Covariance Matrices as regional feature descriptors for vehicle classification from stationary camera

Harish N Sathishchandra, Iciar Ortega Oria de Rueda

Boston University
Department of Electrical and Computer Engineering
8 Saint Mary's Street
Boston, MA 02215
www.bu.edu/ece

May. 04, 2018

Contents

List of Figures

1 Introduction

As the number vehicles on road increases, it is getting increasingly harder to manage traffic. Vehicle classification plays a vital role in order to manage the ever increasing amount of traffic. Electronic toll collection for instance, allows vehicles to travel at normal speed during the toll payment, which helps to avoid the traffic delay at toll collection points. One of the major components of an electronic toll collection is the Automatic Vehicle Detection and Classification (AVDC) system which is important to classify the vehicle so that the toll is charged according to the vehicle classes. We plan to develop a method to distinguish between cars and trucks from a video feed of a highway.

1.1 Literature Review

Related work includes Porikli et al. [1] , where an object is tracked using a Covariance based object description. The covariance tracking method does not make any assumption on the measurement noise and the motion of the tracked objects, and provides the global optimal solution. Bhattacharya et al. [2] extends the concept of Covariance based object description to tracking spatio-temporal changes in a video. Tuzel et al. [3], proposes covariance matrices as a distance metric and describe the concept of integral images for fast computation of the covariances. However, the Region Covariance matrices are handcrafted for specific applications, and thus lack the flexibility that a deep neural network can offer. The Second Order Convolutional Neural Networks (SO-CNNs), proposed by Yu et al. [4], overcomes this problem by using a Covariance Descriptor Unit to exploit second order statistics and hence replace the fully connected layers of standard CNNs.

2 Vehicle Detection

The images being used for this project are obtained from two separate web cams (AXIS 207W Network Camera) placed in different rooms and at different viewing angles overlooking a freeway with a variety of vehicle traffic. The images were taken during a relatively low traffic period to minimize ghosting effects and overlapping vehicles and in most occasions the ambient light present was low enough to not cause excess glare or amplify the dirt on the windows. The camera produced jpeg images with a resolution of 480×640 at a frame rate of around 30 frames per second. Snapshots of the video feed were taken during different number of vehicles passing by, so that a diverse set of 20 images were obtained. A typical image obtained from the web cam is shown in fig. 1

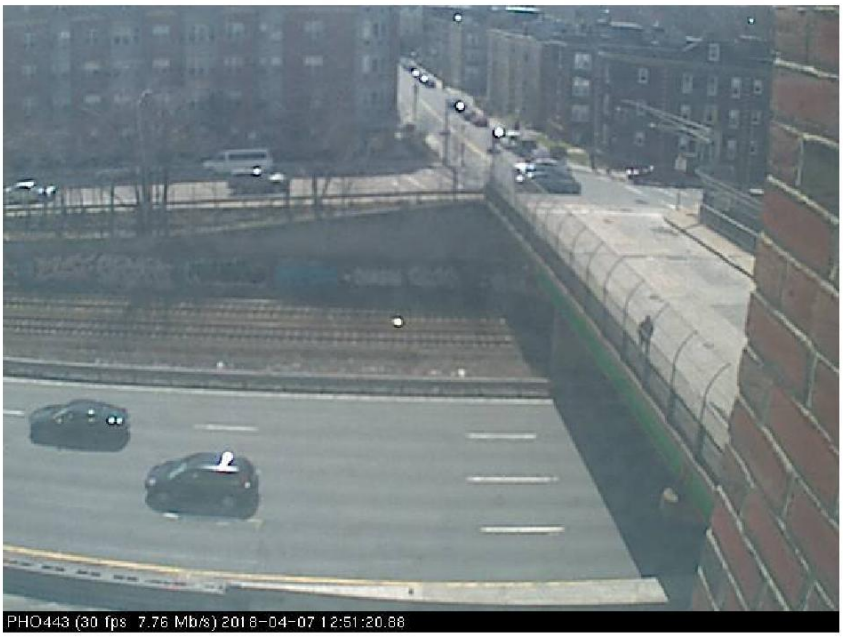

Figure 1 Original Image

2.1 Background Subtraction

Background subtraction is used to remove the information that does not pertain to the objects of interest in the image. Background subtraction is a widely used approach for detecting moving objects in videos from static cameras. We implement background subtraction on the grayscale version of the color image obtained from the web cams, by taking the weighted sum of the color channels, according to equation 1, where the coefficients are obtained from the luminosity function of the CIE standard observer.

