Pedestrian Detection

Víctor Emanuel Ríos Martínez

Implementation on the HOG descriptor with a support vector machine to detect people in a determined area.

Table of contents

Introduction

Detecting humans is a task that has a great importance and value in the field of computer vision. For multiple purposes, from video surveillance systems to cars with automatic pilot, there have been continuous improvements in the techniques for detecting humans.

Recently, since the last decade, the techniques such as the HOG descriptor have been implemented to achieve this purpose. Along with the capabilities of a classifier, the HOG descriptor, can really determine whether an object is a human body or not. One of the most used classifiers is the SVM. An SVM (Support Vector Machine) is a supervised learning algorithm that constructs a hyperplane to classify elements.

In this work, I make an implementation of the HOG descriptor and the SVMs included in the OpenCV library for python.

Objective

Concisely, the objective of this work is to get a reasonable understanding of the methods and techniques used for people detection, in addition to get an overview of

the state of the art. This work, is an implementation, that can help to recognize people on the street and tell the drivers if there is any.

Justification

There are many applications of pedestrian detection but the importance of each one is what gives value to the work. The video surveillance systems, used in banks, airports and stores are one of the most outstanding examples, but also systems managed via automatic pilot, such as cars and motorcycles, can draw some benefit from the development of this area.

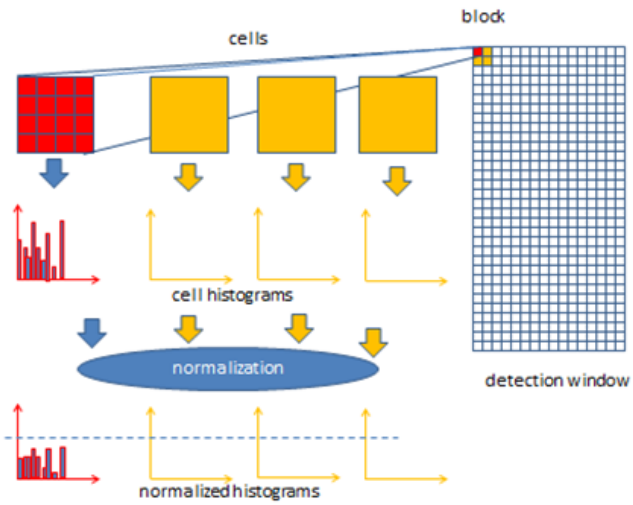
Video Surveillance Systems. Image from http://www.siebel-research.de/

Development

The essential part of the work is the implementation of the HOG descriptor with the support vector machine. The code to achieve the pedestrian detection has been considerably reduced by the OpenCV library. The detection will also consider that there is some space in the image that doesn't have to be analyzed, for this I have made a simple method to select the region of interest.

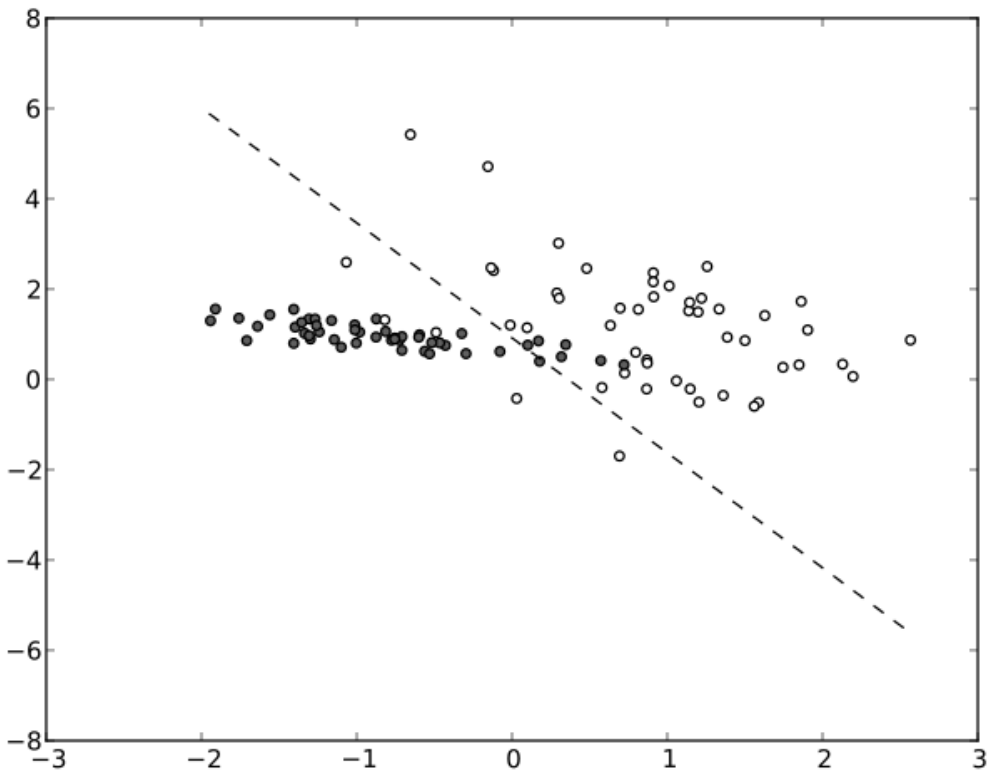
Firstly, I will explain the process to get the HOG Descriptor. The HOG descriptor (Histogram of Oriented Gradient) is based on the gradient orientation of regions in the image. Basically, this is the procedure.

