2023 3rd Asian Conference on Innovation in Technology (ASIANCON)
Pune, India. Aug 25-27, 2023

Lane Detection for Self-driving Cars using AI

[1st] Pallab Banerjee
Amity School of Engg. and Technology
Amity University Jharkhand,
Ranchi, INDIA
pbanerjee@rnc.amity.edu

[2nd] Kanika Thakur
Amity School of Engg. and Technology
Amity University Jharkhand,
Ranchi, INDIA
kthakur@rnc.amity.edu

[3rd] Ambikesh
Amity School of Engg. and Technology
Amity University Jharkhand,
Ranchi, INDIA
kumarambikesh509@gmail.com

[4th] Probal Banerjee
Govt. Polytechnic Ranchi
Jharkhand,INDIA
probalbanerjee17@gmail.com

***Abstract*—The Concern regarding the nature of accidents has grown as the frequency of traffic incidents has increased. It happens frequently as a result of human mistake. In order to assist drivers, lane detection systems are currently being created with the primary aim of identifying lanes and warning drivers about potential lane departures. There are multiple techniques for detecting lanes, with many focusing on straight lanes. This paper's main goal is to examine the drawbacks of various lane detecting algorithms and suggest a solution that does away with them. This work uses artificial intelligence to recognize lanes using OpenCV, NumPy, and digital image processing (noise removal, edge detection, Hough transformation).**

Keywords— Lane Detection, Artificial intelligence, OpenCV, NumPy, Hough Transform.

I. INTRODUCTION

A self-driving car is one that is capable without the aid of a driver, recognising its surroundings and navigating. At no point is a human passenger required to control the steering of the vehicle, and they are not even required to be inside it at all. A self-driving automobile can accomplish everything a qualified human driver can do and travel anywhere a regular car can. A driver is not required for autonomous vehicles to operate; they are virtually driverless. The fact that you can sleep in the driver's seat is the only distinction between them and a standard car.

Let us pretend for the purposes of comprehension that an autonomous car resembles your family PC but has wheels. Your computers include keyboards and mice as input devices, and you can give your computer commands using these tools.

Fig. 1. Autonomous Driving

Once the input is received, your computer uses the data to execute simple or complicated transformations before providing you with a relevant output.

Autonomous vehicles are no different in this regard. They gather information from outside environments via their input devices, such as cameras and sensors. subsequently the collection of data from external sources, they change the data and issue commands like accelerating, braking, steering to the left or right, etc. using exceptionally sophisticated machine learning techniques, comprising predictive models, computerized image recognition, neural networks, etc.

The on-board system is responsible for receiving the information from hardware components that are responsible for detecting environmental parameters. LIDARs, radars, sonars, and cameras have become the most prevalent varieties..

- The autonomous vehicle has the ability to perceive, hear, and move owing to hardware components.
- When contemplating whether to move, cease moving, slow down, or take another course of action, the software acts as the brain, digesting ambient input. Perceptual processing, strategic planning, and control are the three ways that vehicle autonomy software might be differentiated.

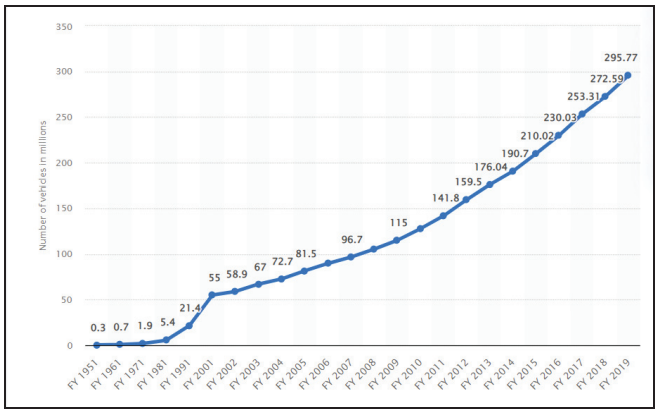

Fig. 2. Number of vehicles over the years

Let us now look over the advantages of using autonomous cars.

- Autonomous vehicle technology may be able to offer some benefits over human-driven vehicles. They

2023 3rd Asian Conference on Innovation in Technology (ASIANCON) | 979-8-3503-0228-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/ASIANCON58793.2023.10269847

might increase traffic safety, which is one such potential advantage.