$$I = 0.2126R + 0.7152G + 0.0722B \quad (1)$$

The rationale in the approach is that of detecting the moving objects from the difference between the current frame and a reference frame. Here, we obtain the reference frame by taking the average image from the set of images, according to equation 2. the background subtracted image is obtained according to equation 3. The reference frame for our image set and a background subtracted image is shown in fig 2 and 3.

$$B(x, y) = \frac{1}{N}\sum_{n=1}^{N} I(x, y, n) \quad (2)$$

$$I'(x, y) = I(x, y) - B(x, y) \quad (3)$$

Figure 2 Reference Image

Figure 3 Background subtracted image

2.2 Noise Reduction

The images obtained from the web cam is generally noisy, either because of dust on the windows or from noisy compression. We use a 5x5 median filter to filter out such noise. We chose the median filter for its edge preserving properties. We use a Matlab function for the median filter, which basically creates a window of given size and sorts all the values in the window in ascending order, and chooses the center value in the sorted array.

We then shift the image up by 20, so that we can filter out some of the high valued pixels, but keeping the pixels corresponding to the moving vehicles. This value of 20 was arrived at emperically, by changing pixel values with the aim of reducing unwanted noise, while keeping the pixel values related to moving vehicles. Note that this value is highly dependent on the amount of ambient light available at any given time. We then set all the negative valued pixels to 255, since the moving vehicles had the lowest luminance, and all positive valued pixels to 0, as shown in equation 4. This image is then converted to a binary image, as shown in fig 4.

$$I(x,y) = \begin{cases} 0, & I''(x,y) > 0 \\ 255, & I''(x,y) \leq 0 \end{cases} \tag{4}$$

where, $$I''(x,y) = MedianFilter(I'(x,y))$$

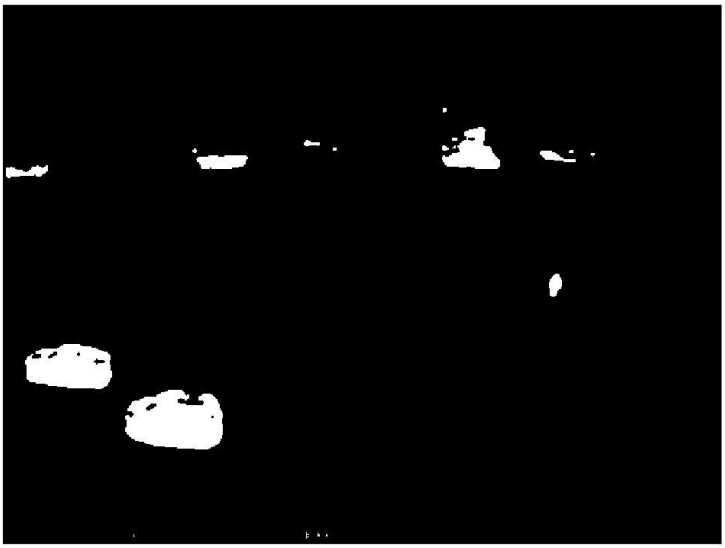

Figure 4 Binary image

2.4 Morphological Closing

The binary image shown in Figure 4 is connected enough for achieving good segmentation and classification. But, this is not always the case when there are large areas on a vehicle with large variations in intensity, such as a white colored bus with black windows. We found out that for such cases, our segmentation algorithm either identified one vehicle as two separate vehicles, or if the original vehicle was small enough, completely missed it altogether. In order to solve this problem, we decided to use morphological operators in order to fill in the gaps within the boundary of a vehicle.

