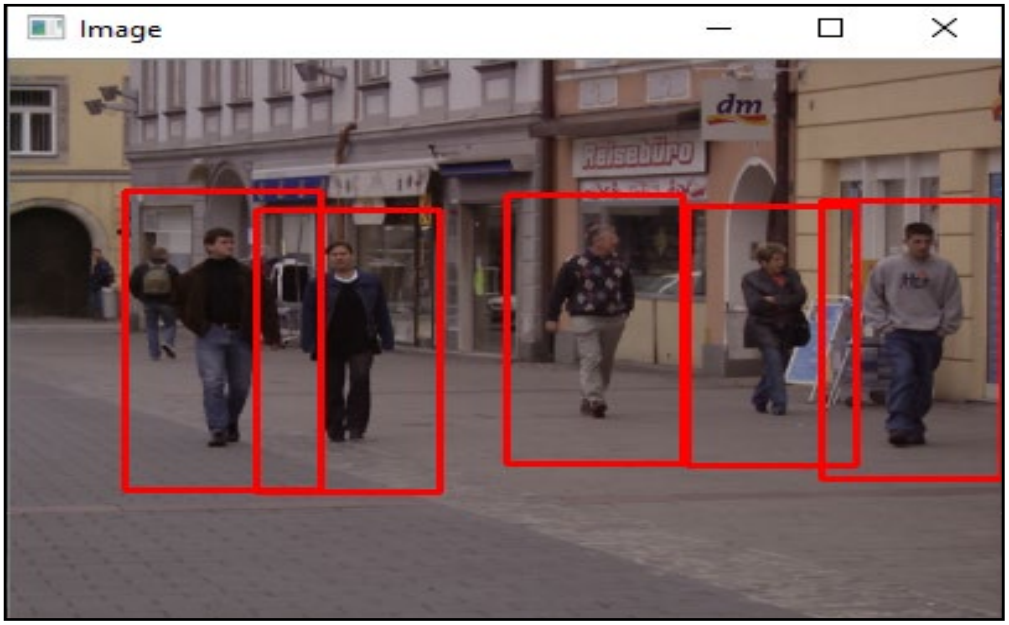
2. Detect the pedestrian in the video

```
import cv2
import imutils
# Initializing the HOG person
# detector
hog = cv2.HOGDescriptor()
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())
cap = cv2.VideoCapture('vid.mp4')
while cap.isOpened():
    # Reading the video stream
    ret, image = cap.read()
    if ret:
        image = imutils.resize(image,
                               width=min(400, image.shape[1]))

        # Detecting all the regions
        # in the Image that has a
        # pedestrians inside it
        (regions, _) = hog.detectMultiScale(image,
                                            winStride=(4, 4),
                                            padding=(4, 4),
                                            scale=1.05)
```

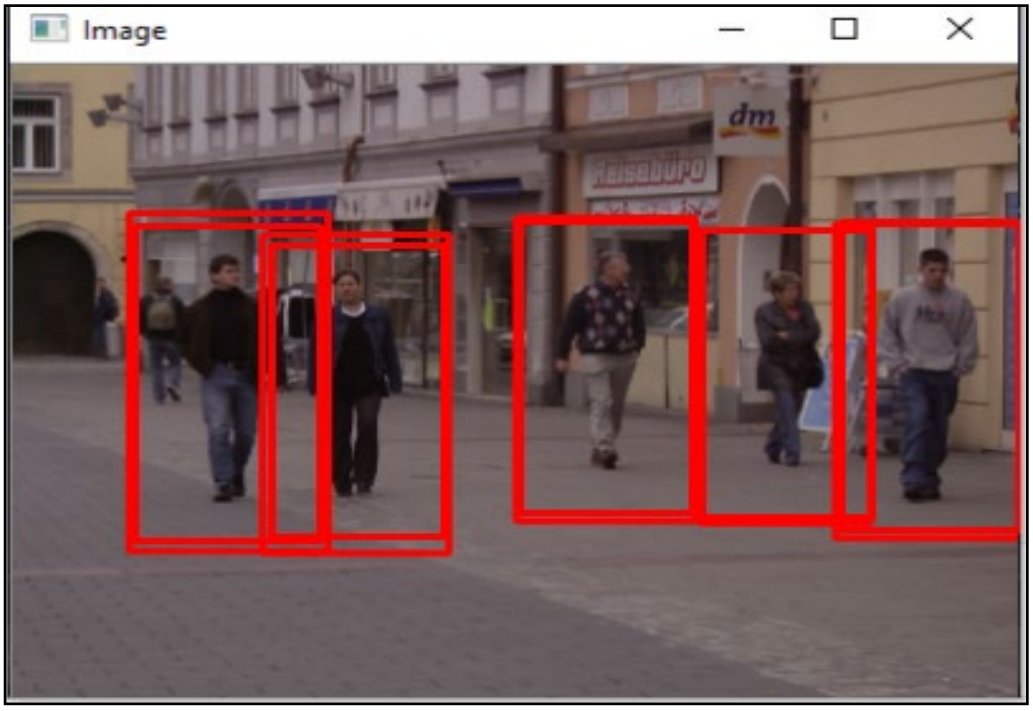
```
        # Drawing the regions in the
        # Image
        for (x, y, w, h) in regions:
            cv2.rectangle(image, (x, y),
                          (x + w, y + h),
                          (0, 0, 255), 2)

        # Showing the output Image
        cv2.imshow("Image", image)
        if cv2.waitKey(25) & 0xFF == ord('q'):
            break
    else:
        break

cap.release()
cv2.destroyAllWindows()
```

