INTERNATIONAL JOURNAL OF
ARTIFICIAL INTELLIGENCE AND EMERGING TECHNOLOGY

Print ISSN 2735-4792

VOLUME 6, ISSUE 1, 2023, 1 – 14.

Online ISSN 2735-4806

Extraction of Features from a Bird's-Eye View for Path Detection

Abdelmoty Ahmed[1], Ahmed ElKattan[2, 3], Ahmed Omar[4], and Hammam Abdelaal[5]

[1]College of Computer Science, Nahda University in Beni Suef (NUB)
2Surrey Business School, University of Surrey, Guildford, UK
3Faculty of Economics and Business, Complutense University of Madrid, Madrid, Spain
4Department of Computer Science, Faculty of Science, Minia University, Egypt
[5]Department of Information Technology, Faculty of Computers and Information, Luxor University

Abstract:

Path detection has emerged as a complex challenge that has garnered significant attention within the computer vision community for a considerable span. The foundational problem of recognizing routes has evolved into a substantial impediment in the realm of computer vision. While a variety of Deep Learning methods have been employed for route identification, the focus has often centered on feature generation. Nonetheless, advanced Deep Learning techniques have exhibited proficiency in discerning these attributes. Yet, the complete implementation of these approaches to efficiently and accurately identify lanes is still pending. In this paper, we introduce an innovative perspective to address this issue. A distinctive approach to route preprocessing, coupled with a bird's-eye view, is presented. The primary aim involves the selection of the bird's-eye view area based on the processed image. Subsequently, an HSL (hue, saturation, and lightness) color transformation is applied to extract white features, along with a unique preliminary edge feature detection method. The lane is then identified using this new preprocessing strategy. It is noteworthy that numerous prior works have drawn upon the same standard datasets.

1. Introduction

There has been significant discussion surrounding autonomous vehicles. Contemporary academia and industry alike have placed a strong focus on achieving full autonomy in driving. The ultimate goal is to achieve a comprehensive understanding of the vehicle's surroundings. Among the vital aspects of environmental perception, camera-based lane recognition plays a pivotal role by aiding the vehicle in staying properly positioned within traffic lanes. Autonomous vehicles primarily operate through three key functions: perception, decision, and action. Put simply, perception involves observing and understanding the situation at hand, decision entails determining whether to turn left, right, or proceed straight, and action encompasses the execution of those chosen maneuvers. Due to rapid societal progress, automobiles have swiftly become a ubiquitous mode of transportation worldwide. This has led to an increasing number of diverse vehicles navigating congested roads [1]. Consequently, the frequency of car accidents resulting in injuries or fatalities has risen due to the surge in road-going vehicles. This has brought to the forefront the critical issue of ensuring safety while driving in environments characterized by diverse vehicles and crowded streets. Advanced driver assistance systems

(ADAS), encompassing features like lane departure warning [3], lane keeping assist, and adaptive cruise control, offer individuals a means to assess their driving surroundings. These systems provide timely feedback for safe driving and alert drivers to potentially risky situations. It is expected that such supplementary driving systems will continue evolving in complexity [5].

Upon investigation, it becomes evident that in complex traffic scenarios featuring high vehicle density and elevated speeds, the risk of accidents is significantly heightened [1]. In such circumstances, the extraction of road colors, detection of textures, identification of road boundaries, and delineation of lanes emerge as pivotal cues for human drivers.

Lane Detection holds a crucial role within the Lane Departure Warning (LDW) system. The process of lane detection can be broken down into two primary components:

1. **Edge Detection:** This involves identifying the edges within the image, which is a fundamental step in detecting lane boundaries.
2. **Line Detection:** Following edge detection, the next step is to detect and characterize the actual lane lines.

By dividing the process into these two core components, the system can effectively identify the lanes and provide the necessary warnings or assistance to the driver. Despite the application of filters, certain instances of incorrect edge detections may persist. To address this, Wang introduced a canny edge detection algorithm to enhance the accuracy of edge detection. This algorithm offers an improved understanding of intricate environments. Notably, an updated version of this algorithm was introduced in 2014, enabling it to effectively handle noisy environments [13]. The two primary algorithms widely employed for edge detection are Sobel and Canny. Line detection stands out as a pivotal aspect within lane detection. Two distinct methods for detecting lines are noteworthy:

1. Niu employed a modified Hough Transform to extract lane segments and incorporated a density-based spatial application noise clustering algorithm for segmentation [14].
2. Mammeri utilized a progressive probabilistic Hough Transform in conjunction with maximum stable extreme area to identify lane lines. Furthermore, a Kalman filter was integrated to enable continuous tracking of lane lines [15].