Morphological operators are a collection of nonlinear operators that modify the shape or morphology of features in an image. They rely only on the relative ordering of the pixel values and not on their numerical values, and therefor are especially suited for our case of processing binary images. These operations depend on a structuring element where the

image is convolved with it, and depending on the shape and size of the element, the image shape is modified.

There are two fundamental operations, called erosion and dilation. Erosion shrinks an image by making the holes and gaps between the pixels larger. Dilation does the exact opposite, it adds pixels in the gaps, thereby filling in any gaps in the binary image. A combination of the two results in opening or closing of an image. Opening tends to eliminate small objects in the foreground, while closing removes small holes in the foreground. Since our main objective is to close the holes in the detected vehicles, we decided to use the closing operation.

The closing operation is performed by first performing dilation, followed by erosion using the same structuring element. Hence after applying the median filter to the binary image, the closing operation was done as follows:

$$I'''(x,y) \bullet S = (I'''(x,y) \oplus \mathrm{S}) \ominus \mathrm{S} \quad (5)$$

Where S is the structuring element of rectangle shape and height and width of 10 cells each. $\oplus$ and $\ominus$ symbols are dilation and erosion operators respectively. Figures 5 and 6 shows result of this operation when applied to a binary image of a large bus with black windows.

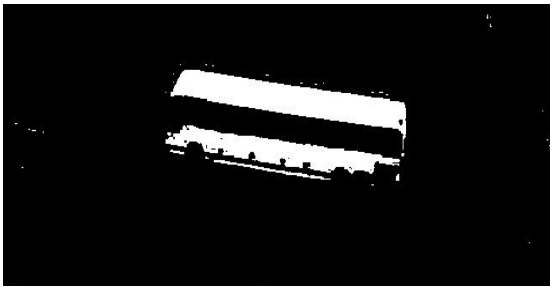

Figure 5 Before closing

Figure 6 After closing

3 Image Segmentation

We scan the entire binary image to find all the contiguous groups of activated pixels. We use Matlab's function 'BWCONNCOMP' from the image processing toolbox, with a 8-connected neighborhood to specify the connectivity for the connected components. This function first searches for an unlabeled pixel 'x'. It then uses a flood-fill algorithm to label all the pixels in the connected component containing the pixel 'x'. It then repeats the above steps until all the pixels are labeled. A bounding box is then drawn around all the connected groups with an area of above 2000px^2. Fig. 5 and 6 shows detection of a single and multiple vehicles in an image. This entire process of vehicle detection takes about 6.3 secs. For running through an image set of 20 images.

Figure 7 Single vehicle detection

Figure 8 Multiple vehicle detection

4 Image Classification

After successfully detecting vehicles in images, the next step would be to select feature vectors, from which covariance matrices would be calculated for each pixel in a region.

3.1 Feature Vectors

We use similar feature vectors as used in [3] but do not consider the color components, as a vehicle color does not tell us much about it's size. The feature vector which we decided to use is given by equation 6. The first order derivatives in the feature vector in the x and y directions is approximated by convolution with a 3x3 Sobel operator given by equation 7. The second order derivatives is approximated by convolution with a 3x3 Laplacian matrix, given by equation 8.

$$F_{x,y} = [x \quad y \quad \frac{\partial I(x,y)}{\partial x} \quad \frac{\partial I(x,y)}{\partial y} \quad \nabla^2 I(x,y)] \tag{6}$$

Where *I(x, y)* is the intensity of the pixel at location *(x, y)*

$$\frac{\partial}{\partial x} \approx \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \frac{\partial}{\partial y} \approx \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{7}$$

$$\nabla^2 \approx \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{8}$$

3.2 Covariance Matrix

The Covariance matrix C_k for a given region R_k is calculated by creating a feature vector F_i for each pixel in R_k. This is given in equation 9. These Covariance Matrices for the different regions is compared with target images' region Covariance matrices using a distance metric, given by Forstner et al. [5]. This distance metric is given in equation 10. Here, μ(u) represents the average value of the feature in the feature vector for all pixels in the region R_k. $\lambda_i(C_1,C_2)$ represents the generalized eigenvalues between C_1 and C_2.

