COMPARATIVE ANALYSIS OF VEHICLE DETECTION METHODS USING OPENCV

By

Ray Anthony Y. Saavedra

SCHOOL OF ARTS AND SCIENCES

ATENEO DE DAVAO UNIVERSITY

OCTOBER 2013

Comparative Analysis of Vehicle Detection Methods Using OpenCV

A Thesis / Research Project

Presented to

The Faculty of the Computer Studies Division

Ateneo de Davao University

In Partial Fulfillment

of the Requirements for the Degree

Bachelor of Science in Information Technology

By

Ray Anthony Y. Saavedra

SCHOOL OF ARTS AND SCIENCES

ATENEO DE DAVAO UNIVERSITY

OCTOBER 2013

TABLE OF CONTENTS

Comparative Analysis of Vehicle Detection Methods Using OpenCV

AY ANTHONY Y. SAAVEDRA

ABSTRACT

An efficient way of detecting vehicles for a system in traffic management or road monitoring is crucial. If a system can't handle detection of vehicles effectively it can lead to many problems. Computer vision techniques allow you to detect and track vehicles in real time. You can use different techniques to detect vehicles in various conditions. This study will focus on comparing different object detection techniques which are: Double differencing method, appearance-based approach, feature-based approach and edge-based tracking and implement them to be able to identify a more effective vehicle detection method. This aims to provide an efficient way of detecting vehicles in different conditions for a better and more effective traffic management system, road monitoring system and other vehicle related systems. The results show that double differencing method has the highest accuracy rate out of the other 3 object detection techniques; it performs with an average accuracy rate of 78% and an average percent error rate of 22%. It performs well on day conditions including day + rain condition. Individual results for each object detection techniques are presented in this paper.

General Terms: Computer Vision, OpenCV

1. BACKGROUND OF THE STUDY

Computer vision has been an active area of research in recent years and this have found an increasing applications in road monitoring and traffic management. This offers a solution to real problems. However, with these kinds of applications and systems come different computer vision techniques to use. You can have a variety of methods to choose on how to achieve a specific task in computer vision, just like in vehicle detection. However, how would a developer know what kind of computer vision technique to use to be able to achieve good vehicle detection? A good comparison on every technique would be very helpful on giving light as to what kind of technique a developer should use. This can provide an in-depth discussion on every technique giving aspiring developers on the field of vehicle detection more knowledge on what to use and how to use it. Having a good and efficient way of detecting vehicles is needed to create a good traffic management system and other road related applications.

OpenCV is an open source computer vision library. The library is written in C and C++ but there are already developments on interfaces for Python and other languages. OpenCV provides a simple-to-use computer vision infrastructure that can help the proponent on detecting objects and other computer vision tasks.

2. TECHNOLOGY APPLICATION CONTEXT

Different computer vision techniques can be used to create automated surveillance applications focused on vehicle detection for a road monitoring or a traffic management system. There are many vision processing libraries and Python OpenCV is one of those. It has been mainly used on detecting objects, motion tracking, and other computer vision tasks. There is already plenty of research in the

field of computer vision mainly on using OpenCV for vehicle detection. Therefore, with this study, the proponent wants to compare different object detection techniques which are: (1)double differencing method, (2)feature-based approach, (3)edge based tracking, and (4)appearance-based approach, to show an efficient way of detecting vehicles in various conditions like in rain, night, day and other more. These following techniques are chosen based on different papers which used these techniques for vehicle detection. These techniques have shown good detection rates based on the results of the papers and can be good subjects for a comparative analysis to verify which the better vehicle detection technique is. This topic can be a good help to those who have plans on creating effective traffic management systems, or any road monitoring systems.

3. OBJECTIVES OF THE STUDY

The general objective of this research is to provide a comparative analysis of different object detection techniques under various conditions like day, night, day + rain, night + rain, to show an efficient way of detecting vehicles.

