THE HAWKS
VEHICLE CUT-IN DETECTION

ABSTRACT

The project on vehicle cut-in detection uses YOLOv5 to detect real-time vehicle cut-in points and collision warnings through live feed from a webcam. The torch.hub.load function is used to load the model of YOLOv5. In order to detect vehicle cut-ins, frames are processed by the YOLOv5 model trained on IDD dataset. Knowing relative speed assumptions and dimensions of vehicles, distance estimation and TTC calculation are done. Bounding boxes with annotated detected vehicles as well as labels along with displayed distance and TTC information of the frame. If TTC goes below a set threshold value, indicating an impending collision, an audible alert will be produced. OpenCV, streamlit, and torch libraries are employed for video processing, interface design and deep learning respectively in this implementation.

INTRODUCTION

Safety and efficiency in transportation systems are a major priority in today's rapidly changing technological landscape. This project employs the latest deep learning techniques, more specifically YOLOv5, to create a reliable real-time vehicle recognition and collision warning system to cater for these challenges. With this system working on live video feed obtained from a webcam, it can detect vehicles with high precision and performance.

This project has two main goals; firstly, detecting accurately whether or not there has been any cut-ins within the video stream using YOLOv5 model and secondly, giving out timely alerts of impending collisions that are based on Time-to-Collision (TTC) values that have already been computed. For instance, PyTorch is used to integrate YOLOv5 model which as far as object detection tasks are concerned is known for its speed and accuracy hence enabling every frame of the video feed to be inferenced upon in real-time. OpenCV is used for video streaming processing because it facilitates efficient framing capturing as well as manipulating.

As soon as vehicles cut in, the system uses known vehicle dimensions and focal length parameters to calculate the distance between them and the camera. This is used to measure TTC which is important in predicting possible accident situations. Situational awareness for users is enhanced by bounding box annotations, label overlays directly on the video stream.

For real-time monitoring and to aid user interaction, Streamlit; a Python framework for creating interactive web applications is integrated into this system. The interface shows annotated processed video feed and dynamic updates of collision warnings based on calculated TTC values. If a detected vehicle’s TTC falls below a critical threshold, an audible alert will be triggered asynchronously indicating an imminent collision thus requiring immediate attention from the driver.

METHODOLOGY

The goal will be to design a deep learning-based vehicle cut-ins detection and collision warning system in real-time. This, in simpler words, comprises the detection of vehicles in every frame of the live video streaming and distance estimation of vehicles from the camera to calculate correct Time-to-Collision in order to provide proactive warnings against possible collision scenarios.

Model Selection and Integration

In the vehicle cut-in detection project, YOLOv5 is chosen for the deep learning model since it is efficient and accurate in detection. PyTorch is utilized in implementing YOLOv5; it's a very famous framework for deep learning applications. This model will be used because it has real-time functionality for processing video streams that are captured from a webcam. This model will further be trained using the IDD dataset.

Video Stream Processing

OpenCV or Open Source Computer Vision Library is utilized to read and process the video stream from the webcam feed. All frames are then supplied to the YOLOv5 model for object detection, where each model works on the RGB form of each frame; hence, vehicle detection with predefined classes and confidence scores is possible.

Distance Estimation

The system calculates the distance of vehicles from the camera after detecting their presence in the frame. This is enabled through known dimensions of standard vehicles and the focal length of the camera. The pixel width of each vehicle detected within a frame is converted into distance units with physical dimensions using simple geometric principles.

Time-to-Collision Calculation

First of all, the TTC is calculated by using estimated distances and assuming constant relative velocity between the camera (observer) and the detected. Another very important metric, for risk assessment in particular, is a time to collision, defined as time remaining until a potential collision if the current relative speed is maintained.

Visualization and Aural Alerts

Detected vehicles are annotated in the video stream by coloured bounding boxes with class and confidence score labels. Distance measurements and calculated TTC values are displayed as text overlays in proximity to the bounding box of each detected vehicle. In case the TTC value for any detected vehicle falls below a user-defined safety threshold, a message warning to that effect will flash on the video feed. Simultaneously, an asynchronous thread triggers an audible alert so that immediate attention will be drawn to the impending collision scenario.

UI Development

The project uses Streamlit, a Python framework that enables users to build interactive web applications, as a way of interacting with users and providing real-time monitoring. Streamlit provides an interface representing the processed sequence of video feeding real-time with annotations updating as computation for vehicle detection and TTC estimation is being carried out in real-time. This gives better situational awareness to the user and allows the user to be responsive in decisions in occurrence of hazardous situations.

ISSUES FACED AND RESULTS

The difficulties in the development process included performance optimization of real-time video processing with YOLOv5 and distance and TTC estimation in changing conditions. Moreover, the UI based on Streamlit was refactored for responsiveness. Notwithstanding, it has achieved robust vehicle cut-ins detection with accurate collision warnings in live video feeds. The results were notably correct in vehicle identification, distance estimation, and TTC estimation; real-time performance was maintained on standard hardware. It provided an intuitive monitoring and alerting integrated UI, increasing safety measures by issuing timely visual and auditory notifications. Looking ahead, improvements are oriented toward enhancing algorithms, enlarging functionality already available with multi-camera support, and validation of deployment in real traffic scenarios showing, therefore, its potential to improve transportation safety and efficiency.

CONCLUSION

The synergy between advanced deep learning techniques and practical safety applications is therefore governmentalized in this project. This would offer not only a demonstration of technical prowess if they are to deploy the YOLOv5 real-time vehicle cut-in detection and collision warning, but highly pre-emptive measures toward safe and efficient transportation systems also. This will keep setting standards in safety technology with its continuous development and fine-tuning as it makes the way toward smarter, more responsive urban environments.