$$C_k(u,v) = \frac{1}{N}\sum_{i=1}^{N}\big(F_i(u) - \mu(u)\big)\big(F_i(v) - \mu(v)\big)^T \tag{9}$$

$$\rho(C_1,C_2) = \sqrt{\sum_{i=1}^{n} ln^2 \lambda_i(C_1,C_2)} \tag{10}$$

3.3 Class Creation

Another image set is made, containing a large amount of diverse vehicles and overlaps to develop a library of covariance matrices divided into 2 categories – {Car, Truck}. Each class comprises of covariance matrices of 50 vehicles, which are labelled manually. We then determine what class of object a given region R_i contains, by calculating the distance between C_i and every matrix in the library. The minimum distance is then taken and the region is identified as the same class as the object it was closest too, similar to k-means clustering, with k equal to 2 in this case. Figure 9, 10, and 11 show truck classification of a moderate size truck, an 18 wheeler, and multiple car classification respectively.

Figure 9 Classification of moderate sized truck

Figure 10 Classification of 18 wheeler

Figure 11 Classification of multiple cars

5 Results

Some of the results produced by the algorithm are listed in Table 1. The sensitivity metric is defined as follows:

$$s_e = \frac{correctly\ \ identified\ \ vehicle\ \ class}{total\ \ number\ \ of\ \ vehicles} \tag{11}$$

Length	No. of cars	Sensitivity
15 frames	16	98%
Length	No. of trucks	Sensitivity
15 frames	2	100%

The table is from a fairly simple scene typical in low traffic conditions where there is minimal overlap of the vehicles. Even though the algorithm works very well, there are a few shortcomings. Firstly, when a truck enters or exits a frame, the algorithm mistakes it to be a car, due to the fact that it appears as a car because of partial visibility of the vehicle. Once the truck is in full view, it correctly labels it. Secondly, the algorithm sometimes groups 2 cars together as one vehicle. This is because of our segmentation algorithm, where due to the morphological closing operation, sufficiently nearby cars and trucks gets joined together. Thirdly, the shadows of the vehicles causes a ghosting effect, which, when strong enough causes the segmentation algorithm to detect a vehicle when there is none. The last

two problems can be solved by using a probabilistic model, such as a Markov Random field (MRF) model.

6 Conclusion and possible improvements

While executing this project, we had to make several assumptions. The first assumption is that the camera is fixed. Some assumptions being made about the size of the image set is that it is large enough that the average image is an accurate representation of the background, yet small enough in comparison to the time-scale of a day to be significantly altered by sunrise or sunset. A time sample of up to about 15 to 30 minutes is about the maximum time length for this invariance to be true and a sample of at least 50 images is required for the average image to be free of ghosting. Additionally, the cars are assumed to be moving quickly only in view for several seconds at most. Conditions such as bumper to bumper traffic would give a poor average image and a significant amount of vehicle overlap.

In addition to the above assumptions, the feature vectors are handcrafted, specific to this problem. One possible solution to this to use deep neural networks to learn these features. To that end, [4] have proposed a novel second order convolutional neural network to construct the covariance matrix from the learned features, making it a more general algorithm.