- Another benefit of autonomous vehicles is that they may be able to reduce traffic congestion via fewer collisions.
- To do this, autonomous driving will do away with human actions that obstruct the road, such stop-and-go traffic.
- Another advantage of automatic driving is that it may allow those who are unable to drive due to age or disability to travel more conveniently.

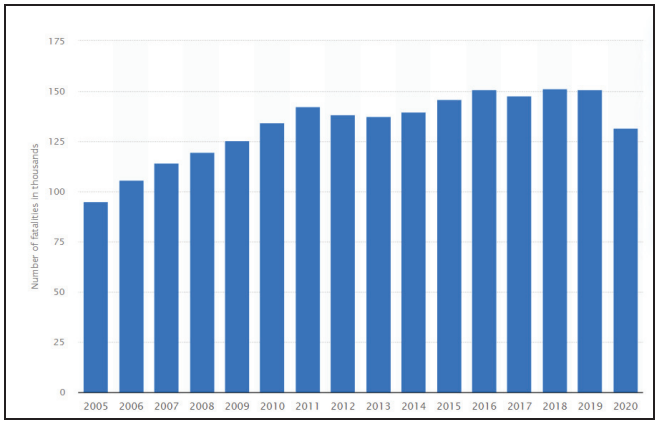

Fig. 3. Number of accidents per year

Artificial intelligence (AI) refers to systems or machines that are designed to perform tasks similar to human intelligence and can improve their performance based on the information they receive. There are several ways in which AI can be used, including:

- Chatbots that utilize AI to better understand customer inquiries and provide more efficient solutions.
- Intelligent assistants that extract important information from large, unstructured datasets to improve scheduling.
- Recommendation engines that suggest TV shows to users based on their viewing habits.

Fig. 4. Smart Cities

The field of artificial intelligence (AI) is not restricted to any particular format or purpose, but rather focuses on the methods, capability for advanced thinking, also processing data. Contrary to popular belief, artificial intelligence (AI) does not work in the way that highly intelligent, human-like robots function. Instead, its objective is to enhance human abilities and productivity, making it a valuable asset in commercial applications.

Artificial intelligence (AI) can be categorized into different types:

Artificial Narrow Intelligence (ANI), or weak AI, is engineered to carry out particular duties. The majority of the current artificial intelligence (AI) tools fit within this category. Weak AI, regardless of its appellation, may enable cutting-edge instructional programs like Apple's Siri, Amazon's Alexa, IBM Watson, and driverless vehicles.

Artificial Super Intelligence (ASI) and Artificial General Intelligence (AGI) make up strong AI (ASI). The notion of "artificial general intelligence" (AGI) depicts a machine with intelligence on accord with its fellow humans, including the capacity for consciousness of oneself, learning, and anticipating future events. Contrarily, artificial super intelligence is more clever and powerful in lieu of the human brain.

Instantaneous initiatives frequently use OpenCV, a toolkit that is open-source for neural networks, image processing, and machine learning. It processes pictures and movies to identify subjects, individuals, and perhaps handwriting in people. When Python is used in conjunction with libraries like NumPy, it is possible to inspect the OpenCV array structure and execute computational procedures in vector space to discover an image pattern and its attributes.

Because it is offered under a BSD licence and is free for both academic and commercial use, OpenCV was initially made available in version 1.0. In addition to being interoperable with Windows, Linux, Mac OS, iOS, and Android, it provides customer interfaces for C++, C, Python, and Java. The development of OpenCV placed a high premium on real-time efficiency, and the program as a whole is written in streamlined C/C++ code that makes use of multi-core handling.

A broad spectrum of sophisticated mathematical computations can be performed on massive multi-faceted data structures such as matrices and arrays using the NumPy Python module. Travis Oliphant assembled NumPy in 2005, expanding upon the Numeric package from the year prior. The "ndarray" information structure, in contrast to Python's embedded list data structure, provides the foundation of NumPy's capabilities and is seamlessly typed.

Ndarrays are compatible with existing quantitative frameworks like BLAS and LAPACK since they can be storage-mapped and inserted into memory reservoirs created by C/C++, Python, and Fortran manifestations to the C Python engine. Many of the aforementioned libraries are included in the SciPy package, which makes use of this function.

