Real Time Hump Detection on urban area at nighttime

Abstract

Speed breakers or Hump has formed to reduce the accidents, but unfortunately excessive use of speed breakers on urban area / inner cities / metropolitan it distracts vehicle drivers. In addition to that, most of time drivers often cannot recognize the appearance of unmarked speed breakers and lose control of the vehicle, causing serious accidents and loss of lives. Especially in India speed breakers may not be visible to driver's eye due to fade borders, having no cross lines to indicate, damaged road, less differentiation from road and so on due to poor maintenance, in this article we proposed a new approach to sort out identified problems using image processing techniques.

As we seen the high severity is in nighttime most cases, we planned to developed algorithms to work on nighttime as driver may avoid / mitigate the critical situation in daytime.

As we had a thought of produce a commercial product out of this solution, we have used 100% real time videos, we captured videos in and around KTC Nagar area in Tirunelveli 627011, Tamil Nadu, India, from swift car by mounting the camera on rear view mirror

In Scope

Detection of Hump with / without markings at Nighttime in urban area, streets, inside cities road etc.,

Out of Scope

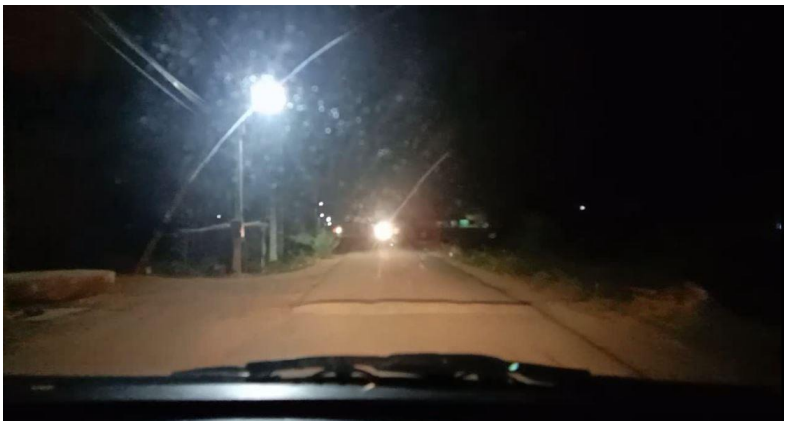
Detection of Hump with / without markings in Daytime

Detection of Hump in highway as driver can easily predict since markings, less traffics helps.

Solved scenarios

Here we have listed few of solved scenarios where in we able to detect Hump before time to alert driver, listed figures will be helpful to understand the scenario better,

a. single or multiple glare present near Hump which is to be detected



b. No markings on Hump which is to be detected

c. Hump occluded by opposite vehicle

d. Hump occluded by on-going vehicle

e. Multiple / consecutive humps

Results:

Figure shows the results of detection of hump for the scenario of Hump occluded by on-going vehicle (d) and single or multiple glares present near hump (a)

Conclusion:

Proposed system was developed in python platform along with OpenCV libraries in windows 10 environment and produces excellent results in terms of accuracy as well as performances, we could achieve almost 90% of accuracy wherein algorithm was being executed for large number of real time videos and takes 25 msec for each frame in other word can say 40 FPS (Frames Per Second)

Performances:

System specifications	FPS Range
Core i5 HDD	30 > FPS < 35
Core i5 SDD	40 > FPS < 45
Accuracy	~90

Author:

Proposed approach designed and developed by Mr. Dhinakaran, who done his Engineering at National College of Engg, Tirunelveli, Tamilnadu. Currently working as Tech lead in imaging Technology Lab (Research and Development unit) at HCL Technologies Bangalore, he published paper in IEOM Dubai conference titled " Dust Particles Detection on Camera Lens **"** , has developed many image processing applications in various domain like Automotive, Medical, product inspection etc.,

Hump Detection - Program execution

PC Specifications

The PC specification used for the Failsafe is:

- Operating System : Windows 10 Enterprise, 64 Bit
- Processor : Intel(R) Core (TM) i3-2120 CPU @ 3.30GHz
- RAM : >4 GB