References

[1] F. Porikli, O. Tuzel, P. Meer. Covariance Tracking using Model Update Based on Means on Riemannian Manifolds. Proc. IEEE CVPR, 2006
[2] S. Bhattacharya, N. Souly, M. Shah. Covariance of Motion and Appearance Features for Human Action and Gesture Recognition. arXiv preprint arXiv:1606.05355v1, 2016
[3] O. Tuzel, F. Porikli, P. Meer. Region Covariance: A Fast Descriptor for Detection and Classification. ECCV, 2006
[4] K. Yu, M. Salzmann. Second-order Convolutional Neural Networks. arXiv preprint arXiv:1703.06817, 2017
[5] W. Forstner, B. Moonen. A metric for covariance matrices. Geodesy-The challenge of the 3rd Millennium, 2003

Appendix

Main Code:

```
cd /home/Harish/Pictures/ec520
```

read images into cell

```
imagefiles = dir('*.jpg');
nfiles = length(imagefiles);

for ii=1:nfiles
   currentfilename = imagefiles(ii).name;
   currentimage = imread(currentfilename);
   currentimage = rgb2gray(currentimage);
   currentimage = imcrop(currentimage,[1 221 640 244]);
   currentimage = double(currentimage);
   images{ii} = currentimage;
   images2{ii} = currentimage;
end
```

Image Detection

```
for jj = 1:nfiles
   if jj == 1
       sum = images{jj};
   else
       sum = sum + images{jj};
   end
end
N = 1./length(images);
B = N.*sum;
nnn = 1;
for kk = 1:nfiles
    images{kk} = images{kk} - B;  % subtract background from each frame
    images{kk} = medfilt2(images{kk},[5 5]);    % use median filter to
                                                 % eliminate noise
    images{kk} = images{kk}+20; % decrese intensity to delete high
                                % intensity noise
    thresh = images{kk};
    [row,col] = size(thresh);
    for i=1:row
        for j = 1:col
            if thresh(i,j) < 0
                thresh(i,j) = 255;
            else
                thresh(i,j) = 0;
            end
        end
    end
```

```
images1{kk} = thresh;
images1{kk} = imbinarize(images1{kk},1);
```

image segmentation

```
images1{kk} = imcrop(images1{kk},[1 34 481 149]);% crop image to remove
                                                  % unwanted detections
cov_mat = [];
cov_mat(:,:,end) = zeros(6,6);
se = strel('disk',10);
images1{kk}= imclose(images1{kk},se); % morphological closing
dd = bwconncomp(double(images1{kk})); % create bounding box
labeled = labelmatrix(dd);
k = regionprops(dd,'BoundingBox');
q = regionprops(dd,'FilledArea');

for i = 1:length(k)
    if q(i).FilledArea >= 1000 % if filled area of bounding box
                               % greater than threshold, then
                               %vehicle detected

        kk1 = k(i).BoundingBox;
        xmin = round(kk1(1));
        xmax = round(kk1(1)+kk1(3));
        ymin = round(kk1(2));
        ymax = round(kk1(2)+kk1(4));
        images2{kk} = uint8(images2{kk});
        images2{kk} = insertShape(images2{kk},'rectangle',...
        [kk1(1) kk1(2) kk1(3) kk1(4)],'Color','red','LineWidth',2);
        % draw bounding box
```

vehicle classification:

```
        a = imcrop(images1{kk},[kk1(1) kk1(2) kk1(3) kk1(4)]);
        % crop out ROI from image
        feat_vec = featureim(a); %create feature vector
        cov_mat(:,:,end+1) = gencov(a); %Calculate covariance matrix
        dist = zeros(size(cov_mat,3)-1,2);

        for ii = 2:size(cov_mat,3)
            dist(ii-1,:) = calc_dist(cov_mat(:,:,ii));
        end

        for iii = 1:size(dist,1)
            [r c] = min(dist(iii,:));

            for no=1:size(c,2)

                if c(1,no) == 1
                    images2{kk} = insertText(images2{kk},...
                    [xmax ymin],'car','FontSize',18,'BoxColor','w',...
                    'TextColor','r');
```

```
                    else
                        images2{kk} = insertText(images2{kk},...
                        [xmax ymin],'truck','FontSize',18,'BoxColor',...
                        'w','TextColor','r');


                    end
                end

            end
```

write frames to disk

```
            imshow(images2{kk});
%                 someFolder = '/home/khurshid-admin/Pictures/ec520/final3';
                  % filename=strcat(num2str(kk),'.jpg');
                  %figure;

                  fullFileName = fullfile(someFolder, filename);
%                 imwrite(images2{kk}, fullFileName);
%                 nnn = nnn+1;
%                 t = text(xmax,ymin,clas,'color','red','FontSize',12);

        end
    end

end
```

