Using Covariance Matrices as Feature Descriptors for Vehicle Detection from a Fixed Camera

Kevin Mader and Gil Reese

Paper submitted in partial fulfillment
of the requirements for the Final Project of
SC520 : Digital Image Processing and Communication

BOSTON UNIVERSITY

Using Covariance Matrices as Feature Descriptors for Vehicle Detection from a Fixed Camera

Kevin Mader and **Gil Reese**
Department of Electrical and Computer Engineering, Boston University
December 6th, 2006

Abstract

A method is developed to distinguish between cars and trucks present in a video feed of a highway. The method builds upon previously done work using covariance matrices as an accurate descriptor for regions. Background subtraction and other similar proven image processing techniques are used to identify the regions where the vehicles are most likely to be, and a distance metric comparing the vehicle inside the region to a fixed library of vehicles is used to determine the class of vehicle.

Introduction

There are many potential uses for object identification ranging from development of automatic key-wording in a photo library to determining the weight load on a bridge or specific road or possibly estimating air pollution by the different types of vehicles present. The specific goal for this paper is to develop a method to distinguish and count the number of cars and trucks on the road at a given time.

The problem of identifying vehicles has been mostly related to tracking uses, but several have proposed interesting approaches to the problem. Methods such as deformable template matching and template differencing (M Betke 1996) have been used for problems that necessitated real-time algorithms. Slower methods using histograms in the wavelet domain were able to detect objects from a variety of viewing angles (H. Schnelderman 2000).

The method we used was derived largely from a paper about covariance matrices as a distance metric (O. Tuzel 2006). However since the specific problem we had of identifying vehicles in a feed of images, we decided to utilize the information that could be obtained for comparing temporally separated frames. So instead of using region growing methods to identify the vehicles we used background subtraction and simple image segmentation methods to identify the regions where vehicles were located. Then we used covariance matrices on these regions to determine what kind of vehicle if any was present in the bounding box.

One of the decisions that must be made before beginning to do any kind of analysis is to determine what are the groups vehicles are being classified into. Our selection was to define smaller vehicles as cars and larger vehicles as trucks with the cutoff being around the size of a midsize SUV such as a Chevrolet Tahoe. The decision was somewhat arbitrary, but the general reasoning behind it was caused by initially the small number of trucks present in the data set we chose which would be prohibitive of doing extensive testing. Secondly though was the simplicity of the separation of car and semi-truck. The two objects are vastly different in size and demonstrating a complex method using covariance matrices could separate these two classes of vehicles would be minimally informative since a simple pixel counting method could perform the task. Thirdly choosing groups that were closer together could allow us to investigate what kind of shape information in the covariance matrices allows the algorithm to make a decision to classify a vehicle as either a car or truck.

Material and Methods

Image Acquisition The images used for this project were obtained from three separate web cams (AXIS 207W Network Camera) placed in different rooms and at different viewing angles overlooking the Massachusetts Turnpike (I-90) a 6 lane interstate 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 320$\times$240 at a frame rate of around 10 frames per second. The different angles of the cameras provided a slightly different view of the scenery especially of a light pole that partially obstructed the view of the cars. In one camera the light pole was almost unnoticeable in the other two it blocked a part of the road so cars going under the pole were partially obstructed.

Image Preprocessing A test image from the camera is rotated and then cropped from manual user input in order to establish the road and setup the cars to be orthogonal to the viewing window so a bounding box is an accurate way to express the boundaries of the car. If the car was moving in a diagonal fashion the bounding box would contain a significantly larger amount of empty space which would 'water-down' many of the calculations later possibly enough to make distinction between a car and a truck impossible.

