Vehicle Classification using IPCP and EsKNN algorithm for Surveillance Camera

Atyanta Nika Rumaksari

Department of Computer Engineering,
Faculty of Electronic dan Computer Engineering,
Satya Wacana Christian University, Salatiga
atyanta.rumaksari@uksw.edu

Abstract
This paper presents a new approach vehicles detection and classification. In these works, a system was created to detect the vehicles and automatically classifying them into three desired classes as Sedan, SUV, and Truck. Datasets from changedetect.net were used for testing the method. The proposed approach combines Incremental Principle Component Pursuit (IPCP) background subtraction for vehicle detection and it complies with Ensemble subspace - Nearest Neighbor (EsKNN) classifier. A combination of background subtraction and classification of filtering background is used in order to focus on vehicle's features extraction using HOG corner detections. Then, this feature extraction is classified using an ensemble-based classifiers technique. The choice of classifier depends on features data characteristics, which they were formed in certain unique distance shape stationary relative between classes. The proposed method is evaluated using three datasets of common highway surveillance video. Comparing with other direct detection and classification technique our method has achieved outstanding result. The proposed approach delivers the accuracy of 96.5%, the highest among the tested methods. Experimental results show the outstanding performance of the proposed method.

Keywords: vehicles detection, vehicle classification, background subtraction, ensemble classifiers, intelligent transportation system.

1. Introduction

Automatic vehicle classification has become an important subject in intelligent transportation systems (ITS) in the last decades. In order to classify the vehicles, one needs to detect vehicles first. This automatic system is needed because there are increasing growth in road and vehicles that are greater than the manual dispatch system. Vehicle detection is the most important system in ITS [1]. It has happened because detection is the phase before doing classification. Thus, better detection will make better classification. In previous research, sensors such as laser, lidars, or radar was used to detect the vehicles. With emerging technology of computer vision, nowadays become a great advantage of simplicity and cost effective of vehicle detector. Computer vision is used for detecting object other than specific vehicles and classifying them more accurate based on its features than any other previous passive sensors. Different type of traffic, number of cars, and accident information record can be extracted only using single camera surveillance.

Vehicle detection in computer vision need to adapt with change of background condition and sudden luminance changing. Not only adaptation capability needed by ITS computer vision based, but also it should adapt to computationally effective of real-time processing.

To overcome this limitation, a new system based on Incremental Principle Component Pursuit (IPCP) background subtraction for vehicle detection is proposed and combined with Ensemble subspace k-Nearest Neighbor (EsKNN) classifier for detecting and classifying vehicles based on its classes. IPCP nowadays, become the most prominent background subtraction technique [2] [3]. It happen because the work of robust-PCA method use one frame for initialization. Therefore, it leads to low computation load. The result leads to applicable for robust and real-time applications.

2. Related Work

In recent years, some prior researchers found a way to detect object and classify them using pixel-wise methods. Zhao [4] proposed method for counting by segmenting the vehicles using local semantic regions by exploiting the temporal co-occurrence of local motions. Even though unsupervised clustering method is used for extracting path of trajectories, this method unable to distinguish pedestrian and vehicles. The other method of vehicles detection using motion estimation based detector was introduced by [5]. They develop vehicles detection and counting based upon Taylor series approximation with embedded virtual entering and virtual bboxes. Proposed method [5] uses optical flow and block matching based on vehicle detection and tracking. This object's features were extracted based on the difference of motion pixels. The main disadvantage of this method is unable to predict vehicles whenever background changing or sudden luminance happens. [6] used background subtraction technique by modeling background with averaging several frames to get empty foreground. Edge detections were employed for extracting object's features. [6] also used an aspect pixel's ratio area for vehicles classifications. Same with it, [7] created vehicles detection using a robust system of automatic threshold-via technique of shadow removal using Otsu's algorithm. The classification has been done by analyzing the area of ROI. One fundamental problem of this method is aspect ratio variant when it used for classification features, although it used a threshold for classification process, there was a condition when a different type of vehicles have same ratio with predicted object, so it will rise false positive rates. Moreover, feature extractions were depending on how far an object from the camera, thus the system will not robust in every condition. This method was inspired by [8] that used corner detection-based as an object's features and Markov chain Monte Carlo method for 3D volume estimation.