Code for Labelling dataset:

```
% code for class creation

img = imread('/home/khurshid-admin/Pictures/ec520/extra/thumb0751.jpg');
img_gray = rgb2gray(img);
figure;
imshow(img_gray);
img_crop = imcrop(img_gray);
%car
load('db1.mat');
%truck
load('db2.mat');
%junk
load('db3.mat');
% car.cov(:,:,end) = [];
% truck.cov(:,:,end) = [];
cova = gencov(img_crop);
if isempty(db2.cov)
    db2.cov(:,:,end) = cova;
else
    db2.cov(:,:,end+1) = cova;
end
```

```
save('db1.mat', 'db1')
save('db2.mat', 'db2')
save('db3.mat', 'db3')
```

Code for Feature Vector creation:

construct feature vector

```
function fd=featureim(k)

    fd=[];

    xax=[1:size(k,1)]'*ones(1,size(k,2)); % x coordinate of pixel
    yax=([1:size(k,2)]'*ones(1,size(k,1)) )';
    sobel=[[1 0 -1];[2 0 -2];[1 0 -1]];
    lap2=[[1 1 1];[1 -8 1];[1 1 1]];

    fd(:,:,end)=xax; % x coordinate of pixel
    fd(:,:,end+1)=yax; % y coordinate of pixel

    fd(:,:,end+1)=k;
    fd(:,:,end+1)=conv2(k,sobel,'same'); % first order derivative
                                         % of intensity along
                                         %horizontal direction

    fd(:,:,end+1)=conv2(k,sobel','same'); % first order derivative
                                         % of intensity along
                                         %vertical direction

    fd(:,:,end+1)=conv2(k,lap2,'same'); % second order derivative
                                         % of intensity

end
```

Code for Covariance Matrix creation:

generate covariance feature descriptor

```
function c=gencov(img)
figure(1)
imagesc(img);
[y,x]=ginput(2);
x=round(x);
y=round(y);
fi=featureim(img(x(1):x(2),y(1):y(2)));
c=cov(reshape(fi,size(fi,1)*size(fi,2),size(fi,3)));
end
```

Code for distance calculation:

```
function dist_car1=calc_dist(obj_cov)
```

```
    %car
    load('db1.mat');
    %truck
    load('db2.mat');
    %junk
    load('db3.mat');
```

for car database comparision

```
    %avg_car
    db_car = [];
    dist_car1 = [];
    db_car = zeros(6,6);
    nn = 1;

    for i =1:size(db1.cov,3)
        db_car(:,nn) =  sqrt(log(eig(obj_cov,db1.cov(:,:,i))).^2);
        nn = nn+1;
    end
    dist_car = zeros(1,6);
    nnn = 1;
    temp = 0;
    while nnn <=6
        for i = 1:size(db_car,1)
            temp = temp + db_car(i,nnn);
        end
        dist_car(1,nnn) = temp;
        nnn = nnn+1;
        temp = 0;
    end
    dist_car1(1) = min(dist_car);
```

for truck database comparision

```
    %avg_car
    db_car = [];
   % dist_car1 = [];
    db_car = zeros(6,6);
    nn = 1;
    for i =1:size(db2.cov,3)
        db_car(:,nn) =  sqrt(log(eig(obj_cov,db2.cov(:,:,i))).^2);
        nn = nn+1;
    end
    dist_car = zeros(1,6);
    nnn = 1;
    temp = 0;
    while nnn <=6
        for i = 1:size(db_car,1)
```

```
            temp = temp + db_car(i,nnn);
        end
        dist_car(1,nnn) = temp;
        nnn = nnn+1;
        temp = 0;
    end
    dist_car1(2) = min(dist_car);

end
```