The next section will provide a literature review of existing lane detection techniques from reputable publishers and researchers. The goal of this survey will be to understand the challenges they faced and the limitations of their systems.

The proposed system architecture and implementation will follow the literature survey.

II. LITERATURE SURVEY

In their study, Jae-Hyun Cho and colleagues [1] utilized the Hough transform on four regions of interest (ROIs) simultaneously, optimizing the accumulator cells to achieve successful lane detection. Although the Hough transform can only identify straight lines, the authors were able to improve lane recognition rates on curved roads.

Chan Yee Low and colleagues [2] developed a reliable technique for detecting left and right lane markers by optimizing fostering the edge recognition and Hough transform methods. Specific characteristics are uncovered using canny edge detection, and then lanes generate themselves by employing the Hough transform. In addition, by scalarizing the image's domain of interest, the Hough transform has a job to find pertinent lines that might act as either the right or left lane margins.

Hongli Fan and Weihua Wang [3] proposed a novel approach to edge detection in color road images by converting the original color information from the RGB model utilizing a new imaging approach and an in-depth colour model to determine the deviations between the grayscale image from the L stream and the red-green image. This is followed by edge detection, resulting in improved edge retention and noise resistance compared to established techniques for color road image edge recognition.

Dajun Ding and co-authors [4] proposed a technique for identifying road regions based on vanishing points and line sections. To lessen the amount of calculation, the suggested strategy comprises looking at the input photographs in a region of interest. The Hough transform is used to pinpoint lines in each frame, and each frame's software-generated road area significance is then determined. The method is effective on a variety of roads.

Wang Jian and colleagues [5] developed a method for accurately extracting road regions based on regional growth. However, the method's accuracy depends on the correct selection of seed points; otherwise, the lane identification can fail and result in interference data. To mitigate the impact of unnecessary information while identifying lanes, an area threshold value is added to restrict the scanned zone's growth. The lane detection and departure warning algorithms exhibit positive pace and frequency of recognizing experimental findings.

N. Phaneendra et al. [6] proposed a lane detection approach that involved pre-processing of the images, binary processing, dynamic threshold selection, and fitting of the Model of the Hough transform. In lieu of using the Hough transform, the Kalman filter was put in place to enhance lane detection performance. The separation between the driving path and the underside centre of the collected visual point was used to propose leaving the lane. The proposed solution demonstrated effectiveness and workability in experiments.

F. Mariut [7] proposed a formula for comprehending pavement markings and their characteristics in order to predict the direction of travel. In order to identify lane marks accurately, a technique for acquiring the inner edge of the lane was developed. The Hough Transform was put to work to discover lines in photographs.

Kamarul Ghazali and colleagues [8] stipulated a method for assessing unforeseen lane shifts based on the H-maxima and a more sophisticated Hough Transform. After identifying the vicinity of concern, the source picture is separated into close and far angles of view. The Hough Transform is used to detect lane markings in the close-up field of sight after suppression of noise. Straight transit routes are where the algorithm performs best.

Yong Chen with Mingyi He [9] suggested the severely curled lane lane boundaries projective model (LBPM) algorithm. The approach determines the highest posterior probability for a given lane using particle swarm tuning and the likelihood of occurrence for a given lane using the lane model. The lane model is used to calculate the lane borders and topological structure. Assessments demonstrate that this method is effective on lanes with severe curves, however it only acknowledges the host lane.

Zhiyuan Xu and co-authors [10] suggested leveraging contrast limiting adaptive histogram equalisation to reduce the impact of fog. (CLAHE). This technique reduces image noise while boosting contrast by clipping the histogram's greatest possible value and distributing the snipped pixel arrays among every grey level.

Mustafa Surti and colleagues [11] implemented an embedded advancement board, such as the Raspberry Pi, with a lane detection system. In both still photos and moving pictures, lanes are identified using the Hough transform in combination with clever edge detection. The Hough transform is made more effective to increase accuracy by adding a portion of focus functionality before execution. On a Raspberry Pi, the algorithm undergoes scrutinized in right away[12].

III. PROPOSED SYSTEM

The proposed system focuses on detecting the lanes for self-driving cars utilizing concepts of Artificial Intelligence, Open-Source Computer Vison Library and Digital Image processing. Working flowchart, methodology, observations, and results will be discussed in the upcoming sections.