The development a similar method was attempted, which used corner detection as features. In this proposed method, HOG features was used instead of color segmentation technique. However, such method has different perspective that need to be considered. [9] created another new approach of classification by predicting the vehicle's class using cascade-regressor-based from the width and height of the segmented object area. For improving classification [9] used a Bayesian Poisson regression technique. This idea was quite different from our method because it used a different technique based on different problem point a view. With focusing foreground without disturbed by background on how features extraction is processing, one will get an effective classification. That was our hypothesis.

3. Proposed Method

In this work, a method to detect and classify different type of vehicles is proposed. Diagram process is shown in Fig. 1. There are two major processes, which is vehicle detection via background subtraction and vehicle classification using Ensemble KNN. From [9] it was difficult to detect the vehicle in low-quality images directly, so motion based background subtraction technique was used to overcome it.

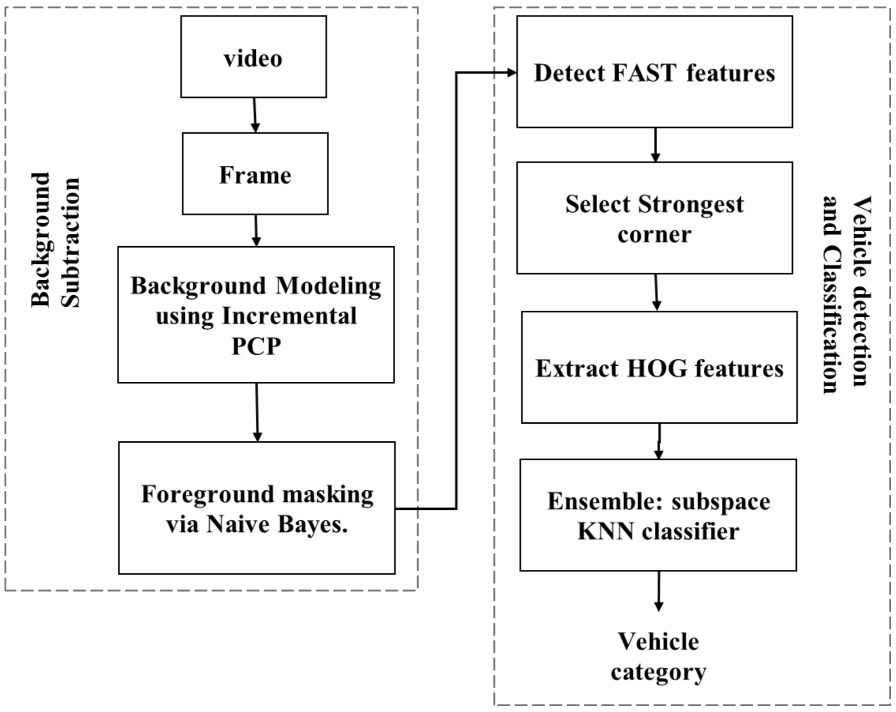

Background Subtraction
video
Frame
Background Modeling using Incremental PCP
Foreground masking via Naive Bayes.
Detect FAST features
Select Strongest corner
Extract HOG features
Ensemble: subspace KNN classifier
Vehicle category
Vehicle detection and Classification

Fig. 1 System Diagram of Proposed Method

Our first step proposed method is creating background modeling using an incremental component pursuit algorithm by giving a trivial initialization step, it can help reference background model to converge near the right background. These activities allow the process to model background one frame in a row. The next process is vehicle classification using ensemble subspace classifier. In the next section, a further step in detail of the proposed methods is explained.

3.1. Background Subtraction using IPCP

In this proposed method, Paul Rodriguez's algorithm [3] of incremental Principle component analysis was used which can deal with input video in incremental fashion without concerning much in batch processing. Since this algorithm has trivial initialization step, it can help reference background model to converge in the right background. This trivial solution helps accelerate the convergent. The PCP based algorithm are further explained in Algorithm 1.