Background Subtraction In order to remove the information that does not pertain to the objects and vehicles in the image the average image from the data set is subtracted. Given a color image set H of dimension $W \times H \times \#\ of\ Color\ Channels \times Images$ we calculate the background for the data set by the following formula

$$B(x,y,c) = \frac{1}{N}\sum_{n=1}^{N} H(x,y,c,n) \tag{1}$$

In order to calculate a given image to be used for in the processing the background image is subtracted and the imaged is converted to grayscale by averaging each of the color channels

$$H'_k(x,y) = \frac{1}{3}\sum_{c=1}^{3} |H(x,y,c,k) - B(x,y,c)| \tag{2}$$

Image Cleaning The process of identifying objects and vehicles from the image $I_k(x,y)$ is a very difficult one, but is essential for determining a car or a truck. The first step is to create a cleaned up version of the image. This is accomplished by first using a median filter with a $[5 \times 5]$ neighborhood.

$$H''_k(x,y) = \mathbf{MedianFilter}(H'_k(x,y)) \tag{3}$$

The next step is to remove all the low valued pixels by shifting the image down by 20 and removing all the negative numbers.

$$I_k(x,y) = \begin{cases} 0, & H''_k(x,y) < 10 \\ H''_k(x,y) - 10, & H''_k(x,y) \geq 10 \end{cases} \tag{4}$$

Binary Image A binary image is important for segmentation process since a grayscale image would contain too much information to use standard `MATLAB` methods. A new image is created from the image and an amplified version of the edges in the image. The **Edges** term is binary image created from doing a canny edge detection on the image.

$$I'_k(x,y) = I_k(x,y) + 10 \times \mathbf{Edges}\{I_k(x,y)\} \tag{5}$$

The image is then converted to a binary image using intensity cutoff

$$I''_k(x,y) = \begin{cases} 0, & I'_k(x,y) < \mu \\ 1, & I'_k(x,y) \geq \mu \end{cases} \tag{6}$$

$$\mu = \frac{1}{\mathrm{N} \times \mathrm{W} \times \mathrm{H}} \sum_{k'=1}^{\mathrm{N}} \sum_{x'=1}^{\mathrm{W}} \sum_{y'=1}^{\mathrm{H}} I'_{k'}(x',y') \tag{7}$$

μ is the average value over all images in the data set to prevent images absent of cars to have the noise amplified disproportionately.

Segmentation From the binary image separate objects are detected by finding all the pixels that formed contiguous groups of more than 60 pixels each of these groups was labeled as a region i out of L, size W_i by H_i, starting at point (x_i, y_i) in image $I''_k(x,y) \rightarrow R_{i,k}(x,y) = I''_k(x_i + x, y_i + y)\forall[(x,y) \leqslant (\mathrm{W}_i, \mathrm{H}_i)]$ For each of these regions a mean (a mean in a binary image represents the percentage of pixels turned on) $f_i$was calculated as follows.

$$f_i = \frac{1}{\mathrm{W}_i \times \mathrm{H}_i} \sum_{x'=1}^{\mathrm{W}_i} \sum_{y'=1}^{\mathrm{H}_i} R_{i,k}(x',y') \tag{8}$$

Since $I''_k(x,y)$ is a binary image and $R_{i,k}(x,y) \in I''_k(x,y)$ then $0 \leq f_i \leq 1$.

- If $f_i \leq 0.45$ and the value of $\mathrm{W}_i \times \mathrm{H}_i \geq 1400px^2$ the object is assumed to contain multiple cars that are out of alignment and therefor have quite a bit of empty space resulting in a low f_i value. In order to seperate these two objects a 'k-means algorithm' (JL Marroquin 1993) is used to search for two groups of pixels inside the region $R_{i,k}$ from this two new regions are created.
- If $f_i \geq 0.80$ and the value of $\mathrm{W}_i \times \mathrm{H}_i \leq 340px^2$ the object is assumed to contain a portion of a vehicle (windshield, hood, etc.). The assumption is then that there are nearby segments containing the other parts of this same car so a "k-means algorithm" (JL Marroquin 1993) is used to search for $L-1$ groups.
- If $\mathrm{W}_i \times \mathrm{H}_i \leq 200px^2$ and none of the other conditions are true the section is deleted

Each of these groups was taken and a bounding box was determined. If the number of and the samples from $I_k(x,y)$ inside the box are used to calculate the covariance matrix.