This paper introduces a novel Path Detection method that demonstrates adaptability across a wide array of traffic scenarios. The approach begins with preprocessing the image to enhance its clarity and comprehensibility for the computational system. Following this, a warp perspective transformation is applied to focus on the region of interest within the preprocessed images. The preprocessing phase consists of several steps: first, the image is undistorted; then, a warp perspective transformation is applied. Subsequently, a canny edge filter and a color filter are employed on the undistorted image. This comprehensive preprocessing approach sets it apart from other methods, which typically employ fundamental operations such as thresholding, blurring, and gradient calculations. Through experimentation, this paper demonstrates that the proposed method yields notably improved results compared to the current state-of-the-art preprocessing techniques in the context of lane detection.

This paper introduces an innovative Path Detection technology aimed at enhancing the precision of real-time lane detection [16]. The lane detection module is divided into two distinct steps: image processing and matching of lane detection, the proposed system provides a simplified visual representation to facilitate ease of comprehension and implementation for readers. The process unfolds as follows:

1. **Frame Reading from Video Stream:** The initial step involves capturing frames from the video stream.
2. **Image Processing:** The subsequent phase delves into image processing. Notably, a distinct aspect that sets this paper apart from others lies in the preprocessing approach. Prior to edge detection and color filtering, the image is undistorted and subjected to a warp perspective transformation. This unique sequence enhances the effectiveness of the subsequent processes.

By visually encapsulating these steps, below figure 1 offers a clear overview of the proposed system's operational flow and sets the stage for a more accessible understanding and application of the paper's concepts. Moving forward, the subsequent step involves fitting a curve through the lines obtained earlier. This curve-fitting process contributes to refining the lane detection. Upon completing the curve fitting, the final stage involves merging the preprocessed image with the original image. This merger results in the production of the ultimate image, encompassing the enhanced lane markings from the preprocessing stage within the context of the original frame.

2. PROPOSED METHODS

This paper builds upon previous research and adopts a systematic approach. It begins by rectifying image distortion caused by camera lenses, a common necessity to accommodate substantial data within a confined frame. By undistorting the image, a clearer foundation for subsequent processing is established. Subsequently, a warp perspective transformation is applied, enabling an overhead view of the road environment. Edge features are then extracted through the application of a canny filter, while color features are captured via histogram analysis. Given the accident-prone nature of high-speed road sections, which are often characterized by straight or occasionally curved lanes [18], the paper emphasizes a dual approach. Combining color feature extraction with edge feature extraction, this approach strives to enhance lane recognition accuracy significantly.

A notable contribution of this paper lies in its preprocessing stage. A novel technique introduced is the application of a color transform using the HSL model. This transformation facilitates the extraction of distinct lane colors, accommodating a broader range of road conditions. The proposed method particularly focuses on capturing both yellow and white lane lines. This novel approach is followed by a conventional sequence of preprocessing steps. The integration of color and edge feature extraction yields an experimental improvement in lane detection accuracy. This paper's distinctiveness rests in its comprehensive preprocessing strategy, encompassing both color-based and edge-based feature extraction, to enhance the lane detection process.

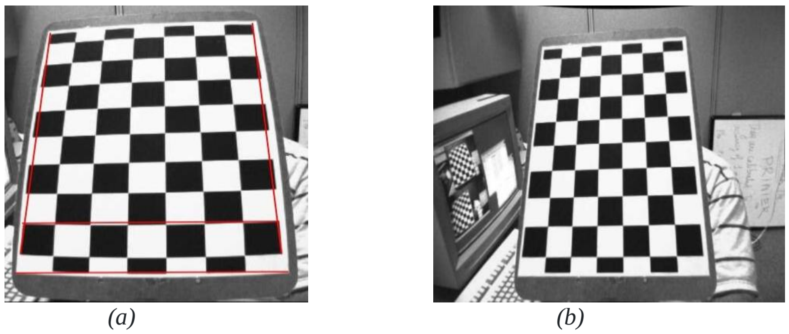

(a) *(b)*

Figure 2 (a) Distorted image capture by the camera with curved lines before calibration lines (b) undistorted image with straight lines after calibration

2.1 Preprocessing

The initial step in our process involves the distortion of images. Raw camera outputs often contain slight distortions, resulting from the utilization of lenses to encompass more of the surroundings within a limited frame. An example of this distortion is illustrated in image (a), wherein a type of distortion known as Barrel distortion causes straight lines to appear curved toward the outside edges. When transforming the image to obtain a bird's-eye view, these lines should ideally appear straight. However, due to the aforementioned distortion, they appear bowed outward. To rectify this, we need to perform distortion correction.

