Product Designing

TEAM 311

Product name: AVINYA

CONTENTS

Motivation & Problem Statement:

Indian statistics reveal that more than 60% accidents occur during night time. Poor visibility, driver's visual fatigue & performance etc were identified as prime factors for more accidents during night time. Visibility is related to the factors like scattered right, insufficient light, reflection of lights, eyes power to regain its original vision etc., During night one's judgment will be poor and cannot able to judge other vehicle's speed correctly. To avoid poor visibility, proper designing of illumination system is stressed. During night, being seen is an important as seeing. Road Geometry will sometimes mar the visibility which may cause accidents. Driver's fatigue is more during night since the driver is doing his work against the nature.

The task is to develop a design for a product which can be used to reduce the accidents occurring during night time in an efficient manner.

Existing Products in the market:

As of now, there are no existing products like ours except for the technology used in cars produced by Tesla. As Tesla cars are not available to the middle-class public, the approach was to create a feasible and cost-efficient model which helps the driver navigate the vehicle with enhanced measures of safety.

Fig 1: Tesla Autopilot & Surround sensing system

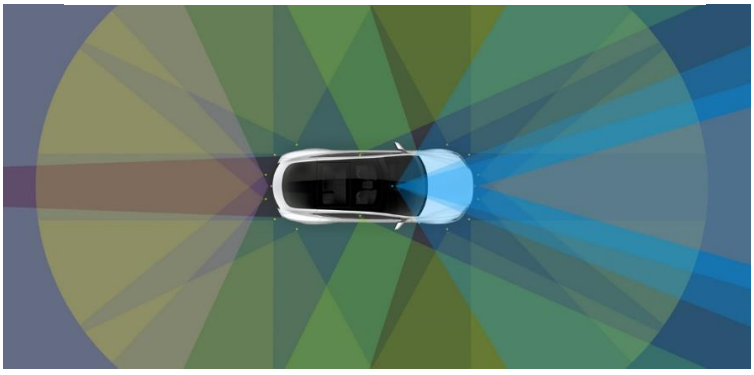


Fig 2: The schematics for range of the sensor systems

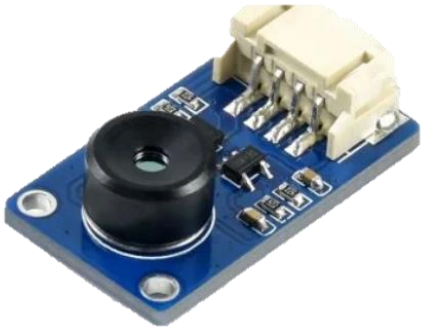
Devised Solution – AVINYA

A Brief understanding of the product:

The design takes advantage of the 2 thermographic cameras and images produced by them to detect any animals, humans or vehicles present in the vicinity of the device. As soon as any of the above mentioned entities is detected a stream of light will be projected in the direction of the detection which lets the driver realise it's presence and therefore, warns him/her to be extra cautious.

Parts of the Product:

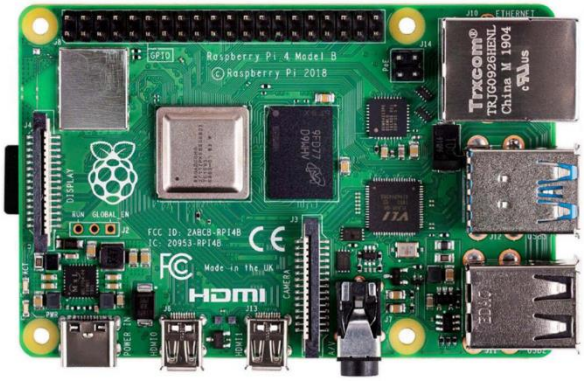
1. **Thermal Camera**:
   The Thermal Imaging Camera is a type of sensor that uses infrared radiation to create a visual representation of temperature differences. It consists of an array of pixels, each of which contains a tiny sensor that detects infrared radiation emitted from an object. The camera works by detecting the amount of infrared radiation emitted from each pixel in the array and converting it into a temperature reading. These temperature readings are then combined to create a thermal image of the object being measured.