Feature Vectors The feature vector used in the Region Covariance paper (O. Tuzel 2006) requires revision since our task requires correctly identifying cars from trucks. In that paper they were trying to identify specific objects that were moving around not classes of objects. So color information is going to be ignored as trucks are not significantly differently colored than cars. The most successful feature vectors used on our data set have been.

$$\mathbf{F}_{x,y} = [\ x \quad y \quad \tfrac{\partial I(x,y)}{\partial x} \quad \tfrac{\partial I(x,y)}{\partial y} \quad \nabla^2 I(x,y)\ ] \tag{9}$$

$$\mathbf{F}_{x,y} = [\ r^2 \quad \tfrac{\partial I(x,y)}{\partial x} \quad \tfrac{\partial I(x,y)}{\partial y} \quad \nabla^2 I(x,y)\ ] \tag{10}$$

$$\mathbf{F}_{x,y} = [\ x \quad y \quad \nabla^2 I(x,y) \quad \mathbf{Edges}(I(x,y))\ ] \tag{11}$$

$r = \sqrt{(x-x_0)^2 + (y-y_0)^2}$ represents the distance the given pixel is away from the center of the region $(x_0, y_0) = (x_i +$

$\frac{W_i}{2}, y_i + \frac{H_i}{2}$). The use of R over x and y was investigated since using a rotation invariant variable would allow the car to be going the opposite direction or changing lanes and still produce a similar feature vector. The derivatives in the x and y directions are approximated by convolution with the 3×3 matrix Sobel in x and y

$$\frac{\partial}{\partial x} \approx \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \frac{\partial}{\partial y} \approx \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{12}$$

The laplacian is approximated by a convolution using the 3×3 Laplacian matrix.

$$\nabla^2 \approx \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{13}$$

Covariance Matrices The covariance matrix C_k for a given region $R_k(x, y)$ was calculated by creating a feature vector $\mathbf{F}_i$ for each point $(x, y) \to i = x + (y - 1) * \mathrm{W}_i$ in the region resulting in N total points. $\mu(u)$ represents the average value of the feature value u for all the pixels in the region.

$$\mathbf{C}_k(u, v) = \frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} (\mathbf{F}_i(u) - \mu(u))(\mathbf{F}_i(v) - \mu(v)) \tag{14}$$

The distance between two covariance matrices is calculated using a previously developed method (W. Förstner 1999).

$$\rho(\mathbf{C}_1, \mathbf{C}_2) = \sqrt{\sum_{i=1}^{n} \ln^2 \lambda_i(\mathbf{C}_1, \mathbf{C}_2)} \tag{15}$$

Where $\lambda_i(\mathbf{C}_1, \mathbf{C}_2)$ repesents the generalized eigenvalues between $\mathbf{C}_1$ and $\mathbf{C}_1$. While this metric is likely non-optimal, it satisfies the requirements for a distance metric and is easily and quickly calculated using `MATLAB`

Ontology Creation One image set was selected as containing a large amount of diversity of vehicles and overlaps to develop the library of covariance matrices divided into 4 categories $\{Car, Truck, Multiple, Junk\}$. Multiple was created to represent when two vehicles were overlapping enough to be identified by the segmentation algorithm as the same vehicle. Junk was created when the algorithm would identify just a portion of a vehicle or when the vehicle is partially out of frame. The method then determined what class of object a given region R_i contained by calculating the distance between C_i and every matrix in the library. The minimum distance was then taken and the region would be identified as the same class as the object it was closest too.

Results

Some of the results produced by the algorithm are listed in quantitative from in Table 1. The only feature vector used to produce these specific results was Equation [11]. The others were used initially but this one appeared to work the best. The sensitivity and specificity terms are defined in the appendix under Equations [16,17] respectively.

Table 1. Quantitative Results

Length	# Cars	Sensitivity	Specificity
15 frames	16	93.75%	0%

Length	# Trucks	Sensitivity	Specificity
15 frames	2	100%	100%

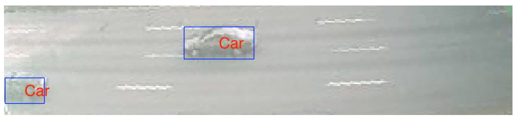
The table is from a fairly simple scene typical in low traffic conditions where there is minimal overlap of the vehicles, and for this scene it works very well only making a mistake on one frame and the mistake was in the segmentation (Figure 4) the algorithm's only mistake was to label the two partial car segments both as cars instead of junk. The specificity number could probably be greatly increased by an improved segmentation.

