Ubiquitous Computing
Midsem Project Evaluation

About Our Project

This project involves extracting data, such as speed, fuel level, time, distance travelled, from car dashboard images. Using image processing techniques in Python, the system analyzes and retrieves the necessary information. The extracted data is then stored in a MongoDB database and displayed on a website, allowing user to view the most recent data.

Speedometer Speed Detection

Overview :

- Convert image to grayscale
- Apply Gaussian blur
- Apply Canny edge detection
- Use HoughLinesP to find lines
- Identify the longest line
- Calculate angle to determine speed

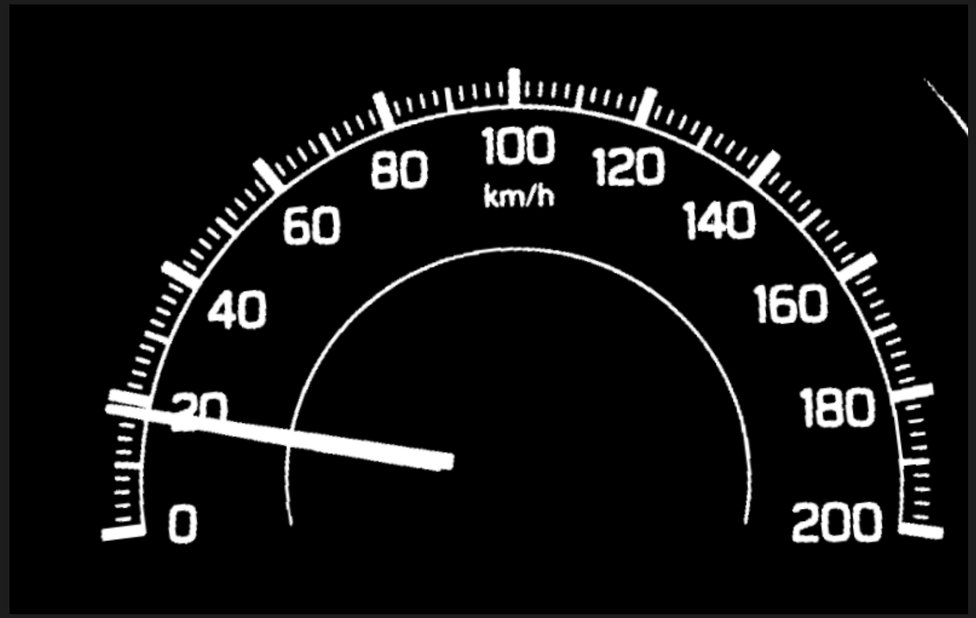
Image Processing & Speed Calculation

Image processing

- **Grayscale Conversion:** Simplifies the image
- **Gaussian Blur:** Reduces noise and smoothens the image
- **Canny Edge Detection**: Identifies edges in the image
- **HoughLinesP**: Detects line segments in the edge image

Determining the Speed

- Identify the longest line (likely the needle)
- Calculate the angle of this line
- Map the angle to a calibrated speed range
- Output the determined speed



Time & Distance Extraction

Overview :

- Convert image to grayscale
- Normalize the image
- Apply Gaussian blur
- Apply thresholding
- Define region of interest (ROI)
- Perform OCR using Pytesseract

Image Processing

- **Grayscale Conversion:** Simplifies the image by removing color information, reducing complexity for subsequent steps.
- **Normalization**: Adjusts the range of pixel intensity values, improving contrast and preparing the image for thresholding.
- **Gaussian Blur**: Reduces noise and smoothens the image, helping to eliminate small imperfections that might interfere with OCR.
- **Thresholding**: Converts the image to binary (black and white), isolating the text from the background.
- **Region of Interest (ROI)**: Focuses the processing on specific areas where the time and distance information is located, reducing errors and improving OCR accuracy.

OCR - Optical Character Recognition

Extracting Time

- The OCR engine is focused on a specific area of the image where the time is displayed.
- OCR detects numbers and formats them as HH.
- Post-processing can be done to ensure that the extracted text adheres to valid time formats.

Extracting Distance

- OCR targets the odometer display on the dashboard.
- OCR recognizes and converts the odometer reading (usually displayed as kilometers or miles).
- The recognized value is saved as a numeric value representing the distance traveled.

Fuel Guage Extraction

Overview :

- Grayscale conversion
- Normalization
- Gaussian blur
- Thresholding
- Region of Interest (ROI)
- Applying threshold to ROI
- Pixel counting

Detailed Steps

- **Grayscale Conversion:** Simplifies the image by removing color information, reducing complexity for subsequent steps.
- **Normalization**: Adjusts the range of pixel intensity values, improving contrast and preparing the image for thresholding.
- **Gaussian Blur**: Reduces noise and smoothens the image, helping to eliminate small imperfections that might interfere with OCR.
- **Thresholding**: Converts the image to binary (black and white), isolating the text from the background.
- **Region of Interest (ROI)**: Focuses the processing on specific areas where the time and distance information is located, reducing errors and improving OCR accuracy.

Detailed Steps

- **Apply Threshold to ROI :** Use threshold determined earlier on ROI, further isolates fuel gauge indicator.
- **Pixel Counting :**

  Use cv2.countNonZero() function (Count white (fuel) and black (empty) pixels.)