The specific objectives are:

- To be able to implement OpenCV with Python bindings to detect and track vehicles and other computer vision tasks.
- To be able to provide a comprehensive comparative analysis using different vehicle detection methods in different conditions.
- To be able to implement different object detection techniques which are the following: Double differencing method, appearance-based approach, feature-based approach and edge based tracking, to show the best way of detecting vehicles.

4. SIGNIFICANCE OF THE STUDY

This study can benefit future researchers in the area of computer vision, especially in vehicle detection, allowing them to have insight on different object detection techniques submitted to different various conditions like rain, night, day. With the following object detection techniques (Double differencing method, appearance-based approach, feature-based approach and edge based tracking), there has not yet been any comparative analysis for these following, this study will show an efficient way of detecting vehicles even in different conditions based on the following object detection techniques. This can also benefit future developers in traffic management system or any road monitoring system to know an efficient way of vehicle detection to be used for their systems. There already had been an active research on the field of computer vision, this study can add to the intelligence of the other researchers who want to perceive on the path of studying vision techniques mainly focused on traffic management and control. Especially on the Python bindings for OpenCV libraries, this is not yet that well documented, that it may provide a clearer understanding on how to use Python interfaces for OpenCV libraries.

5. SCOPE AND LIMITATIONS OF THE STUDY

The study will focus on the use of OpenCV with Python bindings. The methods to be used to be able to track and detect vehicles will all be from the OpenCV library. This scope will only range from the detection and tracking of vehicles in a certain area, like a traffic scene where vehicles are moving continuously, to be able to produce some vehicle detection information like, how many vehicles passed on a certain road, and other more. This research will generally show a result of different comparisons on detecting the vehicles in using different methods. This study will also only focus on the comparison of the vehicles in conditions like: normal day, normal night, day with rain, and night with rain.

The study will be limited only to the detection and tracking of vehicles, any other information like, the type of vehicle and classification, and other vehicle information will no longer be included.

6. REVIEW OF RELATED LITERATURE AND TECHNOLOGY APPLICATIONS

There are already a plenty of works involving computer vision techniques in vehicle detection, and also a plenty of works involving OpenCV which is an open source computer vision library. But there are still a few who used Python as their language in implementing OpenCV, as most are in C and C++.

6.1 Applying Computer Vision to Traffic Monitoring System in Vietnam

[1] The purpose of the paper is to present promising results of the research on applying real-time image processing algorithms to the automatic traffic surveillance system in Vietnam. There are different functions found on the system, these are: solving problems of counting the number of vehicles and estimating the speed of the observed traffic flow from traffic scenes acquired by a camera in real-time. The paper illustrates a complete application of automatic surveillance tested with a large number of real traffic video sequences in Hanoi city. The results are better than expected. There are a lot of vehicles detected and counted, speed is well estimated. The results are more accurate but it depends on different situations of observed traffic flows. The method that they used for the detection of vehicles is double-difference with the combination of edge detection and dilation operator to find out the border of vehicles more completely and hence extract vehicles from background more exactly. The proponent will be using this object detection technique as a part of one of the techniques the proponent will be comparing. This shows good results of vehicle detection and can be tested with other more techniques and into various conditions like in raining, day, night and many more.

6.2 Computer Vision Techniques for Traffic Flow Computation

[2] This research describes an application of computer vision techniques to road surveillance. The project aims to produce system that detects and tracks vehicles in real traffic scenes to generate meaningful parameters for use in traffic management. In modeling the road, they applied a simple homographic transform. Then used two different approaches in detecting vehicles: a feature-based approach that detects and groups corner features in a scene, and an appearance-based approach that trains a cascade of classifiers to learn the appearances of vehicles. Then for the vehicle tracking, the proponents used Kalman filter motion tracker.

The result shows that the feature-based approach using corner features is better than of the appearance based approach using Haar features. However, the appearance-based approach has outperformed the feature-based approach in previous experiments. The proponent will be using the appearance-based approach of this paper as one of the object techniques to be compared with other techniques. Appearance-based approach may show more efficiency in a wider variety of conditions which is not discussed in this paper like in raining, day, night, and other more.