Figures [1,2] demonstrate the algorithm's ability to detect all the objects in the scene and properly identifying the cars. The pictures in Figure [2] show the background subtracted images with the bounding box and indentification. The ghosting sort of effects that appear as the image set gets small appears strongly in images [2(b),2(d)]. In this image set the effect was weak enough to not cause false identification of objects, but the ghosting effects can cause quite a few issues as discussed later.

Discussion

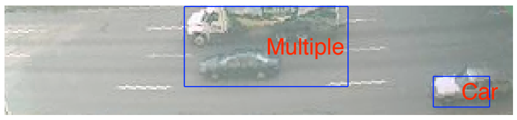
The algorithm worked suprisingly well on most of the images even difficult images as demonstrated in Figure 3. The program seemed to work very well regardless of how the camera angles and lighting changed from image set to image set. Figure 3 shows the correct identification of all vehicles in the picture using an ontology that was created using a camera looking the other direction. The algorithm success was almost entirely limited by the success of the segmentation algorithm on the image.

Shortcomings The segmentation algorithm was able to manage most of the images very well; however, several of the images managed to produce erroroneous regions that either contained multiple cars (Figure 2(d)) or regions that contained portions of cars (Figure 4). One of the largest potential problems with this method is the distances between objects within the library. Small distances between them would mean the difference in distances between a given region and an object in the car class and an object in the truck class could be quite small. With these differences being quite small the proper identification could be thrown off my small noise artifacts in the image. Although there were some cases such as Figure [5, 6] where the distance metric was not distinct enough to separate the vehicles in most cases it seemed

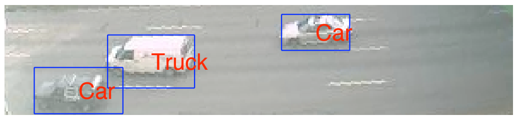


(a) Good multiple car detection

(b) Good truck detection

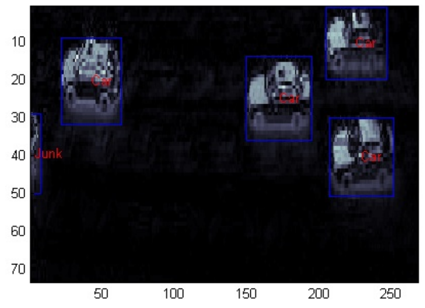


(c) Overlapping Vehicle detection

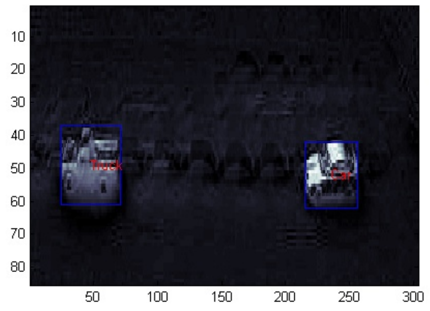


(d) Good multiple object detection

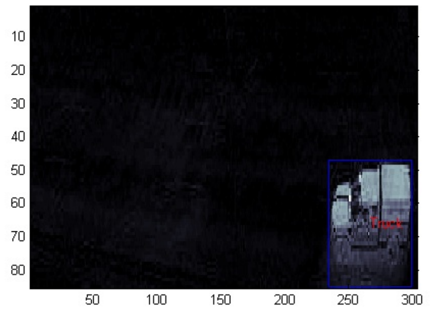
Figure 1. Normal Images



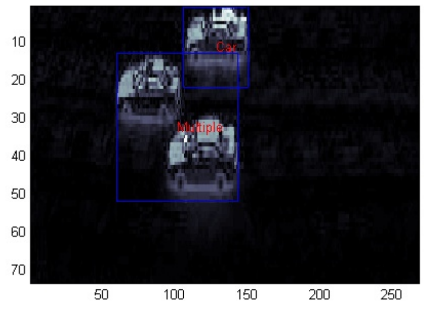
(a) Good multiple car and junk detection