If A is the camera matrix

$$A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

The scaling factors for the transformation are defined as follows:

$$f_x = \frac{\mathrm{dsize.width}}{src.cols}$$
$$f_y = \frac{\mathrm{dsize.width}}{src.rows}$$

These scaling factors have significance in the context of mapping between 2D images:

- f_x and f_y represent the inverse mapping, converting from the destination (output) size to the source (input) size.
- g_x and g_y correspond to the forward mapping, transforming from the source (input) size to the destination (output) size.

These concepts collectively contribute to the process of transforming and correcting distortions in images, facilitating accurate depiction and analysis of visual data.

Then the distortion is corrected by

$$x_{corrected} = x(1 + k_1r^2 + k_2r^4 + k_3r^6)$$
$$y_{corrected} = y(1 + k_1r^2 + k_2r^4 + k_3r^6)$$

Certainly, when tangential distortions are present, corrective actions are taken to mitigate their impact. The formulae used for correction involve adjusting the distorted coordinates to obtain corrected values:
For the x-coordinate:

$$x_{corrected} = x + \left(2p_1xy + p_2(r^2 + 2x^2)\right)$$

And for the y-coordinate:

$$x_{corrected} = y + (p_1(r^2 + 2y^2) + 2p_2xy)$$

Here, x and y represent the initial distorted coordinates, r signifies the radial distance from the image center, and $p_1$and p_2 denote the tangential distortion coefficients. By applying these formulas, the distortion effects caused by tangential factors are effectively addressed.

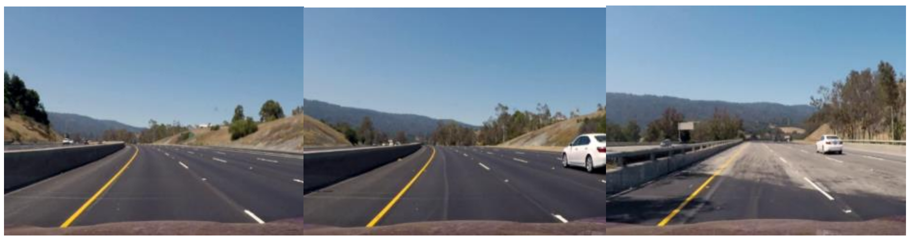


Figure 3 Undistortion under different conditions and different data

2.2. Warping Images

Warping images entails transforming them to present a "Bird's Eye View," essentially allowing us to observe images from a top-down perspective. This transformation is essential for tasks such as fitting curves and detecting lane lines. By obtaining a bird's-eye view, the process of identifying lane lines is made more intuitive. It's akin to looking down onto the road, which simplifies the lane detection process compared to analyzing the image from an on-road viewpoint.

To achieve this transformation, four source points are required from the original image. These points serve as reference markers for the transformation. Through this transformation, these points are reconfigured into straight lines in the transformed image, allowing for a comprehensive bird's-eye view of the road and facilitating accurate lane detection.

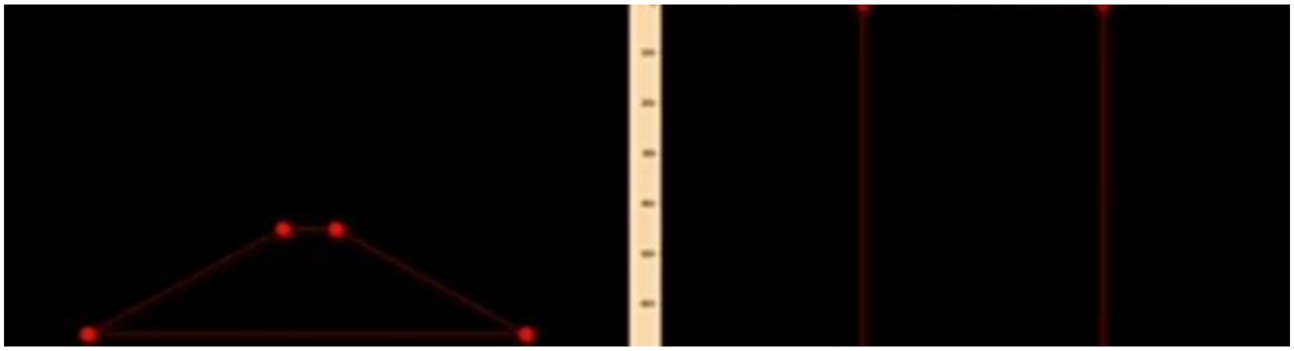


Figure 4 Four warping Source points that looks like going to infinity but when we see it from above those are straight lines

To facilitate this transformation, we can utilize the getPerspective() function, which provides a transformation matrix. This matrix is crucial for mapping the source points in the original image to their corresponding positions in the transformed "Bird's Eye View" image. The transformation matrix essentially guides the process of warping the image to

Print ISSN 2735-4792 VOLUME 6, ISSUE 1, 2023, 1 – 14. Online ISSN 2735-4806

achieve the desired top-down perspective.

$$dst(i) = (x_i^`, y_i^`), src(i) = (x_i, y_i), i = 0,1,2,3$$

dst(i) is the destination (transformed) image, and *src(i)* is the source (original) image.