From the table, one can see that element process in (1)-(2) are the characterization of input, it started with getting the frame D which it has m, n row and column dimension, λ as regularization parameter of sparse matrix S, number of inner loop iteration iL

background form BL, and variable m for incrementing the process according to k until m greater or equal with BL. After all the input are being prepared, initialization process can be started by defining $D(:,1:k_o)$ as all frame which being store by column wise format.

```
Input:   $D\epsilon\mathbb{R}^{m\times n}$, regulatization parameter $\lambda$, number of loop $iL$
background from $BL$, $m = k_o$.
Initialization:   : $L + S = D(:,1:k_o)$  with  initial  rank  $r$ ,
$[U_r, \Sigma_r, V_r] = partialSVD\,(L, r)$.
Output:   $[U_k, \Sigma_k, V_k]$, $L$ and $S$

Process:
For $k = k_o + 1:n$ Do
m=m+1;
$[U_k, \Sigma_k, V_k] = incSVD\,(D(:,k), v_{k-1}, \Sigma_{k-1}, V_{k-1})$
      For $j = 1:iL$ Do
                    $L(:,k) = V_k(:,1:r).\Sigma_k.V_k(end,:)^T$;
                    $S(:,k) = shrink\,(D(:,k) - L(:,k), \lambda)$;
               If $j == iL$ then BREAK;
               $[V_k, \Sigma_k, V_k] = repSVD\,(D(:,k), S(:.k).V(:,k), V_k, \Sigma_k, V_k)$;
     end For $j$

   If $m \geq BL$
   Then downSVD($1^{st}$ column, $U_k, \Sigma_k, V_k$)

      if   $\frac{\|L(:,k) - L(:,(k-1))\|_2^2}{\|L(:,k-1)\|} \geq \tau$
     Then   $m = k_o$. Goes to Filler Procedure:
end For $k$

Filler Procedure:
Compute : $[U, \Sigma] = thinQR\big(D(:,1)\big)$, Set $V = I_1$
Compute : $[U, \Sigma, V] = incSVD(D(:,k), U, \Sigma, V)$ for $k \in [2, r]$
Compute : $[U, \Sigma, V] = incSVD(D(:,k), U, \Sigma, V)$ for $k \in [r + 1, k_0]$
```

Algorithm 1. Incremental Principle Component Pursuit

Process using $partialSVD$ of matrix background L with its rank r for having real complex matrix. Output U_r and V_r are unitary matrices and Σ_r is rectangular diagonal matrix. $partialSVD$ is an algorithm founded by Sabine Van Huffel which it using to compute a basis subspace of a matrix, which it corresponds to its smallest singular value [10]. Having $[U_r, \Sigma_r, V_r]$ from previous computation, the process can be started from initialize incremental variable k from $k_0 + 1$ until n numbers of column matrix frame D, foreground S and background L.

Set a variable m for increment. From [3], set $r = 1$ and $r_0 = 1$. Even though the result SVD gave premature result, it still can be improved at each iteration of k in incremental SVD algorithm $incSVD$ (line 10). From [3], one might have information that setting value of r was set as $r \in [2, 8]$ since the data from static camera were already acquired. Inside iteration k, iteration j is formed for calculating $L, S,$ and $[V_k, \Sigma_k, V_k]$ as much as iL numbers. This procedures are the key of incremental PCP, that has the same result as batch ordinary PCP algorithm [3]. When frame $bL + 1$ are called and processed, $downdate$ function is called, thus its result will same with those whose compute via batch algorithm. $downdate$ function is used to compute thin $SVD(D) = U_1\Sigma_1V_1^T$ with r singular values. It also employ Gram-Schmidt orthonormalization procedure w.r.t V_0. This procedure operates *downdating* the contribution of the last column of the matrix when computing thin SVD [3]. It was stated that: Given to solve PCP problem in the future frame d_k, thus one shall minimize:

$$\frac{1}{2}\|L_k + S_k - D_k\|_L^2 + \lambda\|S_k\|_1 \text{ s.t. } rank(L_k) \le r \tag{1}$$

Using the minimizer of :

$$S^{j+1} = \underset{S}{\arg\min}\|L^{j+1} + S - D\|_F^2 + \lambda\|S\|_1 \tag{2}$$

or it is be called lean SVD or $shrinkSVD$ for solving **Error! Reference source not found.**. Procedure (14) stated that if variable j equal with number of iL then looping variable j is beraked. For the next procedure (16) current $[V_k, \Sigma_k, V_k]$ are called and these are being replaced with new result from $thin\ SVD(D(:,k), S(:,k))$.