(b) Good truck detection

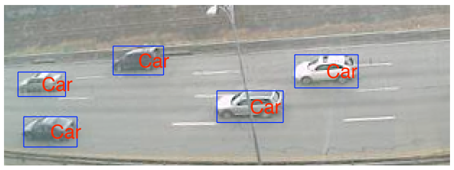


(c) Good partial truck detection

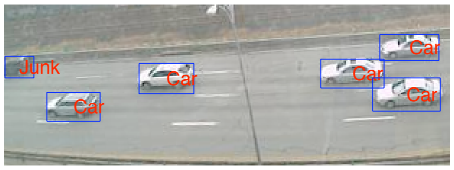


(d) Good multiple object detection

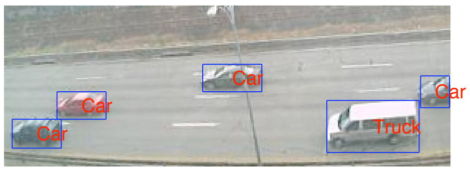
Figure 2. Background Subtracted Images



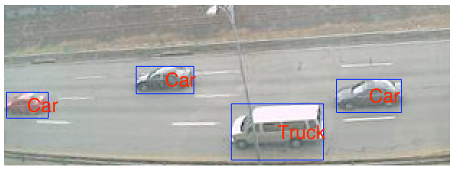
(a) Car detection with pole obstruction



(b) 3 overlapping cars detection and junk detection



(c) Truck detection



(d) Truck detection with pole obstruction

Figure 3. Difficult Images

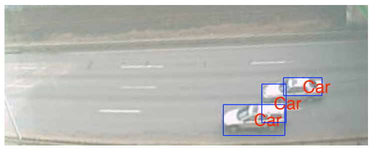


Figure 4. The two vehicles here were split into 3 vehicles

to work well.
One of the most interesting failures of the algorithm that was actually found because of a bug in the program was an object that was eqi-distant from both the car and the truck using the metric (Eq 15). The picture of this object is shown in (Figure 5). Lines on the road show up on the

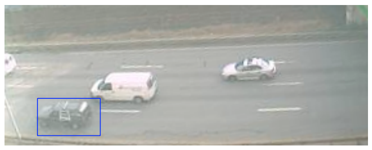


Figure 5. The vehicle equidistant (using Equation [15]) between a car and a truck (the 'eigenvehicle')

background subtracted images as lines on the vehicle. This causes possible extra edges to appear inside the car where there is nothing. This effect was noticeable in several of the images, but it did not seem to pose a problem when classifying the type of image. However it is fairly easy to imagine a configuration where this could pose a real problem.

What is a truck? For the purposes of our project a truck was really just defined as a vehicle that was noticeably larger than a sedan. So a Jeep or small pickup would not be considered a truck, but a Hummer, passenger van, and large pickup would be. Clearly a large vehicle or semi-truck would always be considered a truck, and a small sedan or a 5 passenger SUV would be considered a car. This method worked well on the vehicles which fell on the extremes of being a car or a truck but several vehicles in between were poorly identified by the algorithm. The vehicle would switch between car and truck since the distance difference between the two was quite small. A great example of this is the station wagon (Figure 6). Furthermore many car manufacturers today develop trucks to look like cars and cars to look like trucks so one would expect that these vehicles might be classified incorrectly which again reiterates the importance of having a solid definition of vehicle types for classification.

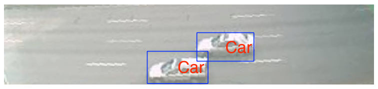


Figure 6. Station wagons were often inconsistently classified from frame to frame

Assumptions and Limitations The first assumption is the camera is fixed so the rotation and cropping procedure needs to only be done once per camera and within an image set the only changes would be related to vehicle motion. 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.

Future Work

A deeper investigation needs to be done on what really defines a car or truck from the covariance statistics. This would allow us to understand exactly which variables play the largest roles in the determination of a vehicle, furthermore this might allow for the development of a better suited metric for measuring distance instead of logs of eigenvalues it might be possible to compare several elements of the covariance matrix.