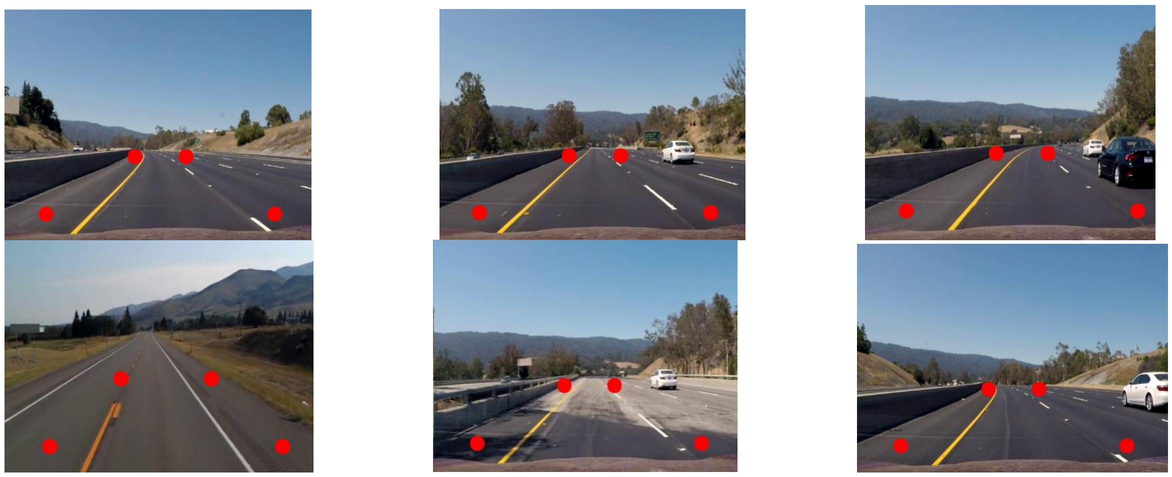


Figure 5 warp perspective under different conditions, curves and different data

After obtaining the transformation matrix, the next step involves applying the warpPerspective () function. This function takes three arguments: the input image (img), the transformation matrix (M), and the desired output image size (img_size). When this function is executed, it performs the perspective transformation based on the provided matrix. The resulting image is effectively transformed into the desired "Bird's Eye View" representation, aligned with the transformed points and dimensions defined by the matrix. This transformed image provides a top-down view of the scene, facilitating subsequent analysis, such as lane detection.

2.3. Finding Lane Lines

In our approach, we will explore two methods: color selection and edge detection. Starting with color selection, we delve into the representation of images within a computer. Analyzing the first ten pixels of an image reveals that each pixel is composed of three values: RGB (Red, Green, Blue) values. These values typically range from 0 to 255 as most images 8-bit, meaning they utilize 8 bits per color channel. As a result, the potential values span from 0 to 255. An image composed of entirely 255 values represents white, while an image consisting of all 0 values results in black. If we encounter a pixel with values like 255, 0, 0, the image takes on a red color. In this color-based approach, we can manipulate these RGB values to isolate specific colors and features of interest within the image. This technique forms the foundation of color selection for our lane detection process.