After all procedure internal looping j from 1 until iL already finished, one might continue the computation, started from checking whether m greather than background looping bL, if the result is $true$ then algorithm $dwnSVD$ of first column background frame is being processed. In some cases, one can assume that there are stationary background changes, or it might change slowly, in reality this is not long lasting. However, *downdate* process can be used as filter to wipe out old data and replace it using the background L estimation process.

Other scenario of sudden background change (e.g. sudden illumination change, moving background because of moving object, etc.) is being handled by operation (23)-(32). Initialization process from equation (23), when it condition is $true$ which is greater than threshold τ, variable m is change to k_0 and filler procedures are being followed. The procedures has two major function that are $thinQR$ and $incSVD$. The difference between (30) and (31) are dimension of variable k, in (30) dimension of k equal with $[2, r]$ and dimension of k in (31) is $[r+1, k_0]$.

3.2. Foreground Masking

This process is creating masking by defining probability distribution function or histogram of image foreground S. From characteristic of output frame D, it is found that number of background pixel greater than number of foreground pixel. From the finding characteristics above, one can design filter as stated by Algorithm 3 from finding pdf from Algorithm 2.

```
Input: foreground $S$
Initialization:
        [Nrows,Ncols]=size(S);
        S = reshape(S,[Ncols*Nrows, 1]);
        S = sort(S,'ascend');
        Buffer=Ncols*Nrows;  i=1; Count=0;
Output: $Y$
Process:
        while isempty(XX)||sum(isnan(XX))~=Ncols*Nrows
            TMP = S==S(1);
            Y(i,1)=S(1);
            Y(i,2)= $\frac{\sum TMP}{Buff}$
            S = S(Not(TMP));
            i=i+1;
            XX(length(XX)+1:length(XX)+sum(TMP))=nan;
        End while
```

Algorithm 2. Probability Density Function of an Image

Algorithm 2 similar with finding histogram of an image. It begun from initialize size of image S and count the similar pixel value if it is true than store it into TMP variable

(10). Information Y were divided into two categories, which it's column wise. First column is the pixel value and second column is the number of this value occur in an image. Procedure (12) is the probability of occurrence.

While Algorithm 3 is the filter. It remove highest occurrence probability pixel. It is considered as background and it results new binary image D_b as our masking. Range filter is being set from 90% maximum dominant pixel until 110% maximum dominant pixel. Is represents input image, background subtraction output, masking image, and reconstructed output. Result's visualization can be seen on Fig. 2 (c). Fig. 2 is resulted from dataset that was given by changedetection.net [11] and Fig. 3 is the filter pdf design with mean is $maxN$ and standard deviation is 0.1.

```
Input : $Y$, image $S$
Output : $D_b$ masking for filter remaining background.
Process:
    $[\sim, maxN(x,y)] = \max(Y)$ ;
    $D_b = 0.9 * S(maxN) \leq S \leq 1.1 * S(maxN)$;
    $if\ D_b == True\ then\ Db(x,y) = 0; else\ Db(x,y) = 1;$
```