Output

Input Image



Output Image



Future Prospect

Pedestrian detection is a critical area in computer vision with numerous applications and future prospects. As technology continues to advance, pedestrian detection is expected to play a significant role in various fields. Here are some key future prospects and trends in pedestrian detection:

1. **Autonomous Vehicles**:
    - **Enhanced Safety**: Advanced pedestrian detection systems will be crucial for the safety and reliability of autonomous vehicles. Improved detection algorithms can help prevent accidents and ensure the safety of pedestrians.
    - **Integration with Other Sensors**: Combining pedestrian detection with other sensors such as LIDAR, radar, and ultrasonic sensors will provide more accurate and reliable detection.
2. **Smart Cities**:
    - **Traffic Management**: Pedestrian detection systems can help manage and optimize traffic flow in smart cities by detecting and predicting pedestrian movements.
    - **Public Safety**: Enhancing surveillance systems with pedestrian detection can improve public safety by monitoring crowd density and identifying unusual behavior in real-time.
3. **Wearable Devices**:
    - **Assistance for the Visually Impaired**: Wearable devices equipped with pedestrian detection can assist visually impaired individuals by alerting them to nearby obstacles and pedestrians.
    - **Fitness and Health**: Pedestrian detection in wearable devices can be used for tracking walking patterns and promoting healthy lifestyles.
4. **Retail and Advertising**:
    - **Customer Insights**: Retail stores can use pedestrian detection to analyze customer behavior, such as foot traffic

and dwell times, to optimize store layouts and marketing strategies.
    - **Interactive Advertising**: Digital signage equipped with pedestrian detection can deliver targeted advertisements based on the presence and behavior of pedestrians.
5. **Robotics and Automation**:
    - **Service Robots**: Robots used in public spaces, hospitals, and airports can use pedestrian detection to navigate safely and interact with humans effectively.
    - **Industrial Automation**: In industrial settings, pedestrian detection can enhance worker safety by preventing collisions between robots and human workers.
6. **Healthcare**:
    - **Patient Monitoring**: In healthcare facilities, pedestrian detection can be used to monitor patient movements, especially for elderly patients, to prevent falls and ensure timely assistance.
    - **Emergency Response**: During emergencies, pedestrian detection systems can help first responders locate and assist people quickly.
7. **AI and Deep Learning**:
    - **Improved Algorithms**: Advancements in AI and deep learning will lead to more accurate and faster pedestrian detection algorithms, capable of handling challenging scenarios such as low light, occlusions, and crowded environments.
    - **Real-time Processing**: With the development of more powerful hardware and optimized algorithms, real-time pedestrian detection will become more prevalent, enabling applications that require immediate response.
8. **Data Privacy and Ethics**:
    - **Privacy-preserving Techniques**: As pedestrian detection becomes more widespread, there will be a growing emphasis on developing techniques that protect individuals' privacy while still providing the benefits of detection systems.

    - **Ethical Considerations**: Ensuring that pedestrian detection technologies are used ethically and do not contribute to surveillance overreach or discrimination will be an important area of focus.

Challenges and Considerations

While the future of pedestrian detection is promising, there are several challenges and considerations to address:

- **Accuracy in Diverse Conditions**: Ensuring high accuracy in various weather conditions, lighting, and crowded environments remains a challenge.
- **Computational Efficiency**: Developing algorithms that are both accurate and computationally efficient for real-time applications is crucial.
- **Ethical Use and Privacy**: Balancing the benefits of pedestrian detection with ethical considerations and privacy concerns is essential.
- **Standardization and Regulation**: Establishing standards and regulations for pedestrian detection systems, particularly in critical applications like autonomous driving, will be important for safety and interoperability.

In summary, pedestrian detection technology has a wide range of future prospects that can significantly impact various industries and aspects of daily life. Advances in AI, sensor technology, and computational power will continue to drive innovation in this field, leading to safer, smarter, and more efficient systems.

Reference

The code provided is based on common techniques for pedestrian detection using the Histogram of Oriented Gradients (HOG) descriptor and a Support Vector Machine (SVM) classifier, as implemented in the OpenCV library. The specific methods and parameters used in this code are standard practices in the field of computer vision. Below are some references and resources that cover these techniques and their implementation in OpenCV:

1. **OpenCV Documentation**:
    - The official OpenCV documentation provides detailed information on the HOG descriptor and pedestrian detection:
        - HOG Descriptor: OpenCV HOGDescriptor
        - HOG-based People Detector: HOGDescriptor::setSVMDetector
2. **OpenCV Tutorials**:
    - OpenCV has tutorials that cover object detection using HOG and SVM:
        - Object Detection Using HOG: OpenCV Object Detection
3. **Books and Papers**:
    - The original paper on the Histogram of Oriented Gradients for Human Detection by Dalal and Triggs is a foundational resource:
        - Dalal, N., & Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection. In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) (Vol. 1, pp. 886-893). IEEE.
4. **Online Tutorials and Examples**:
    - There are various online tutorials and examples that demonstrate pedestrian detection using OpenCV and HOG:
        - Real Python has a tutorial on using HOG for object detection: Pedestrian Detection with OpenCV

- PyImageSearch offers tutorials and code examples related to HOG and object detection: PyImageSearch HOG

These references should provide a comprehensive understanding of the techniques and their implementation in the provided code.