For our project the algorithm was qualitatively tweaked and optimized. While this is much easier to do than developing quantitative methods, it is probably flawed. The algorithm needs to be tested more extensive and statistics about sensitivity and specificity for car and truck detection need to be determined for this method to be useful in a real world setting. The algorithm also needs to be compared to other simpler algorithm's to determine its relative ability to properly identify cars. For example in many of the image sets it appeared that a measurement of the bounding box area would have been a closer match to the vehicle type than using the significantly more complicated and computationally intensive covariance matrix method. Also the problem is probably not properly defined. There are many more classes of vehicles than we described. There are midsize, motorcycles, and all sorts of different SUV's that were lumped into either car or truck based on their relative size which probably weakened the algorithm's ability to discern between these middle class of vehicles.

References

[H. Schnelderman 2000] H. Schnelderman, T. K. 2000. A statistical method for 3d object detection applied to faces and cars. *CVPR.*

[JL Marroquin 1993] JL Marroquin, F. G. 1993. Some extensions of the k-means algorithm for image segmentation and pattern classification. *CBCL.*

[M Betke 1996] M Betke, E. Haritaoglu, L. D. 1996. Multiple vehicle detection and tracking in hard real-time. *IVS.*

[O. Tuzel 2006] O. Tuzel, F. Porikli, P. M. 2006. Region covariance: A fast descriptor for detection and classification. *ECCV.*

[W. Förstner 1999] W. Förstner, B. M. 1999. A metric for covariance matrices. Technical report, Dept. of Geodesy and Geoinfromatics, Stuttgart University.

Appendix

Sensitivity and Specificity

Although these are generally used for medical statistics they have been used as a good measure for the tests especially in reference to proper junk identification when segmentation produces faulty regions. For these test the gold standard is a human user looking at each frame and identifying the number of cars and trucks they see, and then looking and how the program identified everything to determine the number of junk regions labeled as car or truck. A junk region would be all three regions in the splitting picture since none of them represent accurately a car (Figure 4). So the sensitivity test tests the both the segmentation and accuracy of the minimum distance finding/covariance matrix ontology. The specificity test just shows the accuracy of the minimum distance finding/covariance matrix ontology because a statistic representing how often the algorithm correctly identified the absence of a car would be too saturated because the test does work in most cases. The specificity is just how well the program labeled the regions that were found as junk. For example a region that just contained a roof would be junk or contained two portions of different cars should be junk.

$$\texttt{Sensitivity} = \frac{\textit{Correctly Identified Cars/Trucks}}{\textit{Total Number of Cars/Trucks}} \tag{16}$$

$$\texttt{Specificity} = \frac{\textit{Correctly Identified Junk}}{\textit{Junk Labeled as Cars/Trucks} + \textit{Correct Junk}} \tag{17}$$

Background Subtraction/Cleaining Code

```
 function [jk,rk]=loaddata()
d=menu('Select File Loading Method','Folder','File(s)');
if d==1
  path=[uigetdir '\'];
  files=dir([path '*.jpg']);
  filen=files.name;
else
  [filen,path]=uigetfile('*.jpg;*.JPG','JPEG Image','Multiselect','on');
  if ischar(filen)
      filen=filen;
  end
end
test=imread([path filen1]);
imshow(test)
uiwait(msgbox('Click the bottom line of the image'));
[y,x]=ginput(2)
thet=180/pi*atan(diff(x)/diff(y));
test=imrotate(test,thet,'bicubic');
imshow(test);
uiwait(msgbox('Click the boundaries of the image'));
[y,x]=ginput(2)
x=round(x);
y=round(y);
k=[];
for jb=1:length(filen)
  test=imread([path filenjb]);
  test=imrotate(test,thet,'bicubic');
  k(:,:,:,jb)=double(test(min(x):max(x),min(y):max(y),:));
end
test=mean(k,4);
jk=[];
rk=[];
for jb=1:length(filen)
  cImg=abs(k(:,:,:,jb)-test);
jk(:,:,jb)=mean(cImg,3); % convert to psuedo grayscale
```

```
  findcars(jk(:,:,jb));
  pause(.1);
end
rk=uint8(k);
```