2. **Brushless Motor:**
   Brushless motors use electronic commutation to control the power supplied to the motor windings. The motor consists of a rotor (the rotating part) and a stator (the stationary part). Inside the motor are multiple electromagnets on the stator, producing a rotating magnetic field. This field interacts with the permanent magnets on the rotor, causing it to rotate. The electronic commutation system uses sensors to detect the position of the rotor and switches the power supply to the windings accordingly. This allows for precise control of the motor's speed and direction.

3. **Beam Lights:**
   High-beam bulbs are typically designed with a higher wattage and a different filament shape than low-beam bulbs, which allows them to produce brighter and more intense light.

4. **Raspberry-Pi:**
   Raspberry Pi is a small, single-board computer designed for hobbyists, educators, and DIY enthusiasts. It is about the size of a credit card and runs on an ARM processor. The Raspberry Pi works by running a variety of software programs on its operating system, which is typically Linux-based. The user can write and run programs in a variety of programming languages, including Python, Java, and C++. The Raspberry Pi has various input and output options, including USB, Ethernet, HDMI, and GPIO pins, which allow it to interact with sensors, motors, and other hardware components.



5. **Fibre Glass Body:**
   Fibre glass is used as the chassis protecting the whole circuit and the other components such as lights, bearings, etc. The advantages of fibre glass chassis are:

   a. Excellent optical clarity and transparency
   b. Highly resistant to variations in temperature
   c. Has good Impact resistance and resistance to chemicals
   d. Lightweight and easy to manufacture.

6. **Bearings:**
   Bearings are mechanical components that reduce friction between two moving parts and support the weight or load of a rotating shaft. They work by allowing the rotating shaft to spin smoothly inside an outer ring or housing, while the inner ring or race is fixed to the shaft.

Working Principle:

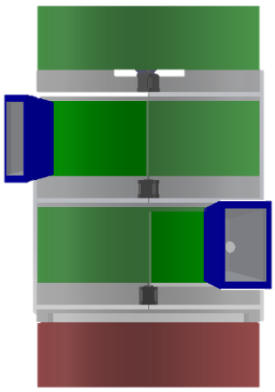
The device consists of 4 compartments stacked on top of one another. The first compartment comprises of the R-Pi module and the thermal cameras. The three compartments above it consists of bearings and motors onto which high beam lights are fixed.

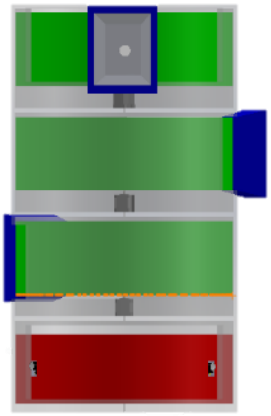
The basic functionality of the device starts with bottommost compartment. The images captured by those cameras are divided into frames and are analysed by the R-Pi module.

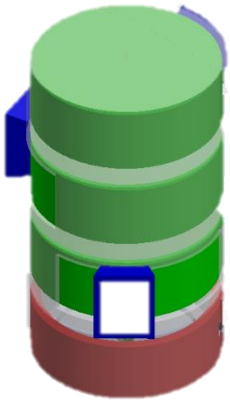
The R-Pi takes in all the frames and analyses it with the code fed into it. Moving objects with respect to the camera are identified and its angular position will be fed to one of the motors of the three light beam compartments depending on availability.

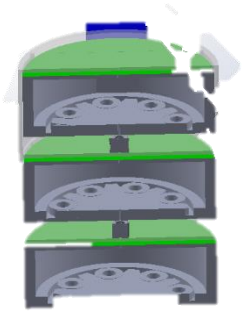
The reason behind 3 light beam compartments is to target multiple movements around the vehicle which may increase the probability of an accident.