A. Sequential Diagram

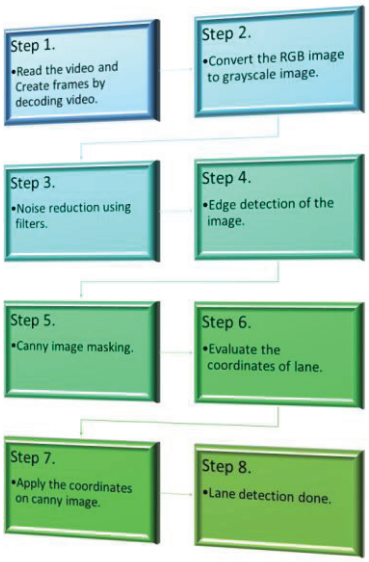

Fig. 5. Steps involved in lane detection

B. *Approach*

Prior to begin the work, primarily, literature reviews are carried out through existing papers and trusted materials to understand the various ways of detecting lanes and challenges they faced. Finally, proposing a system which considers following points:

Video acquisition and decoding: The Video Capture object will be used to capture the video frames. Each captured frame will then be decoded or converted into a sequence of images.

RGB to Grayscale conversion: As the captured frames are in RGB format, we need to convert them to grayscale. To do this, we add the red, green, and blue values of each pixel and divide the sum by 3. This step eliminates colour redundancy and produces a more concise image representation.

Noise Reduction: Digital images often contain noise, which can introduce errors in pixel values. To nullify the noise and make it easier to work with the image, we convolve it with a Gaussian filter.

Edge detection using Canny: By using Canny filter, we can extract meaningful structural information from vision objects while drastically reducing the amount of data that needs to be processed.

Region of Interest Evaluation: We will only consider the area covered by the road lane, so we create a mask with the same dimensions as the road image. We carry out a bitwise AND operation between each pixel of our Canny image and this mask. Finally, we hide the Canny image and reveal the area of interest outlined by the polygonal shape of the mask.

Hough Line Transformation: We can find, employing the technique of the Hough Line Transform, our image exhibits uninterrupted lines.. For this project, we will use the probabilistic Hough Line Transform, which outputs the extremes of the identified lines.

C. *Implementation*

These are the libraries have been used in my project for detecting the boundary lanes of a road for self-driving cars.

Finding_Lane_Lines_Code.py

C: > Users > ambik > OneDrive > Desktop > Lane Detection > Finding_Lane_Lines_Code.py >

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

Fig. 6. Libraries

Creating points on the boundary lanes for formulation of straight-line equation.

```
def make_points(image, line):
    slope, intercept = line
    y1 = int(image.shape[0])# bottom of the image
    y2 = int(y1*3/5)
    x1 = int((y1 - intercept)/slope)
    x2 = int((y2 - intercept)/slope)
    return [[x1, y1, x2, y2]]
```

Fig. 7. Make Point Function

Creating two lines in each frame over the detected lanes by aggregating slope intercept.

```
def average_slope_intercept(image, lines):
    left_fit    = []
    right_fit   = []
    if lines is None:
        return None
    for line in lines:
        for x2, y2, x1, y1 in line:
            fit = np.polyfit((x1,x2), (y1,y2), 1)
            slope = fit[0]
            intercept = fit[1]
            if slope < 0: # mirror image of y
                left_fit.append((slope, intercept))
            else:
                right_fit.append((slope, intercept))

    if len(left_fit) and len(right_fit):
    ##over-simplified if statement (should give an idea of why the error occurs)
        left_fit_average  = np.average(left_fit, axis=0)
        right_fit_average = np.average(right_fit, axis=0)
        left_line  = make_points(image, left_fit_average)
        right_line = make_points(image, right_fit_average)
        averaged_lines = [left_line, right_line]
        return averaged_lines
```

Fig. 8. Avg Slope Intercept Function

Applying canny filter to the image created by single frame of the video.

```
#Applying canny filter to the image
def canny(img):
    gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
    kernel = 5
    blur = cv2.GaussianBlur(gray,(kernel, kernel),0)
    canny = cv2.Canny(gray, 50, 150)
    return canny
```