Tools used for development

- Python
- OpenCv, NumPy (pip install opencv-python, pip install numpy)
- Visual studio coder

System Input / Output

- Input Sources:
  - video captured from car on live
- Input Data:
  - Captured videos
- Output:
  - Hump detected results

There are 2 python files to execute this program

1. calibration_set.py -> to set calibration according camera mounted on car

2. hump_detection.py -> to detect hump from given videos

Here we can see one by one

Chapter 1: setting calibration

why calibration:

camera fitted on car is differ from vehicle to vehicle or person to person some may fit at center of the car when somehow far away from center of the car inside windshield

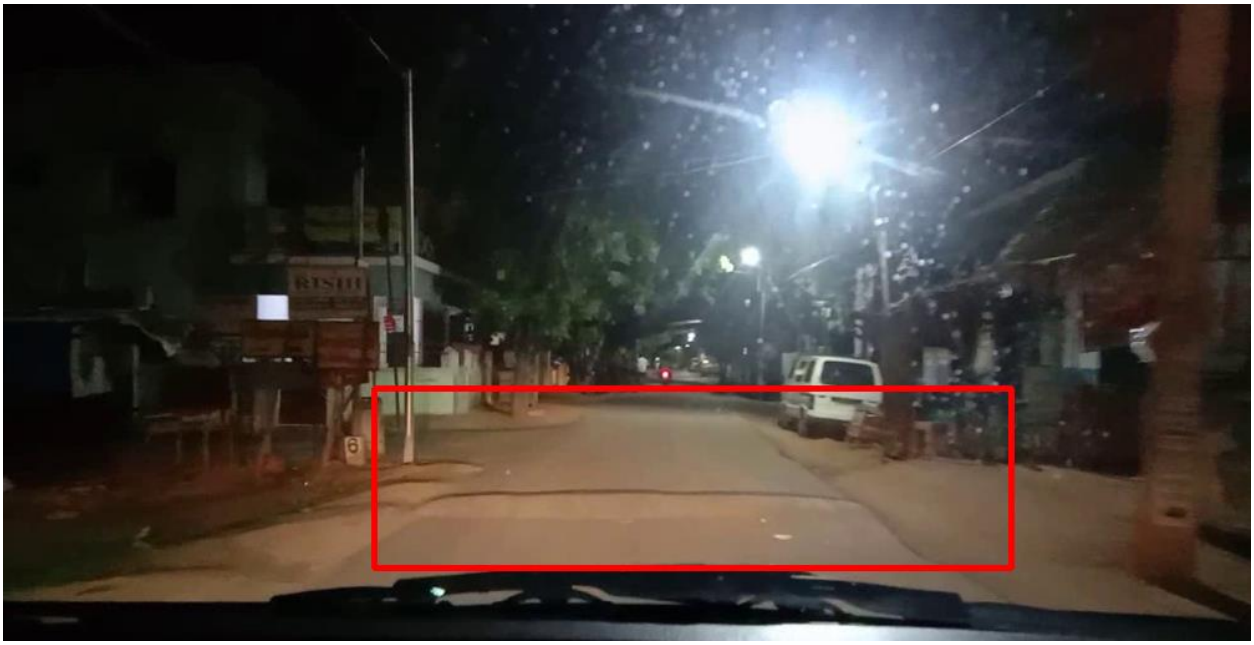
for example,

- given sample videos has been captured by fitting the camera on rear view mirror
- swift Lxi car was used for this experiment
- parameters values are being seen in calibration.txt based on current set up