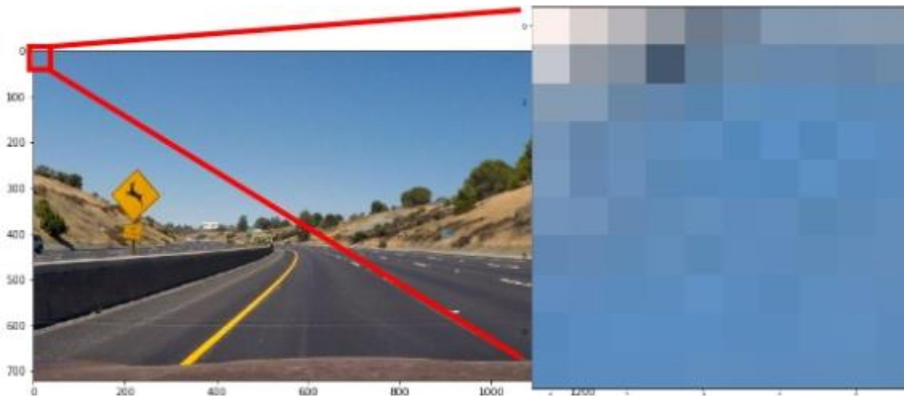


Figure 6

Indeed, beyond the Red, Green, and Blue (RGB) color space, there exist various other color spaces that provide different ways of representing colors. One such color space is HSL (Hue, Saturation, Lightness), which proves valuable in isolating specific color ranges for analysis. By transforming RGB colors into the HSL color space, we gain the ability to manipulate and isolate colors based on their hue, saturation, and lightness values. This offers a flexible approach for identifying specific colors or ranges of colors within an image, which can be particularly useful in tasks such as lane detection.

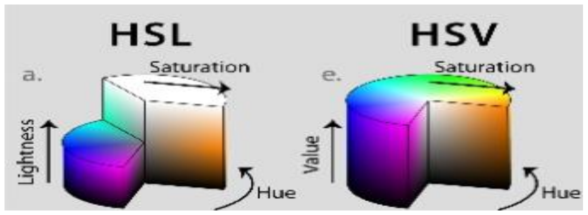


Figure 7

OpenCV simplifies the process of converting colors between different color spaces, including RGB and HSL. This is achieved through the cvtColor() function, where you can use cv2.COLOR_RGB2HLS as the conversion coefficient. This function allows you to convert an RGB image (or any other supported color representation) into the HLS (Hue, Lightness, Saturation) color space. The result of this conversion is an HLS image. While this image might appear slightly different, it essentially represents the same image using a distinct color space. This conversion enables you to work with colors in the HSL space and perform operations or analysis based on hue, lightness, and saturation values, offering a versatile approach to image processing and analysis.

- **Hue Layer:** This layer represents the color itself, with different hues corresponding to different colors around the color wheel.
- **Saturation Layer:** The saturation layer focuses on the intensity or vividness of the color. Higher values indicate more intense colors, while lower values indicate less vivid or even grayscale areas.
- **Lightness Layer:** This layer depicts the overall brightness of the image. It ranges from black (dark) to white (bright), with various shades of gray in between.

Isolating and manipulating these layers independently can help you achieve various image processing effects, identify specific colors or features, and enhance the overall analysis of the image, which can be particularly useful in lane detection and similar applications.

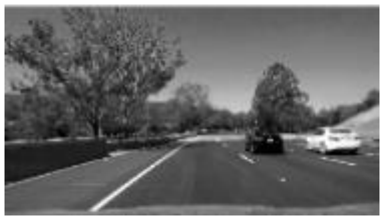
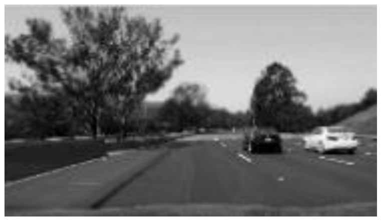

Red channel

Figure 8


blue channel

It's evident that the yellow line becomes distinguishable primarily in the Red and Green Channels, while its visibility is diminished in the Blue Channel due to the red and blue combination creating yellow. Shifting our focus to edge detection, OpenCV offers a couple of key functions for this purpose: Sobel and Canny operators. The Sobel operator calculates the derivative of a color channel. For instance, in the red channel, it identifies edges at transitions like red meeting white, computing the derivative along that line to yield a value corresponding to the slope of the transition. This information forms the basis of the resulting plot.

2.4. Curve Fitting:

With the binary image now at our disposal, we're ready to employ an algorithm to detect the left and right lane lines, followed by curve fitting to these lines. There are two distinct methods for curve fitting. The first method can be applied to any image, while the second method requires a preceding fit but offers a notably faster execution. We'll primarily focus on utilizing the second method. In the first method, we begin by generating a histogram for the lower half of the image. Wherever this histogram exhibits peaks indicates the likely starting points of the lane lines. In this context, the y-value corresponds to the sum of pixels at a specific x-location. The two highest peaks in the histogram signify the locations where the respective lane lines are expected to originate.