Fig. 9. Canny Filter

Displaying lines on the detected road boundary lanes.

```
#Function for displaying Lines on the detected lanes
def display_lines(img,lines):
    line_image = np.zeros_like(img)
    if lines is not None:
        for line in lines:
            for x2, y2, x1, y1 in line:
                cv2.line(line_image,(x1,y1),(x2,y2),(255,0,0),10)
    return line_image
```

Fig. 10. Display Lines

Extracting region of interest after removing irrelevant part of the road image by simply masking and filtering the required portion of image.

```
#Function for extracting the region of interest from the entire road image
def region_of_interest(canny):
    height = canny.shape[0]
    width = canny.shape[1]
    mask = np.zeros_like(canny)

    triangle = np.array([[
    (200, height),
    (550, 250),
    (1100, height),]], np.int32)

    cv2.fillPoly(mask, triangle, 255)
    masked_image = cv2.bitwise_and(canny, mask)
    return masked_image
```

Fig. 11. Extracting Region of Interest

Applying all the function to the captured video and processing each frame to detect lanes.

```
#Applying lane detection to the video
vcap = cv2.VideoCapture("original.mp4")
while(vcap.isOpened()):
    ret, frame = vcap.read()
    if ret == True:
        canny_image = canny(frame)
        canny_cropped = region_of_interest(canny_image)
        lines = cv2.HoughLinesP(canny_cropped, 2, np.pi/180, 100, np.array([]), minLineLength=40,maxLineGap=5)
        averaged_lines = average_slope_intercept(frame, lines)
        line_image = display_lines(frame, averaged_lines)
        img_combi = cv2.addWeighted(frame, 0.8, line_image, 1, 1)
        cv2.imshow("Lane Detection", img_combi)
        #Holds the output window for 10 seconds or untill 'q' is pressed
        if cv2.waitKey(10) & 0xFF == ord('q'):
            break
    else:
        break
vcap.release()
cv2.destroyAllWindows()
```

Fig. 12. Consolidating All Functions

Single frame of input video.

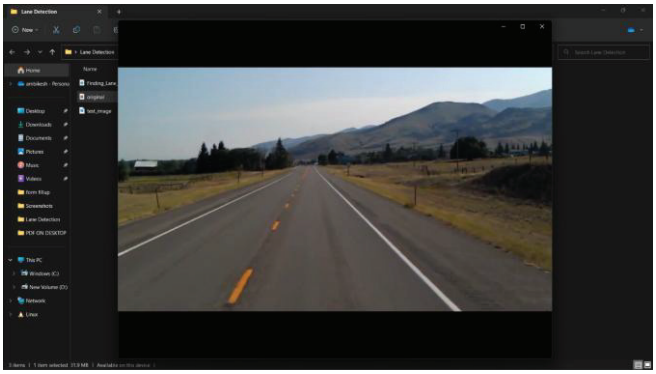

Fig. 13. Input Frame

D. Results

Firstly, video is captured and decoded in frames after that digital image processing is done like gray-scaling, reduction of noise, image filtering, edge detection and fitting the canny image according to coordinates. Finally, the lane is detected which can be seen in following image where detected lanes are covered by bluish white lines. The implementation showed better efficiency in evaluation of region of interest and hence lane detection from many existing systems and papers.

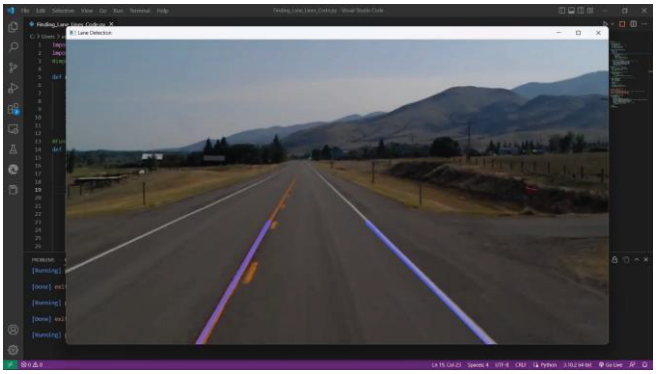

Fig. 14. Output Frame

IV. CONCLUSION

Now, it is the time to discuss about the findings and performance of the system implemented earlier in previous sections.