1. The image is divided in cells.
2. Compute gradient orientation of each pixel.
3. The cells are discretized taking into account the gradient orientation.
4. Each cell's pixel votes for the orientation based on the gradient orientation.
5. The cells are grouped in blocks.
6. Histograms are normalized for each block.
7. The descriptor is the set of normalized histograms.



HOG descriptor. Image from https://software.intel.com

The SVM as a supervised learning algorithm needs to be trained before it can classify objects. OpenCV includes a method that returns the coefficients of the SVM for pedestrian detection.



The SVM defines a hyperplane for classification.
"Linear-svm-scatterplot" by Qwertyus Wikimedia Commons

This is part of the code that implements the HOG descriptor with the trained SVM.

```
# Create a hog descriptor object.
hog = cv2.HOGDescriptor()
# Set the coefficients for the SVM
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())
# Detect people in the image
hogParams = {'winStride': (0, 0), 'padding': (0, 0), 'scale': 1.05}
found = hog.detectMultiScale(bigger_img, **hogParams);
```

The code for selecting the region of interest (ROI) is also important.

```
def selectROI(event, x, y, flags, params):
    global roi
    if event == cv2.EVENT_LBUTTONUP:
        roi.append((x, y))
        if len(roi) == 2:
            cv2.rectangle(img, roi[0], roi[1], (255, 0, 0), 2)
```

Basically, it waits for 2 parameters, selected in a window with the mouse, that would be the region of interest.

I also used background subtraction and a gaussian filter to reduce the noise, but the HOG descriptor by itself seems to work with better results.

```
# Gaussian filter and background subtraction, both seem to negatively affect in the detection
# blur = cv2.GaussianBlur(img, (3,3), 0)
# fgmask = fgbg.apply(img, None, 0.5)
# fgmask = cv2.morphologyEx(fgmask, cv2.MORPH_OPEN, kernel)
```

Finally, the bounding boxes are drawn in the frame and the number of people detected is shown in the window.

```
# A subroutine to get different colors
col = colorsfile.color(noofpersons)

# Bounding box
for i in xrange(0, noofpersons):
    person = found[0][i].tolist()
    x=person[0]
    y=person[1]
```

```
    xf=person[0]+person[2]
    yf=person[1]+person[3]
    cv2.rectangle(img, (x, y), (xf, yf), col[i], 1)

cv2.putText(img, "Number of persons: " + str(noofpersons), (0, 30),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2)
```

Results

Selecting different areas of the same video. I can select a region of interest in the video. The selection can be restarted before starting the detection using the key “r”.

Future Work

This approach to pedestrian detection needs to be improved to handle occlusion and improve the performance. Also it is a good idea to track the detected people. Although OpenCV has the coefficients for an already trained SVM, there are SVMs better trained that can deal with occlusion and other problems in the input image.

Regarding the user interface, it should be improved to provide a better user experience, adding a toolbar with the necessary functions.

Conclusions

Pedestrian detection is a useful tool for different applications, however there are still a lot of issues that should be solved in order to get the best results. The occlusion, accessories and clothes are some of the problems that are experienced trying to detect human bodies. Using background subtraction and techniques for noise reduction did not throw better results.

HOG descriptors along with support vector machines work very well trying to detect people but it still has to be improved in the areas mentioned above.

References

Dalal, N., & Triggs, B. (2005, June). Histograms of oriented gradients for human detection. In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on (Vol. 1, pp. 886-893). IEEE.

Histogram of Oriented Gradients (HOG) Descriptor. (n.d.). Retrieved May 23, 2015, from https://software.intel.com/en-us/node/529070

Rosebrock, A. (2015, March 9). Capturing mouse click events with Python and OpenCV - PyImageSearch. Retrieved May 25, 2015, from http://www.pyimagesearch.com/2015/03/09/capturing-mouse-click-events-with-python-and-opencv/