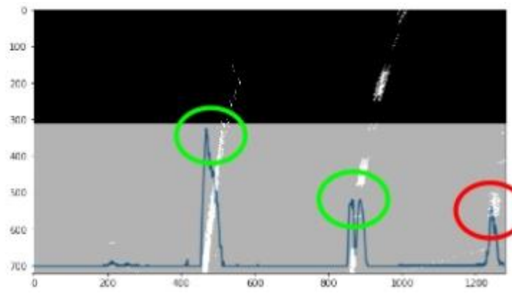


Figure 9 Histogram of the image

The red circle in the figure illustrates the subtle distinction between a potentially erroneous value and an accurate value. In this context, the lane to the right of the image is also detected as a line, underlining the importance of filtering out extraneous points to enhance the algorithm's accuracy. Eliminating these unnecessary points can significantly benefit the performance of this algorithm. Once we have identified the starting points for both the left and right lanes, we proceed by enclosing these points within boxes. Any white pixels from the binary image that fall within these boxes are collected into a consolidated list of x and y values. Moving forward, we superimpose another box atop the previous one and perform the same operation, effectively progressing along the lane. To determine the positioning of these boxes, we base their placement on the average position of the previous line. For instance, if the average shifts leftward, the subsequent box will also move leftward. This iterative approach ensures that the boxes align with the lane lines, contributing to the formation of these distinct lanes as depicted in the figure.

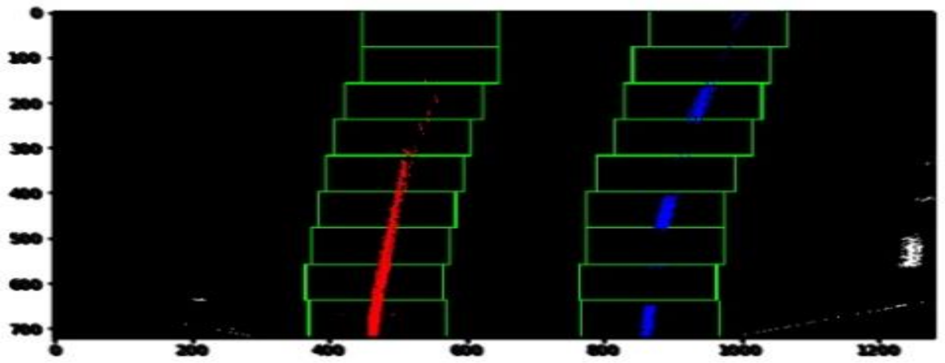


Figure 10

Upon completing this process, we amass a comprehensive list containing all the Y and X values of the pixels that constitute each lane line. To proceed with curve fitting, we utilize the `numpy` library's `polyfit()` function, applying it to these collected pixel values.

Print ISSN 2735-4792 VOLUME 6, ISSUE 1, 2023, 1 – 14. Online ISSN 2735-4806

For the left lane, we execute:

```
leftfit = np.polyfit(lefty, leftx, 2)
```

For the right lane, we utilize:

```
rightfit = np.polyfit(righty, rightx, 2)
```

Here, the value "2" signifies a second-order polynomial ($Ax^2 + Bx + c$) that characterizes the curve. The `polyfit()` function returns a fit array `[A, B, c]`, representing the coefficients of the polynomial equation.

This curve fitting process helps us define the estimated trajectories of the left and right lane lines based on the collected pixel coordinates, facilitating a more accurate depiction of the lane boundaries. The curve fit obtained through this process is represented in pixel space. To translate this into real-world measurements, we employ a conversion from pixels to meters. In the United States, lanes are often around three meters wide, although this might vary in different countries. The conversion allows us to establish the real-world dimensions of the lane boundaries.

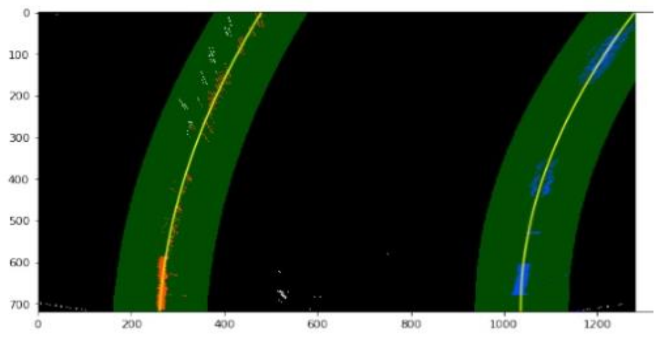


Figure 11

2.5. Final Image:

In the concluding phase, we craft a visually appealing image while also determining the radius of curvature. This crucial metric serves as a valuable tool for the vehicle, aiding in predicting the necessary turn tightness and steering adjustments needed to navigate curves effectively. By evaluating the radius of curvature, the vehicle gains insights into the curvature's severity, enabling it to make informed decisions about steering wheel inputs required to navigate curves with precision. This final step amalgamates lane detection outcomes, curvature radius computation, and other pertinent details, culminating in a comprehensive image that enhances the vehicle's ability to navigate safely and proficiently. The formula for calculating the radius of curvature is given by:

$$\text{Radius of curvature} = \frac{\left[1 + \left(\frac{dy}{dx}\right)^2\right]^{3/2}}{\left|\frac{d^2y}{dx^2}\right|}$$

By employing the previously obtained curve fits, we can readily calculate the radius of curvature. Additionally, we derive the distance to the center of the lane, which holds substantial utility. If the car begins to deviate towards the left, a simple control mechanism can be engaged. By maintaining a record of this deviation, the car can respond by slightly adjusting the steering wheel to the right, guiding it back towards the center of the lane.

