Cairo University
Faculty of Engineering

Credit Hours system
Mechanical Department

Lane Detection for Autonomous Vehicles

Ali Hossam Monier	1180061
Ammar Mahmoud Elsaied	1190079
Ammar Yasser Mohamed	1190134
Mariam Essam Mohamed	1180027

Table of Contents

Lane Detection for Autonomous Vehicles

Introduction

The Lane and Curve Detection project uses image processing techniques to create an advanced system for recognizing and tracking lanes and curves on highways. The system will analyze video or picture data acquired by a camera placed on a vehicle using computer vision techniques. The project intends to improve the safety and efficiency of autonomous driving systems, advanced driver assistance systems (ADAS), and other related applications by precisely detecting and monitoring lanes and curves.

Project Objectives

1. Lane Detection: The project's major goal is to detect and localize the lanes on the road. In real-time, the technology will recognize lane markers such as dashed and solid lines and establish their locations and orientations.

2. Curve Detection: In addition to lane detection, the project intends to recognize road bends. Bends, turns, and junctions are examples of curves. The system will recognize and track these curves correctly, delivering useful information to navigation and control systems.

3. Robustness and Accuracy: The system will be designed to work robustly and accurately in variable lighting circumstances, weather conditions (e.g., rain, fog), and road surface conditions (e.g., potholes, faded lane markings).

4. Real-time Processing: The project will concentrate on creating efficient algorithms capable of processing image or video data in real-time. To provide fast and responsive recognition of lanes and bends, the system should be capable of analyzing frames at a high frame rate.

Methodology

To achieve the goals, the research will use a combination of image processing and computer vision techniques.

Procedure Stages

1) **Image Acquisition**: Capture images or video frames using a camera or other image-capturing device and read the frames.

2) **Image Preprocessing**: The acquired photos or video frames will be subjected to preprocessing operations in order to improve important properties such as contrast improvement, noise reduction, and image stabilization.

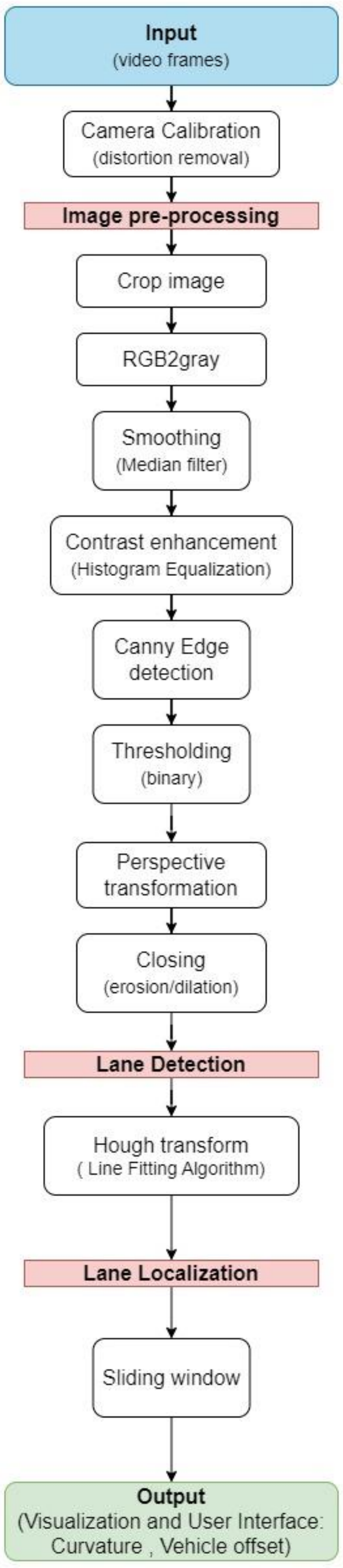
3) **Lane Detection**:

    - Hough Transform: Detect straight lines or line segments that may correspond to lanes.
    - Lane Model Fitting: Fit mathematical models (e.g., polynomials) to detected lane lines.

4) **Lane Localization**:
    Sliding Window Algorithm: Determine the position and curvature of detected lanes relative to the vehicle.

5) **Visualization and User Interface:**
    Display the processed images or video stream with lane markings and the values of the vehicle offset and the curvature value.