How to set calibration properly

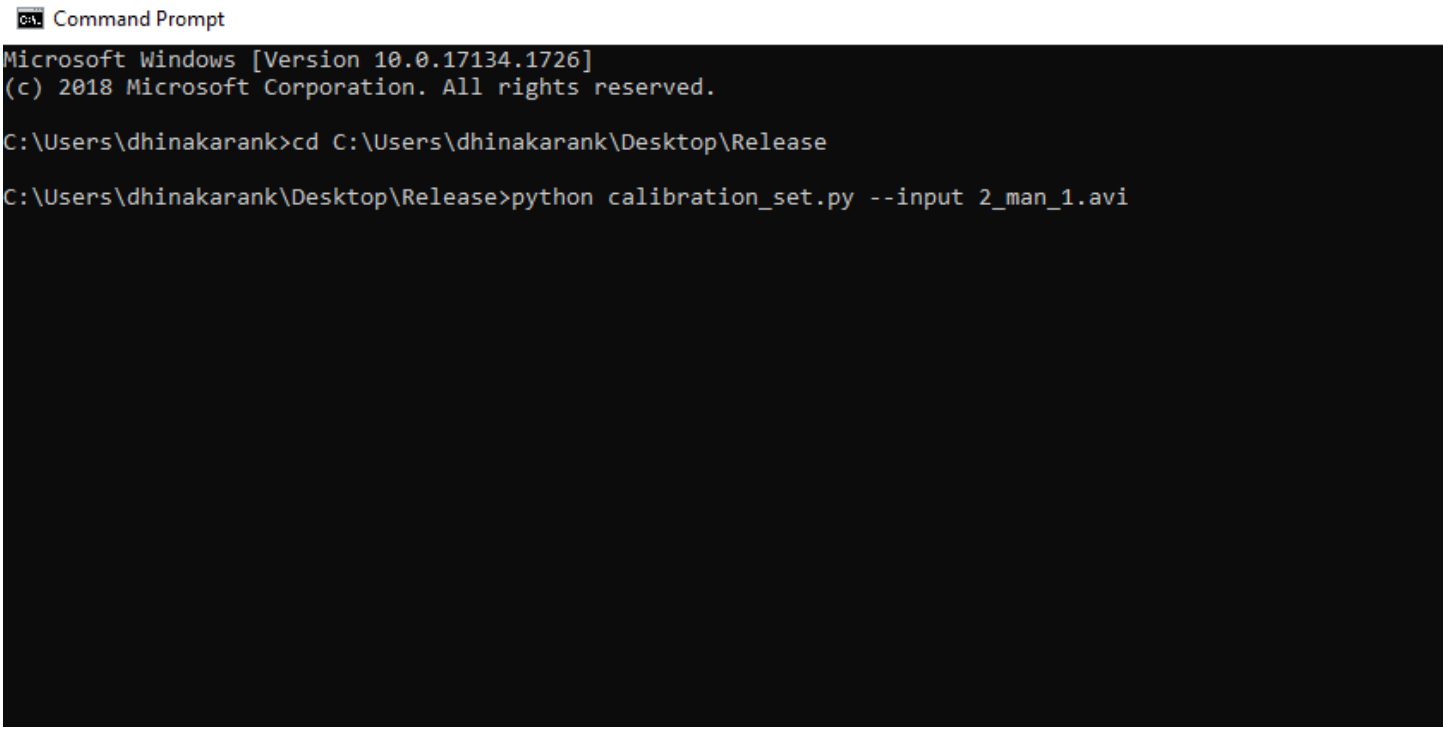
execute calibration_set.py script to see your ROI and confirm either vehicle trajectory is fitted on ROI or not, if not please adjust the values accordingly in calibration.txt file

sample calibration result:



shown red color box is an expected ROI which is calibrated from user input given in calibrated.txt

example of setting calibration or how to run calibration_set.py to see ROI on screen



Chapter 2: executing Hump Detection

we can see how to run hump_detection.py to see the results

```
Command Prompt
Microsoft Windows [Version 10.0.17134.1726]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\dhinakarank>cd C:\Users\dhinakarank\Desktop\Release

C:\Users\dhinakarank\Desktop\Release>python hump_detection.py --input 2_man_1.avi --display 1
```

sample hump detection results

Shown Green color is a detected Hump on given frame and we can get FPS information on top left corner when Red box is processed ROI

Common information:

- Please keep temp.png file in Hump_Detection folder
- Result calculated from single frame not with multi frame / tracking
- Program made to be run in nighttime only

What next release will have (ver 2.0):

- Program may produce results for Daytime also
- Multi frame analysis will be done for better performance
- Try to bring Dynamic calibration with detection of vanishing point so that it may not require user inputs

Help

- Please mail us if you are not able to execute or any mismatched results

  dhinakarank@hcl.com