Algorithm 3. Filter remaining background

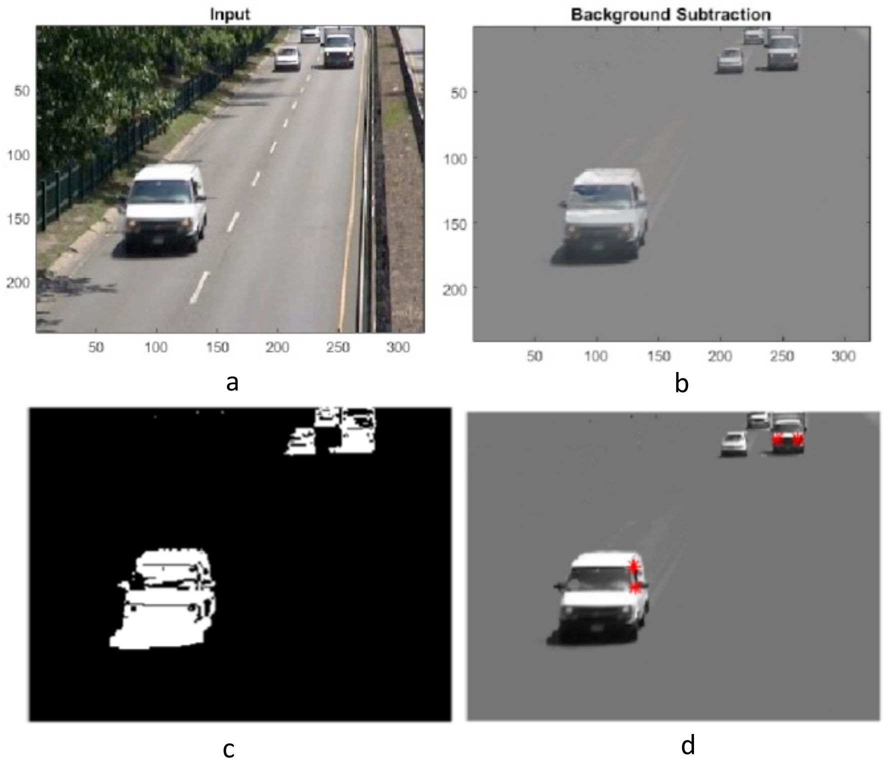


Fig. 2. Result of Algorithm 1,2, and 3. (a). Image input, (b) Background subtraction, (c) Masking image, (d) constructed output by input image times (c) add with (b). (d) is used for corner detection since it has high contrast edge.

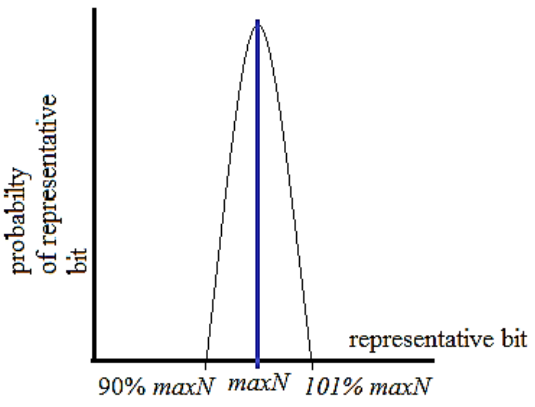


Fig. 3 Filter design with 0.1 of standard deviation σ

3.3. Detect FAST Features

Features from accelerated segment test (FAST) was algorithm proposed by Edward Rosten and Tom Drummond [12]. This algorithm is used for finding interest point of an image. Interest point is defined as high local information content and it is repeatable between images in the frames. In this proposed method, the interest point as a corner of an object was defined. Here are the brief steps of the FAST algorithms [12]. This *FAST* algorithm started with input of p pixel image, with initialization select pixel p with intensity I_t to be zero at first initialization, and select threshold τ to be minimum 20% from the total image. Process of Algorithm 4 is started from (7)-(8) with taking a pixel in the circle point rounded by 16 pixels. Compare the intensity of middle (x,y) point of interest and taking the logic if first comparison intensity and middle have more than three pixels greater than threshold τ then the candidate of interest point is exist. Stability of this candidate is being checked by if three resulted intensity values I_1, I_5, I_9, I_{13} are not above or below $I_p + \tau$, then remove p as interest point and vise versa. Finally repeating this procedure for all whole pixels.

```
Input : $p$ pixel of an image
Output : Corner points $I_i$
Initialization :
Select pixel $p$ with intensity $I_i$ to be zero at first.
Select threshold $\tau$ to be at least 20% of the total image.
Process:
Consider a pixel in the weight point middle $x,y$ is rounded by circle of 16 pixels.
Compare intensity in the middle $(x,y)$ and the intensity pixel 1,5, 9, and 13.
$if(i \geq 3\ of\ this\ 4\ pixel\ (1,5,9, and\ 13)) \geq \tau\ Then: I_t = 1.$
$if(i \geq 3\ of\ -I_1, I_5, I_9, I_9\ are\ not\ above\ or\ below\ I_p$ then
$p\ is\ not\ interest\ point.$ Else $I_p is\ the\ interest\ point.$
```