Design Snippets:









In the CAD diagrams depicted above, the green compartments are the ones which house the high beam light and the motors whereas the heart of the product, i.e., the raspberry-pi module and the thermographic cameras are kept inside the red compartment.

The fourth image shows the cross section of the green compartments and the bearings used to rotate the lights can be noticed.

The first 3 images are the isometric and side views of the product.

R-Pi Code:

```
#get_background.py

import numpy as np
import cv2


def get_background(file_path):
    #df_gpa.replace(to_replace=[None], value=np.nan, inplace=True)
    cap = cv2.VideoCapture(file_path)
    # we will randomly select 50 frames for the calculating the median
    frame_indices = cap.get(cv2.CAP_PROP_FRAME_COUNT) * np.random.uniform(size=50)
    # we will store the frames in array
    frames = []
    for idx in frame_indices:
        # set the frame id to read that particular frame
        cap.set(cv2.CAP_PROP_POS_FRAMES, idx)
        ret, frame = cap.read()
        frames.append(frame)
    # calculate the median
   # try:
        median_frame = np.median(frames, axis=0).astype(np.uint8)
    #except TypeError:
        #median_frame = 0
    return median_frame


#detect.py

import cv2
import argparse
from get_background import get_background
import math

parser = argparse.ArgumentParser()
parser.add_argument('-i', '--input', help='path to the input video',
                    required=True)
parser.add_argument('-c', '--consecutive-frames', default=4, type=int,
                    dest='consecutive_frames', help='path to the input video')
args = vars(parser.parse_args())

cap = cv2.VideoCapture(args['input'])
# get the video frame height and width
frame_width = int(cap.get(3))
frame_height = int(cap.get(4))
save_name = f"outputs/{args['input'].split('/')[-1]}"
# define codec and create VideoWriter object
out = cv2.VideoWriter(
    save_name,
    cv2.VideoWriter_fourcc(*'mp4v'), 10,
    (frame_width, frame_height)
)

#get the background model
background = get_background(args['input'])
# convert the background model to grayscale format
background = cv2.cvtColor(background, cv2.COLOR_BGR2GRAY)
frame_count = 0
consecutive_frame = args['consecutive_frames']

while (cap.isOpened()):
    ret, frame = cap.read()
    if ret == True:
        frame_count += 1
        orig_frame = frame.copy()
        #convert the frame to grayscale first
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        if frame_count % consecutive_frame == 0 or frame_count == 1:
            frame_diff_list = []
        # find the difference between current frame and base frame
        frame_diff = cv2.absdiff(gray, background)
        # thresholding to convert the frame to binary
        ret, thres = cv2.threshold(frame_diff, 50, 255, cv2.THRESH_BINARY)
        # dilate the frame a bit to get some more white area...
        # ... makes the detection of contours a bit easier
        dilate_frame = cv2.dilate(thres, None, iterations=2)
        # append the final result into the `frame_diff_list`
        frame_diff_list.append(dilate_frame)
        # if we have reached `consecutive_frame` number of frames
        if len(frame_diff_list) == consecutive_frame:
            # add all the frames in the `frame_diff_list`
            sum_frames = sum(frame_diff_list)
            # find the contours around the white segmented areas
            contours, hierarchy = cv2.findContours(sum_frames, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
            # draw the contours, not strictly necessary
            for i, cnt in enumerate(contours):
                cv2.drawContours(frame, contours, i, (0, 0, 255), 3)
            for contour in contours:
                # continue through the loop if contour area is less than 500...
                # ... helps in removing noise detection
                if cv2.contourArea(contour) < 500:
                    continue
                # get the xmin, ymin, width, and height coordinates from the contours
                (x, y, w, h) = cv2.boundingRect(contour)
                # draw the bounding boxes
                cv2.rectangle(orig_frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
                # distance variable gives the pixel value of middle point of contour
                distance=x+(w/2)
                deg=math.atan(math.tan((math.pi*55)/180)*(1-(x/320)))
                print(deg)
            cv2.imshow('Detected Objects', orig_frame)
            out.write(orig_frame)
            if cv2.waitKey(100) & 0xFF == ord('q'):
                break
    else:
        break
cap.release()
cv2.destroyAllWindows()
```