Ground Truth/Ontology Creating Tool

```
 function [imdb,obdb]=labelcars(imgs,realim,obdb,imdb)
if nargin≤3
  %cardb
  obdb1.covs=[];
  imdb1=[];
  %truckdb
  obdb2.covs=[];
  imdb2=[];
  %multidb
  obdb3.covs=[];
  imdb3=[];
  %junkdb
  obdb4.covs=[];
  imdb4=[];
end

colormap('bone');
for k=1:size(imgs,3)
  [xmin,xmax,ymin,ymax]=findcars(imgs(:,:,k),0);
  fk=featureim(imgs(:,:,k));
  for j=1:length(xmin)
      imshow(realim(:,:,:,k));
      line([xmin(j) xmax(j)],[ymin(j) ymin(j)]);
      line([xmin(j) xmax(j)],[ymax(j) ymax(j)]);
      line([xmin(j) xmin(j)],[ymin(j) ymax(j)]);
      line([xmax(j) xmax(j)],[ymin(j) ymax(j)]);
      %[xmin(j),xmax(j),ymin(j),ymax(j)]
      %figure
      %imagesc(imgs(ymin(j):ymax(j),xmin(j):xmax(j),k))


      fi=fk(ymin(j):ymax(j),xmin(j):xmax(j),:);
      c=cov(reshape(fi,size(fi,1)*size(fi,2),size(fi,3)));
      a=menu('Identify Object','Ignore','Car','Truck','Multiple','Junk')-1;
      if a≥0
        ing=imgs(ymin(j):ymax(j),xmin(j):xmax(j),k);

        if isempty(obdba.covs)
          obdba.covs(:,:,end)=c;
          imdba.imgs1.img=ing;
        else
          obdba.covs(:,:,end+1)=c;
          imdba.imgsend+1.img=ing;
        end
      end
  end
end
```

Image Cleaning and Segmentation Code

```
 function [xmin,xmax,ymin,ymax]=findcars(imgin,dp)
if nargin<2
  dp=1;
end
imgin=medfilt2(imgin.*(imgin>10)-10,[5 5]);
img=imgin+edge(imgin,'canny',.2)*10;
bimg=bwareaopen(img>15,80);
xmin=[];
xmax=[];
ymin=[];
ymax=[];
[bimg,n]=bwlabel(bimg);
if dp
  imagesc(bimg);
end
k=1;
while k<=n
  tImg=(bimg==k);
  xI=find(mean(tImg)>0);
  yI=find(mean(tImg')>0);
  ttImg=tImg(min(yI):max(yI),min(xI):max(xI));
  pn=mean(mean(ttImg));
  if pn<.45 & (size(ttImg,1)*size(ttImg,2))>350*4 % probably 2 vehicles, lots of empty space
      disp('KMean Split');
      ibImg=ikmeans(tImg,2);
      [x,y]=find(bimg==k);
      bimg(x,y)=0; % delete the current one
      bimg=bimg+(ibImg==1)*k;
      bimg=bimg+(ibImg==2)*(n+1);
      imagesc(bimg)
      n=n+1;
  elseif pn>.8 & (size(ttImg,1)*size(ttImg,2))<340 % probably part of a vehicle
      disp('Junk Filter');
      if n>2
        bimg=ikmeans(bimg>0,n-1);
      end
      xmin=[];
      xmax=[];
      ymin=[];
      ymax=[];
      k=1;
      n=n-1;

  elseif (size(ttImg,1)*size(ttImg,2))<250
      bimg=bimg-k*(bimg==k); %remove section
      k=k+1;
  else

      xmin(end+1)=min(xI);
      xmax(end+1)=max(xI);
      ymin(end+1)=min(yI);
      ymax(end+1)=max(yI);
      if dp
        line([xmin(end) xmax(end)],[ymin(end) ymin(end)]);
        line([xmin(end) xmax(end)],[ymax(end) ymax(end)]);
        line([xmin(end) xmin(end)],[ymin(end) ymax(end)]);
        line([xmax(end) xmax(end)],[ymin(end) ymax(end)]);
      end
      k=k+1;
  end
end
```