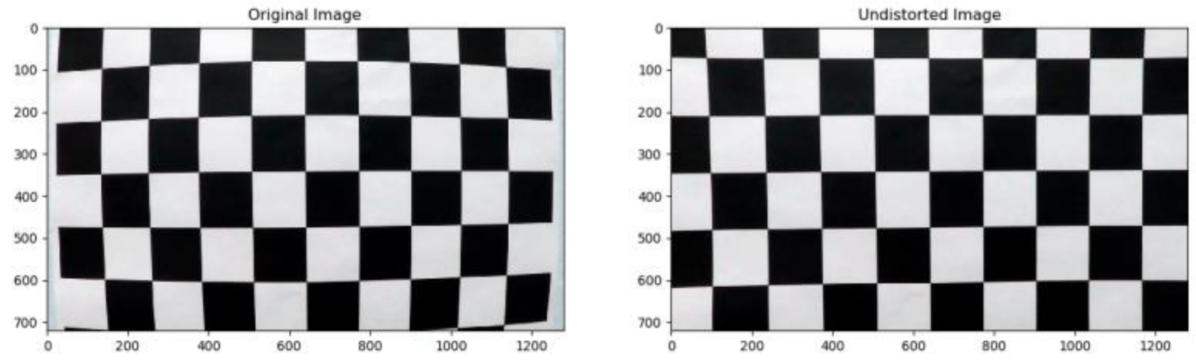
Block Diagram



Procedure Explanation

Camera Calibration:

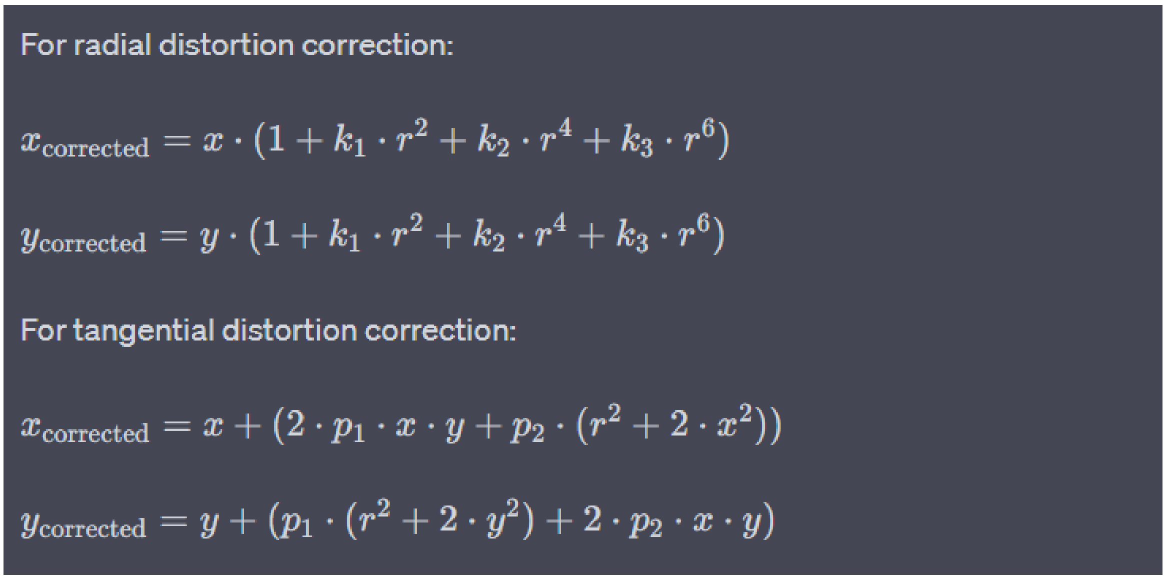
Vehicle-mounted cameras have increasingly utilized wide-angle lenses to expand their field of view, resulting in noticeable distortions in captured images. These distortions are typically categorized into radial and tangential distortions. Radial distortion affects the apparent shapes of lanes, vehicles, and the surrounding environment compared to the real-world scene. Conversely, tangential distortion arises when the camera lens is not perfectly aligned with the image plane of the visual sensor, leading to tilted images that can make the lanes, vehicles, and background appear closer or farther away than they are in the actual three-dimensional environment. Correcting these distortions is crucial for faithfully representing the true driving environment and ensuring the accuracy of image analysis and processing.



Implementation:

SciPy and NumPy:

Apply the equations for distortion correction, such as the Brown-Conrady model, to the image coordinates.



For radial distortion correction:

$$x_{\text{corrected}} = x \cdot (1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6)$$

$$y_{\text{corrected}} = y \cdot (1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6)$$

For tangential distortion correction:

$$x_{\text{corrected}} = x + (2 \cdot p_1 \cdot x \cdot y + p_2 \cdot (r^2 + 2 \cdot x^2))$$

$$y_{\text{corrected}} = y + (p_1 \cdot (r^2 + 2 \cdot y^2) + 2 \cdot p_2 \cdot x \cdot y)$$

OpenCV: For remapping the image to undistorted image.