Output of the given code will be as shown below:

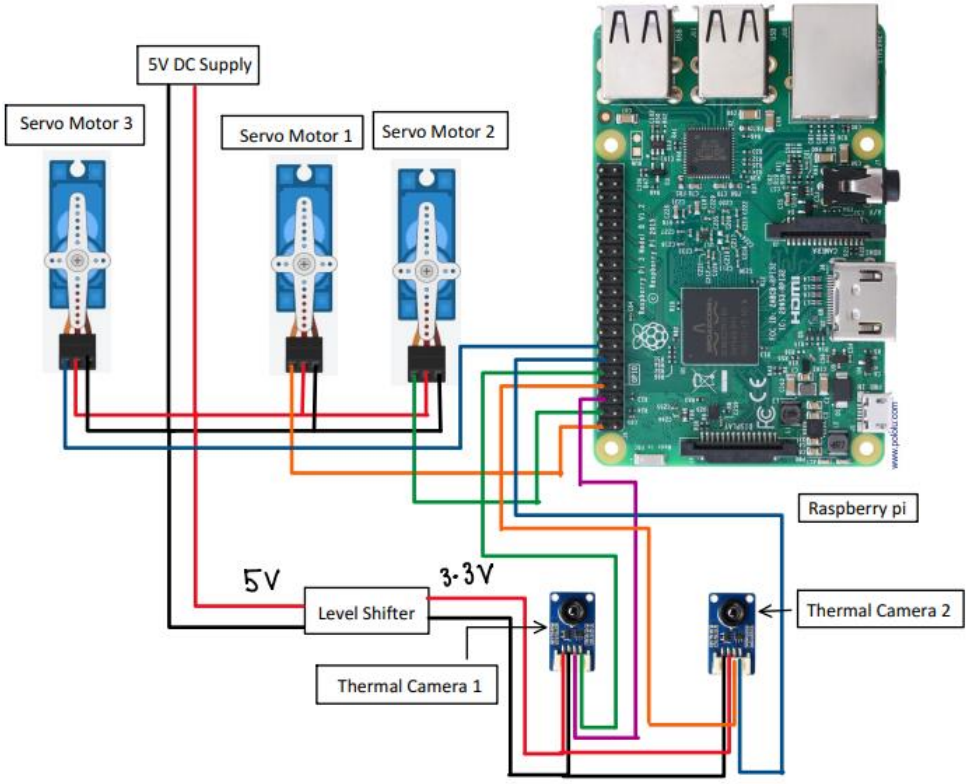
Development Possibilities:

Bluetooth modules can be added to the R-Pi board and the data collected via the thermal signatures can be quantified in many user-friendly forms and can even be synced to the car's UI system.

Even the number of thermal cameras can be increased and can be fit into the car chassis even though that reduces the comfort of the device being detachable.

ANNEXURE

ANNEXURE-1: CIRCUIT DIAGRAM



The circuit shows the how the R-Pi is connected to all the servos and also the thermal cameras.

The power supply is directly given from the car/vehicle which saves the work of changing the power source constantly.

ANNEXURE 2: COST ANALYSIS

SR. No.	Different Parts Required	Quantity	Per unit cost (in Rs)	Total cost (in Rs)
1	Thermographic cameras	2	5000	10000
2	Brushless Motors 24W, 7.2V	3	900	2700
3	Bearings	3	100	300
4	Bulbs	3	700	2100
5	Connecting wires	-	-	100
6	Raspberry pi	1	1000	1000
7	Fibre body	1	1	200
	TOTAL			16400