ISSN 2349-6193
Impact Factor: 2.805

IJESMR
International Journal OF Engineering Sciences & Management Research

CALCULATING THE SPEED OF A VEHICLE USING PYTHON OPENCV IN REAL TIME IMAGE PROCESSING ON A RASPBERRY PI PROCESSOR

Ganeshan M
Alpha college of Engineering, Bangalore, VTU Belagavi

Keywords: Image Thresholding, Speed Detection, Python OpenCV, Raspberry Pi.

ABSTRACT
The speed detection camera for residential users can be developed by using a surveillance/security camera by the help of the chosen Raspberry Pi Processor. Image detection and processing can be accomplished by using Python programming. We can import several pre-written libraries such as motion library. We can set the camera to take the specified number of images over a specified period of time. The Raspberry takes multiple photos of the over-speeding vehicle, calculates the vehicle's speed and stores the data and pictures on its memory card. This information could then be useful to police to realize that there is a speeding problem in the neighborhood. Image thresholding method for speed detection is better than the commonly used RADAR as it has greater reliability, especially in unfavorable weather conditions. It will not compete or interfere with the steps taken by the local police. It will only prove that the problem exists as it gives the right information as proof.

INTRODUCTION
One of the methods for calculating a vehicle's speed is by using RADAR. This method is used by the police to determine a vehicle's speed and such systems are not affordable by the common public since this system is very costly. The concept used by the RADAR system is Doppler Shift in which the RADAR gun is used to determine moving object speed. The rate of change of frequency of a moving object is also determined by the RADAR gun [2]. But there are several disadvantages in this system. There are many ways to interfere with the signal developed by the RADAR. Detectors are available which will alert the drivers the areas where RADARs are deployed [1]. Scramblers are designed to corrupt the signals produced by the detector.

Designing a cost-effective system for speed detection of vehicles which can be commonly used by public is a challenging task. This paper explains a method simpler when compared to RADAR system using Doppler Shift. In this process we use raspberry Pi, Python Programming for image processing using OpenCV and a surveillance camera to meet our need.

RASPBERRY PI
The Raspberry Pi is a very capable minicomputer that's small enough to fit in your pocket. The Raspberry Pi is powerful and inexpensive. It was developed by the Raspberry Pi Foundation in the UK basically to teach computer science fundamentals in schools. A little setup and a lot of other parts are required to get it up and running- 2 USB ports, HDMI and Ethernet hookups, SD card slot, memory, video/audio outputs, and power source. The Pi works on Linux operating system and uses Python for programming.

The camera module has a five megapixel fixed-focus camera that supports 1080p30, 720p60 and VGA90 video modes, as well as stills capture. It attaches via a 15cm ribbon cable to the CSI port on the Raspberry Pi.

The camera works with all models of Raspberry Pi 1 and 2. It can be accessed through the MMAL and V4L APIs, and there are numerous third-party libraries built for it, including the Pi camera in Python library, the modules necessary for the process are:

```
#!/usr/bin/python

import subprocess

import os

import picamera

import time

import shlex
```

ISSN 2349-6193
Impact Factor: 2.805

IJESMR
International Journal OF Engineering Sciences & Management Research

```
from datetime import datetime
from datetime import timedelta
import datetime as dt
import tgl
import sys
import RPi.GPIO as GPIO
from functools import partial
ppth = os.path.abspath(__file__)
print ppth
```

PYTHON OPENCV

Open Source Computer Vision, often called as OpenCV, is a more advanced image manipulation and processing software than Python Image Library. It has been implemented in many languages and is widely used [5]. OpenCV supports Python, Java, C++ etc., wide variety of programming languages and it compactable on different operating system such as Windows, Linux, OS X, Android, and iOS. Interfaces for high-speed GPU operations based on CUDA and OpenCV are also under active development. This paper is mainly focused on OpenCV 3.x version.

IMAGE PROCESSING

Image processing involves a three-step algorithm to find the speed of a moving object. The first step is called image detection. The simplest thresholding methods replace an image with a black pixel if the image intensity is less than some fixed constant or a white pixel if the image intensity is greater than that constant [3]. So, each pixel will either be black or white. The resulting image is the object of our interest so that it can be easily tracked (Figures 1 and 2). We have the option of choosing which color is the foreground and which is the background. White is the foreground in Figure 2.

The step that follows object detection is object tracking. As shown in the pictures above, the parked cars are not to be considered. By subtracting the background from several frames we can decide whether an object is moving or not (Figure 3). Subtracting the background of two consecutive frames will remove all the stationary objects, displaying only the moving object. The final step in the process is to measure the speed. In Figure 2, the leading edge of the moving object is used in speed detection. Using a ruler in the images will find the distance covered by the leading edge of the object and the corresponding time taken. The ratio of distance and time yields the speed. The flowchart shown in figure 4 depicts the step by step procedure and the logic behind speed detection of over-speeding vehicles and storing the log information.