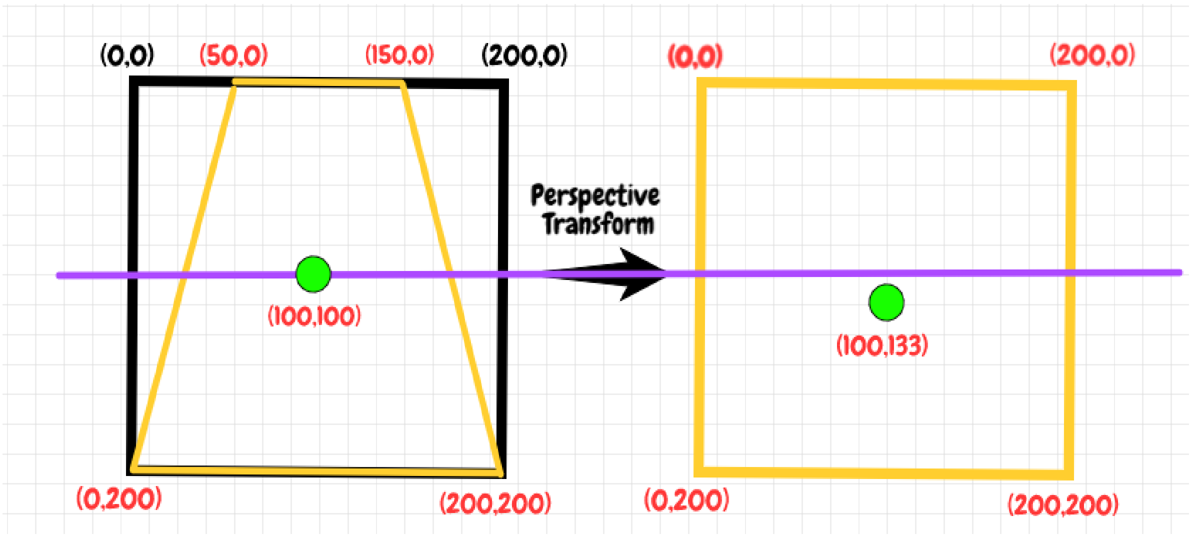
Region of interest:

Lane detection typically focuses on the area of the image where the lanes are expected to appear. Cropping allows you to define a specific region of interest (ROI) within the image, reducing the processing workload and improving detection accuracy. You can exclude irrelevant parts of the image, such as the sky or surrounding landscape.

Implementation: From scratch using SciPy and NumPy.

Perspective transformation:

Perspective transformation is a key technique in computer vision, used to convert a camera's view of the road into a top-down perspective. This transformation helps map objects in the image to their positions as if viewed from above. It's crucial for tasks like lane detection and traffic analysis, offering a clearer and more consistent view for accurate decision-making in applications such as autonomous driving and road monitoring.



Implementation:

Using OpenCV: “cv2.warpPerspective” function.

Hough transform:

A technique which is used to detect curves with specific shapes (circles, ellipses, lines, etc). The Hough Transform works by converting the Cartesian coordinate space of an image into a parameter space, known as the Hough space. In this space, each point represents a possible line in the original image. By applying a voting scheme, the algorithm identifies the lines with the highest number of votes, indicating their presence in the image. We use it to get the best fitting line of road lines after images preprocessing steps.

Implementation: from scratch

Sliding window Algorithm:

The sliding window approach involves dividing the region of interest in an image (typically the area where the lanes are expected to be) into a series of horizontal slices or windows. Each window represents a portion of the image's height, and these windows are stacked from the bottom of the image to the top. The goal is to determine which parts of each window contain lane markings.

The general steps we will follow are:

- Starting from the base of the image, a sliding window is placed around the position of the peak from the histogram. The window's width can be fixed or adaptive, depending on the algorithm we will choose. The algorithm searches for lane markings within the window.
- After processing one window, the algorithm may adjust the position of the next window based on the detected lane markings. This update helps to follow the curvature of the lanes.
- The sliding window search is repeated as it moves up the image, iteratively updating the window positions based on the detected lane markings.

Implementation: manually implemented using NumPy

Expected Deliverables

The following sections will be delivered by the Lane and Curve Detection project:

- A strong and reliable algorithm capable of recognizing and tracking lanes and curves in real-time.

- Software Implementation: A software program or library that combines the established algorithm and offers an interface for entering video or image data as well as displaying the discovered lane and curve information.

- Evaluation and Validation: A thorough assessment of the system's performance utilizing benchmark datasets or real-world scenarios. The system's accuracy and robustness will be evaluated by comparing it to ground truth data.

- Documentation and User Guide: Detailed documentation covering implementation details, algorithms employed, and program usage instructions.

Needed Libraries:

- SciPy.
- NumPy.
- OpenCV.
- Matplotlib
- Scikit-Image (skimage).

References:

1. https://www.researchgate.net/publication/334571033_Lane_Detection_Algorithm_for_Intelligent_Vehicles_in_Complex_Road_Conditions_and_Dynamic_Environments
2. https://doi.org/10.1051/itmconf/20214003011
3. https://www.researchgate.net/publication/368342321_Real_Time_Road_Lane_Detection_using_Computer_Vision_Techniques_in_Python