To generate the visually appealing yellow lane depiction, we utilize the curve fits we've established. The polylines() function allows us to plot these curve fits onto the image, while the fillpoly() function is employed to fill the space between these lines with the yellow color. It's important to note that these operations are carried out in the warped space, which corresponds to the bird's-eye view.

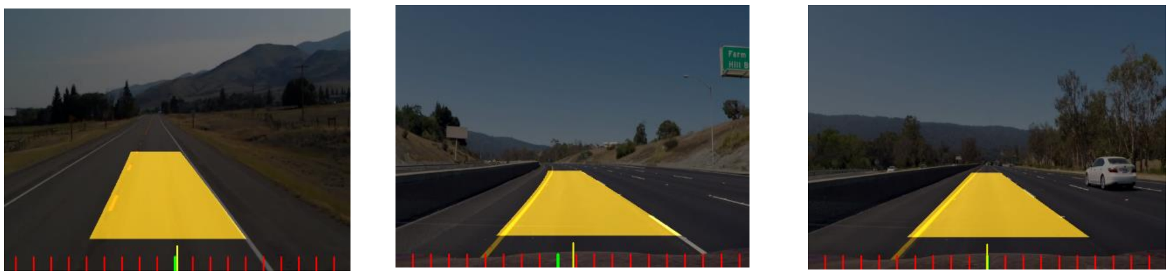


Figure 12 Paths detection under different light conditions

3. RESULTS AND DISCUSSIONSR

In contrast to the common approach employed by many research scholars, which involves selecting a region of interest and applying Hough transform or Kalman Filter primarily suited for detecting straight lanes, this paper introduces an innovative method capable of efficiently detecting both lanes and curves. Unlike the conventional techniques that often struggle with accurately identifying lane boundaries during turns, the proposed method stands out by bypassing the use of Hough transform while achieving enhanced efficiency and effectiveness in detecting lanes and curves. To ascertain the improved accuracy of the path detection process, the performance of the proposed method is evaluated using a correct detection rate. The evaluation is carried out on a publicly available dataset. The method's proficiency is demonstrated by conducting tests on sets of 100, 500, 1000, and 1500 images. Notably, the results depicted in the figure underscore the superiority of the proposed method, showcasing the highest correct detection rate in comparison to other previously proposed techniques. This approach heralds a significant advancement in lane detection, particularly in scenarios involving curves, and serves as a promising contribution to the field.

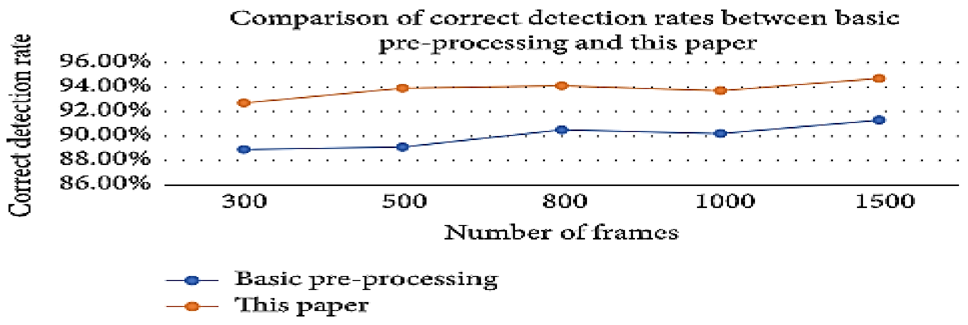


Figure 13

4. Conclusions

In conclusion, this paper has introduced a novel approach to path detection by leveraging preprocessing techniques and a Bird's Eye View methodology. The central concept revolves around the application of warp perspective prior to executing edge and color feature extractions, a strategy that markedly enhances the accuracy of lane detection. Additionally, we introduced a Histogram-based lane detection technique to identify both the left and right lanes. By incorporating these advancements, we have successfully minimized the impact of non-lane parameters, leading to a considerable improvement in the accuracy of path detection. This innovative combination of techniques has demonstrated its efficacy in effectively addressing the challenges associated with lane detection in various road scenarios, marking a significant step forward in the field. The proposed approach holds great promise for enhancing the capabilities of path detection systems and contributing to the advancement of autonomous driving technologies.