Figures1 and 2: Actual picture and image after applying thresholding

ISSN 2349-6193
Impact Factor: 2.805

IJESMR
International Journal OF Engineering Sciences & Management Research

Matplotlib is an optional choice for displaying frames from video or images. Numpy represents "numbers and Python."Numpy's array functionality is being used here. At the end, we are using the python-specific bindings for OpenCV called python-OpenCV.

```
import cv2
import matplotlib
import numpy
```

Video Recordings are actually frames, displayed one after another, at the rate of thirty to sixty times a second. Basically, they are static frames. Object recognition can be done with almost the exact same code on images and video. Next, a lot of image analysis requires simplifying the source as much as possible. This usually begins with a conversion to gray Scale. Here, the matter is simple. If pixel value is greater than a threshold value, it is assigned one value (white), else it is assigned another value (black). The function used is cv2.threshold. First argument is the source image, which should be a gray scale image. Second argument is the threshold value which is used to classify the pixel values. Third argument is the maxVal which represents the value to be given if pixel value is more than the threshold value. OpenCV provides different styles of thresholding and it is decided by the fourth parameter of the function. Different types are:

- cv2.THRESH_BINARY
- cv2.THRESH_BINARY_INV
- cv2.THRESH_TRUNC
- cv2.THRESH_TOZERO
- cv2.THRESH_TOZERO_INV

Code :

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

img = cv2.imread('gradient.png',0)

ret,thresh1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)

ret,thresh2 = cv2.threshold(img,127,255,cv2.THRESH_BINARY_INV)
ret,thresh3 = cv2.threshold(img,127,255,cv2.THRESH_TRUNC)
ret,thresh4 = cv2.threshold(img,127,255,cv2.THRESH_TOZERO)
```

The output of this code is our threshold image.

BACKGROUND SUBTRACTION

Background subtraction is an important step before image processing. If we have an image of background alone, like image of the road without vehicles, it is very easy. Simply subtract the new image from the background and we get the foreground objects alone. But usually, we will not have such an image, so we need to extract the background from whatever images we have. It is all the more complicated when there is shadow of the vehicles. Since shadow is also moving, simple subtraction will mark that also as foreground. It complicates things [4].

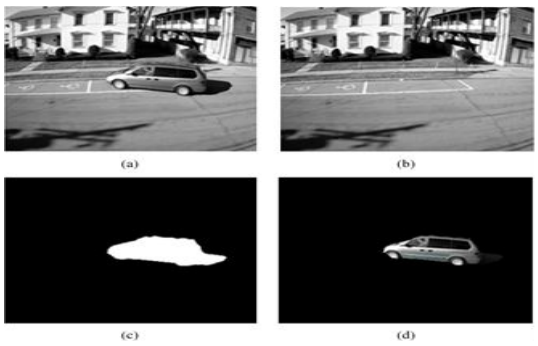

Figure3. Example of background subtraction

ISSN 2349-6193
Impact Factor: 2.805

IJESMR
International Journal OF Engineering Sciences & Management Research

Background Subtractor MOG

It is a Gaussian Mixture-based Algorithm. It uses a method to represent each background pixel by a mixture of K Gaussian distributions (K = 3 to 5). The weights of the mixture correspond to the time proportions that those colors stay appear in the images. The probable background colors are those which stay longer and more stationary.
While coding, we create a background object using the function **cv2.createBackgroundSubtractorMOG()**. The optional parameters like length of history, number of gaussian mixtures, threshold are set to some default values. Then inside the video loop, background subtractor.apply() method is used to get the foreground mask.

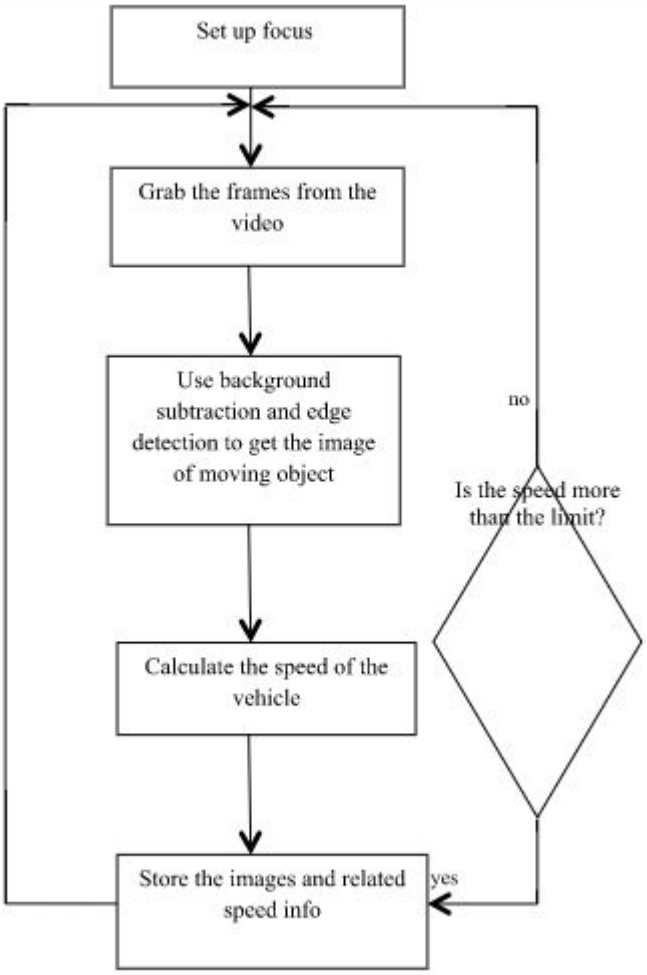

Figure 4. Python OPENCV programming flow chart for speed detection