6.3 Edge Based Tracking for Traffic Surveillance

[3]This paper focuses on using edge detection for tracking and detecting vehicles. One problem that this paper wants to study in application is velocity detection which can be a challenging way to detect in a traffic surveillance system. With samples of a normal road, a reference image is created and moving objects are determined from edge detection. A new noise filtering technique is proposed in this paper. The proposed method did not detect invalid objects, and shows an accurate detection. The proponent will be using this object detection technique to be compared with other techniques.

7. PROJECT METHODOLOGY

Figure 1.1 Operational Framework

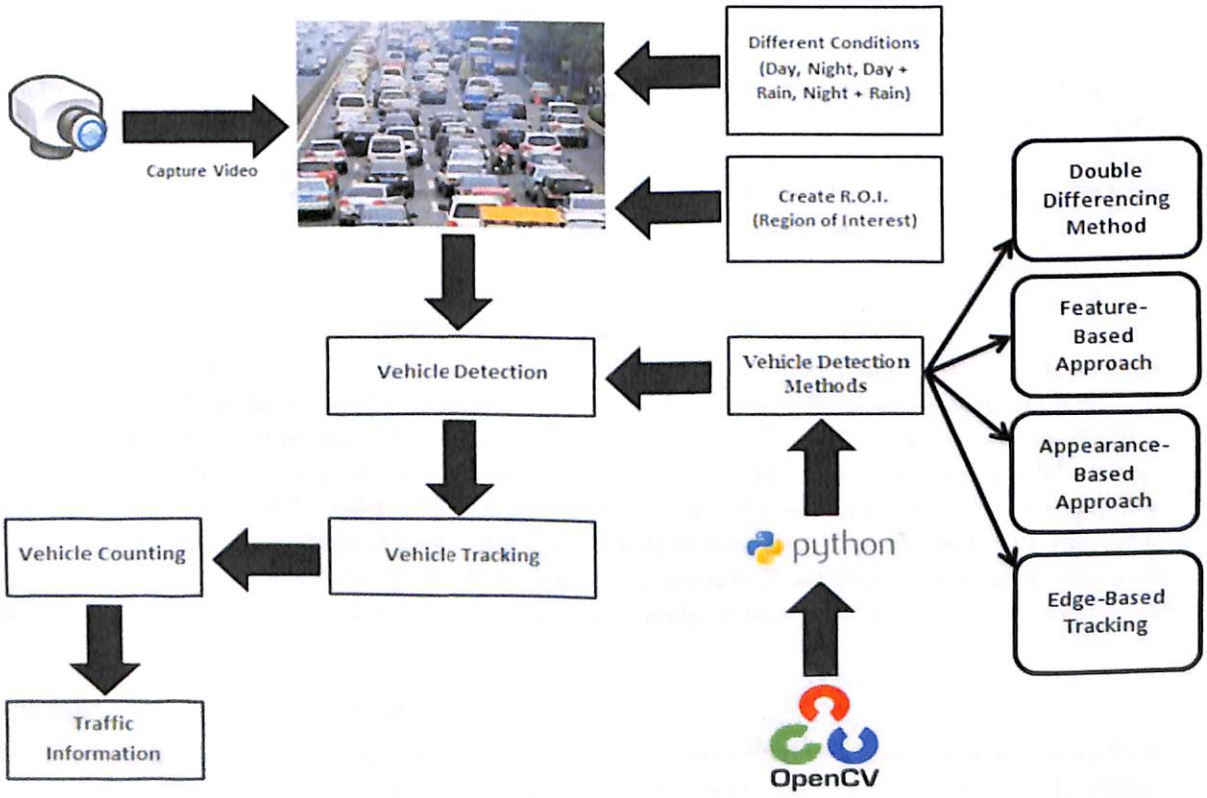
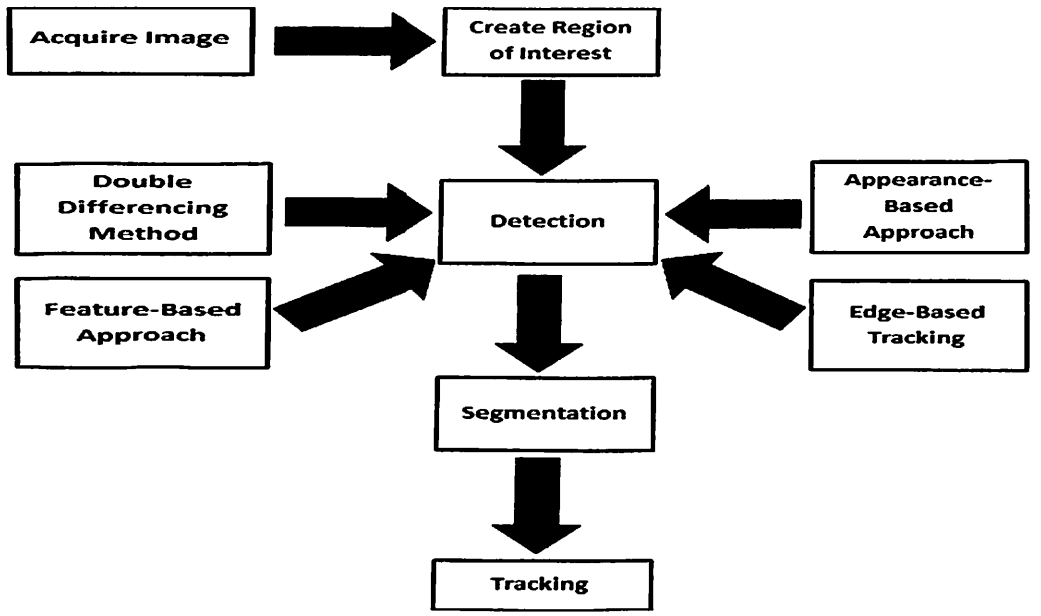