Identification Code

```
 function identifycars(imgs,realim,obdb)
%identifycars(imgs,realim,obdb)
colormap('bone');
for k=1:size(imgs,3)
  [xmin,xmax,ymin,ymax]=findcars(imgs(:,:,k),0);
  imshow(realim(:,:,:,k));
  for j=1:length(xmin)

      line([xmin(j) xmax(j)],[ymin(j) ymin(j)]);
      line([xmin(j) xmax(j)],[ymax(j) ymax(j)]);
      line([xmin(j) xmin(j)],[ymin(j) ymax(j)]);
      line([xmax(j) xmax(j)],[ymin(j) ymax(j)]);
      %[xmin(j),xmax(j),ymin(j),ymax(j)]
      %figure
      %imagesc(imgs(ymin(j):ymax(j),xmin(j):xmax(j),k))
      cImg=imgs(ymin(j):ymax(j),xmin(j):xmax(j),k);
      fi=featureim(cImg);
      c=cov(reshape(fi,size(fi,1)*size(fi,2),size(fi,3)));
      dists=calcdist(obdb,c);
      obj=find(dists==min(dists));
      obj=obj(1);
      switch obj
        case 1
          txt='Car';
        case 2
          txt='Truck';
        case 3
          txt='Multiple';
        case 4
          txt='Junk';
        otherwise
          txt='err';
      end
      text((xmin(j)+xmax(j))/2,(ymin(j)+ymax(j))/2,txt,'Color',[1 0 0])

  end
  menu('Ready?','Yes');
end
```

K-means Segmentation Code

```
 function [segimage]=ikmeans(img,groups)
[x,y]=find(img);
```

```
u=kmeans([x';y']',groups,'dist','city');
segimage=zeros(size(img,1),size(img,2));
for k=1:groups
  %segimage(x(u==k),y(u==k))=k;
  xk=x(u==k);
  yk=y(u==k);
  for z=1:length(xk)
      segimage(xk(z),yk(z))=k;
  end
end
```

Feature Vector/Image Creation Code

```
 function fd=featureim(k);
mtlabd=0;
fd=[];
% position values
xax=[1:size(k,1)]'*ones(1,size(k,2));
yax=([1:size(k,2)]'*ones(1,size(k,1)) )';
  sobel=[[-1 2 -1];[0 0 0];[1 2 1]];
  lap1=[[0 -1 0];[-1 4 -1];[0 -1 0]];
  lap2=[[-1 -1 -1];[-1 8 -1];[-1 -1 -1]];
%fd(:,:,1)=sqrt(xax.^2+yax.^2); % First axis is distance from center
fd(:,:,end)=xax;
fd(:,:,end+1)=yax;
%image intesity
fd(:,:,end+1)=k;

if mtlabd==1 % diff command
  % first order derivative
  fd(:,:,end+1)=zeros(size(k,1),size(k,2));
  fd(2:end,:,end)=abs(diff(k,1,1));
  fd(:,:,end+1)=zeros(size(k,1),size(k,2));
  fd(:,2:end,end)=abs(diff(k,1,2));
  % second order derivatives
  fd(:,:,end+1)=zeros(size(k,1),size(k,2));
  fd(3:end,:,end)=abs(diff(k,2,1));
  fd(:,:,end+1)=zeros(size(k,1),size(k,2));
  fd(:,3:end,end)=abs(diff(k,2,2));
elseif mtlabd==2

  fd(:,:,end+1)=conv2(k,sobel,'same'); %vertical
  fd(:,:,end+1)=conv2(k,sobel','same'); %horizontal
  % fd(:,:,end+1)=conv2(k,lap1,'same'); %first laplacian approximation
  fd(:,:,end+1)=conv2(k,lap2,'same'); %second laplacian approximation (includes diagonals)
else
  fd(:,:,end+1)=conv2(k,lap2,'same');
  fd(:,:,end+1)=edge(k,'canny',.2);
end
```

Covariance Matrix from Feature Vector Code

```
 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)));
```