Algorithm 4. *FAST* algorithm

3.4. Select the Strongest Corner

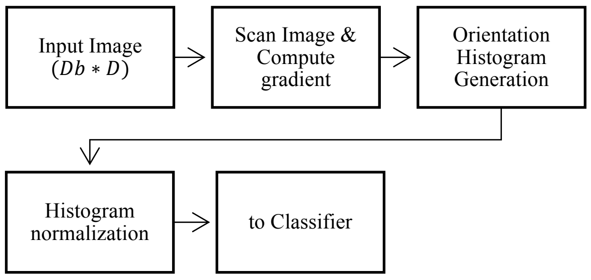
Basic ideas of the strongest corner selection is that each finding corner has nominal value, thus one can store and sort this value in ascend form. By selecting required number of corners one can perform simple matrix selection with respect on location and its nominal value. This result will be inputted into next step which is HOG features extraction. By using blob analysis, crop every object and analyze the corner's location. Corner points are the features of an object and these are being used for classification. This algorithm is defined into two categories. First, filtering all corners with $Blob$ analysis function. This function will filter each location of respected union pixel that having area greater than threshold as an object. Second, filtering corner results with the maximum length of 15 corners. The explanations of Algorithm 5 will be explained in the following section. It starts with input $L_{x,y}$ as corner location and $\widehat{L_i}$ nominal value that correspond with the corners from previous FAST algorithm. The desired output is BLOB variable that has multidimensional information such as unfiltered corner information in first dimension, new sorted ascend-wise corner information in second dimension, and filtered corner's location in third dimension. In this proposed method, the corner was limited into 15 strongest points in the exact object location. Procedure (7)-(10) in Algorithm 5 explain about initialization. Procedure (6) is initializing the object into bboxes with respect to $Blob$ function number of centroids. Next, the algorithm were executed with two looping standards such as $i = 1 \leq i \leq number\ of\ object\ in\ bbox$ and $j = 1 \leq j \leq the\ length\ of\ corners$. By defining logic in procedure (14) and (15) that limit corner location L_x and L_y should be lower than maximum row of bbox rectangle and lower than maximum column of bbox rectangle respectively, the next procedure (16)-(21) can be processed. This process was set up variable $Buff$ and $Buff2$ for location of corner point that lies inside the bboxes. Those looping are processed until its finished. The next procedure (26)-(37) are used to filter whenever the length of matrix $Buff2$ greater than 15. If its condition is $true$ then pick only 15 strongest corners. This logic is defined by procedure (29). After all logics were being processed, BLOB variable was introduced to store the desired results. These were being done in procedure (37)-(40).

```
Input: $L_{x,y}$ corner location of an image , $\hat{L}_i$ nominal value that
correspond with corners.
Output: BLOB variable that has object's information such as
filtered corners in exact object based on blob information
bounding box.
Initialization: $bbox = initBlob(Db)$;
[Row,~]=size(bbox);
[Row2,~]=size(Loc);
Set variable BLOB, Buff, Buff2, and Buff3 as an array with
respective row-column,
Process:
for i=1:Row
for j=1:Row2
if $L_{x,y}$ (j,1)>=bbox(i,1))&($L_{x,y}$(j,1)<=(bbox(i,1)+bbox(i,3))
if $L_{x,y}$ (j,2)>=bbox(i,2))&( $L_{x,y}$ (j,2)<=(bbox(i,2)+bbox(i,4)
     set Buff(k,1) = $L_{x,y}$ (j,1);
     set Buff(k,2) = $L_{x,y}$ (j,2);
     set Buff2(k,1)= $\hat{L}_i$ (j,1);
     increment k=k+1;
     set $L_{x,y}$ (j,1) and $L_{x,y}$ (j,2) and $\hat{L}_i$ (j,1) = nan;
   end if
     end if
        end for j
Check Buff2's size should be $\geq 2$ and max allowed number is 15
     if length(Buff2)>=2
            [Buff2,idx]=sort (Buff2,'descend');
            Buff = Buff(idx,:);
            if length(Buff2)>=15
            Buff3(1:15,1:2) = Buff(1:15,:);
            else
                [szx,~]=size(Buff);
                Buff3(1:szx,:) = Buff(:,:);
            end if
            Buff = Buff(1:2,:);
            Buff2 = Buff2(1:2);
      end if
        BLOB{i,2}=Buff;
        BLOB{i,3}=Buff2;
        BLOB{i,4}=Buff3;
    end for i
```

Algorithm 5. Selecting strongest corner

3.5. Extract HOG Features

HOG features is an descriptors for features extraction. The descriptor was made up from M*N cells covering the image kernel in a grid form. Each cell is being represented by a edge orientations histogram. The number of discretized edge orientations was a parameter (normaly 9). The visualization form of histogram is a 'star'. It was showing the the histogram edge orientation's strength: stronger a specific orientation, makes longer to its relative with the others. Fig. 4 shows the HOG algorithm parameter using HOG features. Normally calculation compute floating point calculation, by using Mizuno algorithm [13] it can be simplified as integer point using round up method.