Figure 1.2 Computer Vision Framework



This section will discuss the approach used by the proponent on how the research is done and implemented shown on figure 1.1. The camera used in this research is a Sony Cyber-Shot Digital Camera with a 14.1 Mega Pixels. With a stationary camera, it is placed on an elevated and strategic area wherein it can clearly see the wide road and avoid vehicle occlusion. For this, the proponent used an overpass to clearly see the vehicles passing on the road. The estimated height of the overpass chosen by the proponent is 16 foot, plus the 3.75 foot of the wall where the tripod is placed. The height of the tripod used is 4.16 foot, so the overall height where the camera is situated is 23.91 foot or 24 foot. Different frames are captured from different conditions which are day, night, day + rain and night + rain. Next would be creating a region of interest on a specific area, this region of interest will serve as the tracking area of the vehicles. The length of the region of interest is fixed which is 0.5cm while the width can be changed depending on the choice of the user, but for this width depends on the width of the road chosen as the region of interest. All the pixel points inside the region of interest will be stored in a numpy array which will be used in the later parts. Once the vehicles enter on the region of interest, the vehicles are tracked and once it leaves the area, it will be counted. After the region of interest is set, different object detection techniques will be applied on the different frames. Each object detection techniques as shown on figure 1.2 will be discussed separately on this section.

7.1 Double Differencing Method

Double differencing method relies on motion detection to segment moving regions from a frame. This kind of motion detection is commonly known as frame differencing. In double differencing method, it is obtained by performing a logical AND between pixel points belonging to two consequent frames. Further explanation of the double-difference algorithm can be seen on [1]Anh et al.'s paper. For this, the proponent implemented the double differencing based from [1]Anh et al.'s method.