```
import numpy as np
import cv2
cap = cv2.VideoCapture('vtest.avi')
fgbg = cv2.createBackgroundSubtractorMOG()
while(1):
ret, frame = cap.read()
fgmask = fgbg.apply(frame)
cv2.imshow('frame', fgmask)
k = cv2.WaitKey(30) & 0xff
if k = 27:
```

ISSN 2349-6193
Impact Factor: 2.805

IJESMR
International Journal OF Engineering Sciences & Management Research

```
break
cap.release()
cv2.destroyAllWindows()
```

EDGE DETECTION

The process of Edge-detection involves four steps:

Since edge detection is susceptible to noise in the image, noise is removed using 5x5 Gaussian filter [6].

The image is filtered with a Sobel kernel in both horizontal and vertical direction to get first derivative in horizontal direction and vertical direction. From these two images, we can find edge gradient and direction for each pixel.

A full scan of image eliminates any unwanted pixels which may not form the edge. In order to do this, every pixel is checked if it is a local maximum in its surrounding in the direction of gradient. The output is a binary image with "thin edges".

Any edges with intensity gradient more than *maxVal* are definitely edges and those below *minVal* are definitely non-edges. The non-edges are then removed. Those which lie between these two values are divided as edges or non-edges depending on their connectivity. If they are connected to "sure-edge" pixels, they are considered to be part of edges. This step also removes small pixels noises based on the fact that edges are long lines. Therefore the final output is strong edges in the image.

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
img = cv2.imread('car.jpg',0)
edges = cv2.Canny(img,100,200)
plt.subplot(121),plt.imshow(img,cmap = 'gray')
plt.title('Original Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(edges,cmap = 'gray')
plt.title('Edge Image'), plt.xticks([]), plt.yticks([])
plt.show()
```

CONCLUSION

This goal of this paper is the possibility of a fully working cost-effective and simple surveillance and calculation system so that residents can prove that there is over speeding problem in their neighborhood. The microprocessor is made to be used for educational projects, and the camera was a simple and cheap USB webcam. The Python image processing libraries were imported to calculate the thresholding. The ability to capture and store pictures in a log was demonstrated as was the potential for estimating speed using the thresholding technique.

A concealed, automatic speed detection system that could be placed outside a common man's house and produce a log and images of speeding vehicles in the neighborhood has many potential benefits as well as buyers. Embedded systems projects are a great teaching tool as it ties in both the hardware and software components of computer engineering.

REFERENCES

[1] *Osman Ibrahim, Hazem ElGendy, and Ahmed M. ElShafee, "Speed Detection Camera System using Image Processing Techniques on Video Streams," International Journal of Computer and Electrical Engineering vol. 3 December 2011*

[2] *"Introduction to Radar System and Component Tests" White paper by Rohde and Schwarz*

[3] *Paper on "Image Edge Detection Based On Opencv" byGuobo Xie and Wen Lu School of computer, Guangdong University of technology, Guang zhou, China 510006*

[4] *Independent Multimodal Background Subtraction Domenico Bloisi and Luca Iocchi Department of Computer, Control, and Management Engineering - Sapienza University Of Rome, Italy*

[5] *http://docs.python-guide.org/en/latest/scenarios/imaging*

[6] *http://scikit-image.org/docs/dev/auto_examples/plot_canny.html*