REFERENCES

1. D. Pomerleau, "RALPH: rapidly adapting lateral position handler," in Proceedings of the Intelligent Vehicles '95. Symposium, pp. 506–511, Detroit, MI, USA, 2003
2. J. Navarro, J. Deniel, E. Yousf, C. Jallais, M. Bueno, and A. Fort, "Influence of lane departure warnings onset and reliability on car drivers' behaviors," Applied Ergonomics, vol. 59, pp. 123–131, 2017.
3. P. N. Bhujbal and S. P. Narote, "Lane departure warning system based on Hough transform and Euclidean distance," in Proceedings of the 3rd International Conference on Image Information Processing, ICIIP 2015, pp. 370–373, India, December 2015
4. V. Gaikwad and S. Lokhande, "Lane Departure Identification for Advanced Driver Assistance," IEEE Transactions on Intelligent Transportation Systems, vol. 16, no. 2, pp. 910–918, 2015
5. H. Zhu, K.-V. Yuen, L. Mihaylova, and H. Leung, "Overview of Environment Perception for Intelligent Vehicles," IEEE Transactions on Intelligent Transportation Systems, vol. 18, no. 10, pp. 2584–2601, 2017.
6. F. Yuan, Z. Fang, S. Wu, Y. Yang, and Y. Fang, "Real-time image smoke detection using staircase searching-based dual threshold AdaBoost and dynamic analysis," IET Image Processing, vol. 9, no. 10, pp. 849–856, 2015.
7. P.-C.Wu, C.-Y. Chang, and C. H. Lin, "Lane-mark extraction for automobiles under complex conditions," Pattern Recognition, vol. 47, no. 8, pp. 2756–2767, 2014.
8. M.-C. Chuang, J.-N. Hwang, and K. Williams, "A feature learning and object recognition framework for underwater fish images," IEEE Transactions on Image Processing, vol. 25, no. 4, pp. 1862–1872, 2016.
9. Y. Saito, M. Itoh, and T. Inagaki, "Driver Assistance System with a Dual Control Scheme: Effectiveness of Identifying Driver Drowsiness and Preventing Lane Departure Accidents," IEEE Transactions on Human-Machine Systems, vol. 46, no. 5, pp. 660–671, 2016.
10. Q. Lin, Y. Han, and H. Hahn, "Real-Time Lane Departure Detection Based on Extended Edge-Linking Algorithm," in Proceedings of the 2010 Second International Conference on Computer Research and Development, pp. 725–730, Kuala Lumpur, Malaysia, May 2010
11. C. Mu and X. Ma, "Lane detection based on object segmentation and piecewise fitting," TELKOMNIKA Indonesian Journal of Electrical Engineering, vol. 12, no. 5, pp. 3491–3500, 2014.
12. J.-G. Wang, C.-J. Lin, and S.-M. Chen, "Applying fuzzy method to vision-based lane detection and departure warning system," Expert Systems with Applications, vol. 37, no. 1, pp. 113–126, 2010.
13. S. Srivastava, M. Lumb, and R. Singal, "Improved lane detection using hybrid median filter and modified Hough transform," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 4, no. 1, pp. 30–37, 2014.
14. J. Piao and H. Shin, "Robust hypothesis generation method using binary blob analysis for multi-lane detection," IET Image Processing, vol. 11, no. 12, pp. 1210–1218, 2017.
15. J. Niu, J. Lu, M. Xu, P. Lv, and X. Zhao, "Robust Lane Detection using Two-stage Feature Extraction with Curve Fitting," Pattern Recognition, vol. 59, pp. 225–233, 2015.
16. J. Son, H. Yoo, S. Kim, and K. Sohn, "Real-time illumination invariant lane detection for lane departure warning system," Expert Systems with Applications, vol. 42, no. 4, pp. 1816–1824, 2015.
17. A. Mammeri, A. Boukerche, and Z. Tang, "A real-time lane marking localization, tracking and communication system," Computer Communications, vol. 73, pp. 132–143, 2016.
18. C. J. Chen, B. Wu, W. H. Lin, C. C. Kao, and Y. H. Chen, "Mobile lane departure warning system in," in Proceedings of the 2009 IEEE 13th International Symposium on Consumer Electronics, pp. 1–5, 2009.
19. J. W. Lee, C. D. Kee, and U. K. Yi, "A new approach for lane departure identification," in Proceedings of the IEEE IV2003 Intelligent Vehicles Symposium, pp. 100–105, 2003.
20. J. W. Lee and U. K. Yi, "A lane-departure identification based on LBPE, Hough transform, and linear regression," Computer Vision and Image Understanding, vol. 99, no.