First is to extract the current frame and the previous frame to be able to apply frame differencing, next is to use cv2.absdiff:

```
difference = cv2.absdiff (currentframe, previous)
```

The result for the following will be different blobs of moving vehicles in the two consecutive frames. To group the following blobs, the proponent used contours to get these blobs and to draw bounding boxes on each detected blobs. The following codes will show on how the proponent implemented contours on the differenced frame:

```
contours,hierarchy= cv2.findContours(medfilter,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)

if hierarchy is None:
   pass
else:
   hierarchy = hierarchy[0]
   for component in zip(contours, hierarchy):
      currentContour = component[0]
      currentHierarchy = component[1]
      if currentHierarchy[2] < 0:
         pass
      elif currentHierarchy[3] < 0:
         x,y,w,h = cv2.boundingRect(currentContour)
         cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),3)
```

7.2 Feature-Based Approach

Another approach on object detection is the feature-based approach. This approach detects features of a vehicle such as corners, windshield, pair of headlights and other more. This kind of approach can provide more advantage to other kind of techniques since this can handle vehicle occlusion and less computation. For this, the proponent proposes a method on how to detect vehicles based from its feature points. First is to create an empty contour, which will serve as a container for our features, because the feature points found in a vehicle comes in clusters, these clusters must be group to be able to identify fully the vehicle moving. The following lines will show how this was implemented.

An empty contour is created:

```
contours, hierarchy = cv2.findContours(frame, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
```

Next is to create a mask on the specific contour blob found, this mask will be used to get all the features on the blob:

```
mask = np.zeros(frame2,shape, np.unit8)
cv2.drawContours(mask, [currentContour], 0, 255, -1)
features_prev = cv2.goodFeaturesToTrack(mask, mask = None, **feaure_params)
```

After applying cv2.goodFeatureToTrack on the mask, we will check if there are features found on the specific contour blob, if features are found, a bounding box will be drawn on that blob.

7.3 Edge-Based Tracking

The proposed method as [3]Zabihollahi et al. discussed on their paper is based on frame differencing. This method uses a pure edge base background modeling. Based from frame differencing, we will first get the differenced image from two consecutive frames of the video; this will enable us to find all moving objects on the frame.

```
difference = cv2.absdiff (currentframe, previous)
```

After applying frame differencing, we will now use edge detection to detect all edges from the moving objects.

```
edge = cv2.Canny(thresh, retval, retval * 2, apertureSize = 5)
```

The edges will draw blobs on each moving vehicles in the frame, after getting the blobs, we will now group them using contours. And after getting the contours, a bounding box will be drawn on each detected vehicles.

```
contours,hierarchy= cv2.findContours(medfilter,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)

if hierarchy is None:
    pass
else:
    hierarchy = hierarchy[0]
    for component in zip(contours, hierarchy):
        currentContour = component[0]
        currentHierarchy = component[1]
        if currentHierarchy[2] < 0:
            pass
        elif currentHierarchy[3] < 0:
            x,y,w,h = cv2.boundingRect(currentContour)
            cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),3)
```

7.4 Appearance-Based Approach

Appearance-based approach uses pattern recognition techniques on detecting vehicles and non-vehicles through their appearances. An AdaBoost classifier training algorithm is also used as stated in [2]Bai et al.'s paper. This uses positive training images and negative training images to extract Haar features on each vehicle. Further discussions on Haar features can be found on [2]Bai et al.'s paper. For this, the proponent stored 1,000 positive images, and 900 negative images. These positive images are different pictures of vehicles, while the negative images are the non-vehicle pictures. These images will be used to train to the haar cascade classifier. The proponent uses [6]Kumar's haar cascade classifier trainer to train the following images. After training, this will produce an XML file which will be used later on.

```
<?xml version="1.0"?>
<!-- Add more negative to training. -->
<opencv_storage>
<cars3 type_id="opencv-haar-classifier">
 <size>
  20 20</size>
 <stages>
  <_>
   <!-- stage 0 -->
   <trees>
    <_>
     <!-- tree 0 -->
     <_>
      <!-- root node -->
      <feature>
       <rects>
        <_>
         6 12 8 8 -1.</_>
        <_>
         6 16 8 4 2.</_></rects>
       <tilted>0</tilted></feature>
      <threshold>0.0452074706554413</threshold>
      <left_val>-0.7191650867462158</left_val>
      <right_val>0.7359663248062134</right_val></_></_>
.....
```

Xml filed created on the haar cascade classifier trainer

After creating the XML file, the proponent uses cv2.CascadeClassifier to create our new classifier:

```
cascade2 = cv2.CascadeClassifier('cars.xml')
```