The project's objective is to find lanes. When the popularity of automobiles with autonomous driving is growing every day, lane detection is a blessing. People can locate lanes pretty simply, but computers find it difficult to do so in a variety of weather situations, such as snowy mornings, wet evenings, or even at night. This paper suggests a model in which firstly video is captured and decoded in frames after that digital image processing is done like Gray-scaling, reduction of noise, image filtering, edge detection and fitting the canny image according to coordinates. Finally, the lane is detected which can be seen in aforementioned implementation. The results of implementation showed better efficiency in evaluation of region of interest and hence lane detection from many existing systems and papers.

V. FUTURE WORKS

- Obstacle detection and accident avoidance: The exact and precise detection of impediments as well as the automobile's track is one of the most crucial aspects of self-driving automobiles. The suggested design will be improved by the ability of the vehicle or car to precisely and promptly identify the existence of an obstruction, enabling it to stop at an adequate distance and avoid a collision.
- Lane Keep Assistance: A control system known as a lane keeping assist (LKA) system helps a vehicle maintain safe travel within a highway's indicated lane. Without further input from the driver, the Lane Keep Assist system recognizes when the car veers from a lane and automatically corrects the steering to get it back on course. The proposed system can be improved by adding this smart module.

REFERENCES

[1] Cho, Jae-Hyun, Young-Min Jang, and Sang-Bock Cho. "Lane recognition algorithm using the Hough transform with applied accumulator cells in multichannel ROI." In Consumer Electronics (ISCE 2014), The 18th IEEE International Symposium on, pp. 1- 3. IEEE, 2014.

[2] Low, Chan Yee, Hairi Zamzuri, and Saiful Amri Mazlan. "Simple Robust Road Lane Detection Algorithm". IEEE, 2014.

[3] Fani, Hongli, and Weihua Wang. "Edge Detection of Colour Road Image Based on Lab Model." In Computational and Information Sciences (ICCIS), 2013 Fifth International Conference on, pp. 298-301. IEEE, 2013.

[4] Ding, Dajun, Chanho Lee, and Kwang-yeob Lee. "An adaptive road ROI determination algorithm for lane detection." In TENCON 2013-2013 IEEE Region 10 Conference (31194), pp. 1-4. IEEE, 2013.

[5] Jian, Wang, Sun Sisi, Gong Jingchao, and Cao Yu. "Research of lane detection and recognition technology based on morphology feature." In Control and Decision Conference (CCDC), 2013 25th Chinese, pp. 3827-3831. IEEE, 2013.

[6] Nalla, Phaneendra, GCL AbhiramanGoud, and V. Padmaja. "ACCIDENT AVOIDING SYSTEM USING LANE DETECTION." IJRECE 1, no. 1 (2013): 01-04.

[7] Mariut, F., C. Fosalau, and D. Petrisor. "Lane mark detection using Hough Transform" In Electrical and Power Engineering (EPE), 2012 International Conference and Exposition on, pp. 871-875. IEEE, 2012.

[8] Ghazali, Kamarul, Rui Xiao, and Jie Ma. "Road lane detection using H-maxima and improved hough transform." In Computational Intelligence, Modelling and Simulation (CIMSiM), 2012 Fourth International Conference on, pp. 205-208. IEEE, 2012.

[9] Banerjee, P., Tiwari, A., Kumar, B., Thakur, K., Singh, A., & Dehury, M. K. (2023, April). Task Scheduling in cloud using Heuristic Technique. In 2023 7th International Conference on Trends in Electronics and Informatics (ICOEI) (pp. 709-716). IEEE.

[10] Chen, Yong, and Mingyi He. "Sharp curve lane boundaries projective model and detection." In Industrial Informatics (INDIN), 2012 10th IEEE International Conference on, pp. 1188-1193. IEEE, 2012.

[11] Xu, Zhiyuan, Xiaoming Liu, and Na Ji. "Fog removal from color images using contrast limited adaptive histogram equalization." In Image and Signal Processing, 2009. CISP'09. 2nd International Congress on, pp. 1-5. IEEE, 2009.

[12] Mr. Mustafa Surti, Prof. (Dr.) Bharati Chourasia "Real time lane detection system using python and OpenCV on Raspberry pi" International Journal for Research in Engineering and Application (IJREAM) ISSN: 2454-9150 Vol-05, Issue-06, Sep 2019.