  Provides basis for fuel level calculation.
- **Calculate Fuel Percentage :**

  Determine ratio of white pixels to total pixels.

  Convert ratio to percentage.

  It represents current fuel level

User access to data

https://catkin-magenta-path.glitch.me/new

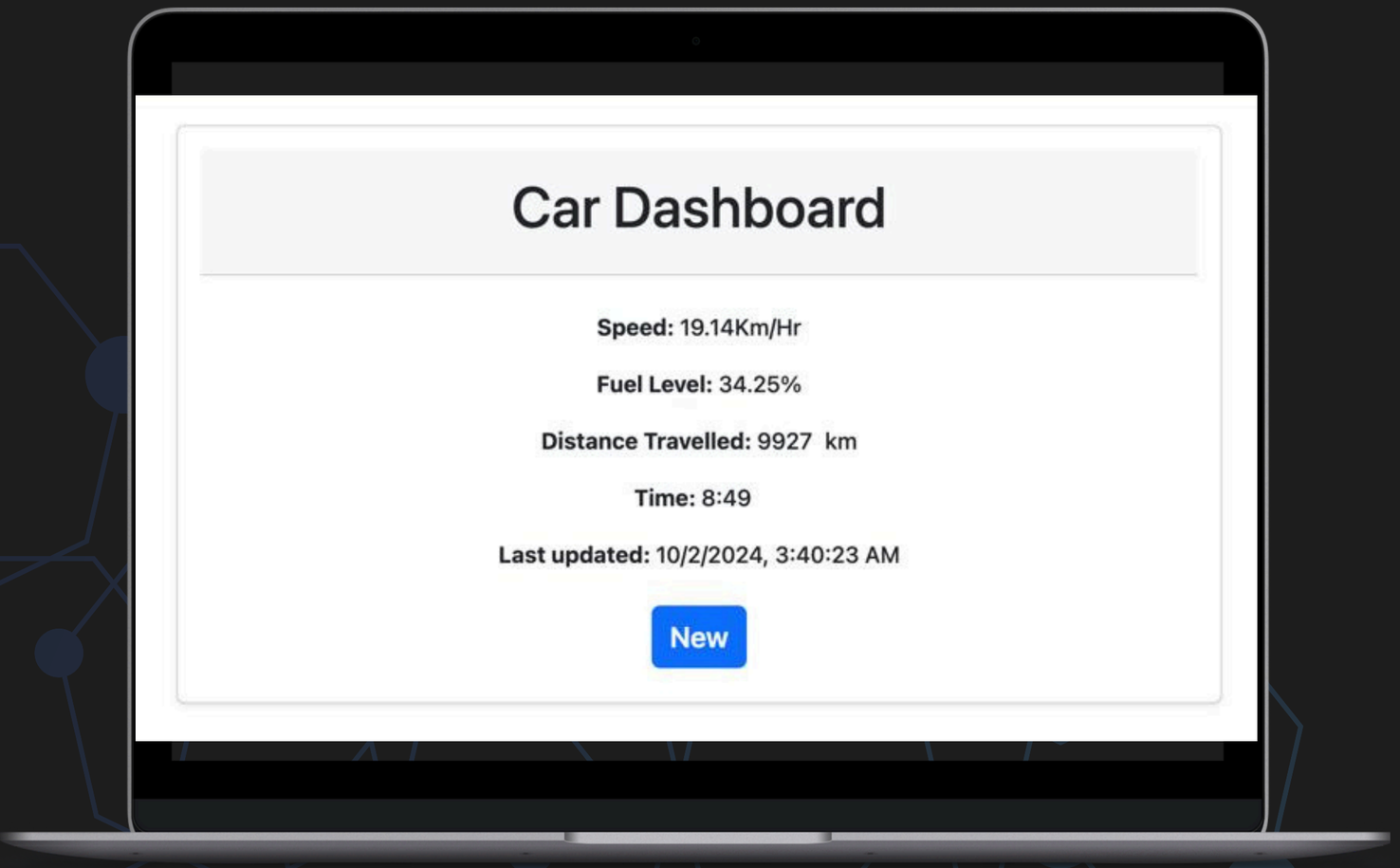
A get request to this URL fetches the most recent data stored in our database with the help of timestamp to identify the recent one. And displays the data on our website.

https://catkin-magenta-path.glitch.me/add

From the python script, we make a post request to this URL which stores the data to our databse.

```
const carSchema = new mongoose.Schema({
    speed: String,
    fuelLevel: String,
    distanceTravelled: String,
    time: String,
    timestamp: { type: Date, default: Date.now }
});
```

User access to data



User can access the data via a website hosted on Glitch. https://catkin-magenta-path.glitch.me/new

The website uses MongoDB as a database to store data and displays the most recently added data from the database.

Frontend: HTML, Bootstrap

Backend: Node.js, Express.js

Team members

Dushyant Agrawal (22095034)

Kothapalli Santosh (22095054)

Kritarth (22095055)

Mamidwar Shravani Vinod (22095059)

Mayank Agrawal (22095061)