Input Image ($Db * D$)
Scan Image & Compute gradient
Orientation Histogram Generation
Histogram normalization
to Classifier

Fig. 4 HOG algorithm parameter

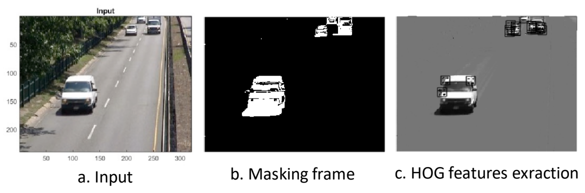


Fig. 5 Visualization of HOG features extraction.

From [13] it stated that kernel of scanning image is created by using image-cell-division arrangement, so that histogram of gradient orientation can be accumulating over the pixels of cell. For better stability regarding with illumination change, histogram normalization was done by gathering a measure of local histogram energy over the block of cell above. The result then, is ready for classification process.

3.6. Ensemble Subspace KNN

Ensemble classifier is defined as a method for combining classifiers for getting a better result. In this proposed method one of ensemble method was chosen using KNN as its classifier [14]. Out-of-sample accuracy was used for selecting the classifiers based upon their individual performance. The selected classifiers are then combined in sequential basis. Starting from the best model and followed by other collective performances on a validation data set.

Algorithm 6 shows the pseudocode algorithm of ensemble KNN based on [14]. This algorithm require two input which are labeled dataset and HOG features from previous process (Section **Error! Reference source not found.**). For the initialization, dataset was divided into two categories. First is training dataset and second is test dataset. The label is used for validation scheme. Process of this algorithm also divided by three major processes. They are compute KNNs for every samples, selecting the best model based on second quartile accuracies, and the final step is the fusion between existing model with the resulted best model based on their collective performance. The final result will classify data test based on trained KNN model.

Procedure for compute KNNs can be seen at (10)-(21) in Algorithm 6. Looping i from $1\ to\ m$ number of HOG features. Each of step i do the following procedures. Started from select random sample of l features, replace random sample with sample of training data and defining variable B_i as bootstrap sample. Store and save calculation's residue from ith of B_i. It's continued by process the calculation using KNN algorithm to create C_i. Then the accuracy of each resuled KNN are stored into buffer ACC.

```
Input: Dataset of $Dout$, HOG
Output: Classifier output matrix.
Initialization:
Randomly split the training data into two parts, first is data
training and secondly data test. Data training is used for
validation parts.
Process:
I.    Compute the KNNs
For $i = 1{:}m$
  i. Select random sample of $l$ features from HOG of total $d$
     $HOG$ features
 ii. Select random sample of size $n$ with replacement from training
     data. $B_i$ is the bootstrap sample used for creating ith model
     based on $l$ features.
iii. Store and save the residue from ith bootstrap sample as $OOB(i)$.
 iv. Process using $KNN$ algorithm to create $C_i$
  v. Find the accuracy of $C_i$ using $OOB_i$ and store it into variable
     buffer $ACC$.
end for $i$
II.   Select the best Model
For $j = 1{:}m$
 If $ACC(j) > Q_2$ then
 Select $C_j$, where $Q_2$ is the second quartile of accuracies in all
 $h$ models.
 else
 Remove $C_j$
 end if
end for j
III.  Do the Fusion between existing model with the resulted best
   model based on collective performance
   Arrange the selected model, and sort it based on their performance
   rank. Initialize the best model with highest accuracy to be above.
   For $q = 2{:}h$
     if $BrierScore(q) < BrierScore(q-1)$
       Select $qth$ of $KNN$ models, where $BrierScore(q)$ is having the $q$
       model and $BrierScore(q-1)$ is the score that not having the $q$
       model after applying validation.
     else
       Do nothing
     end if
   end for $q$
```

Algorithm 6. Ensemble subspace KNN

Next procedure is selecting the best model, procedure (22)-(29) will explain the algorithm. Selection will happen when buffer accuracy $\mathrm{ACC(j)}$ is greater than second quartile accuracy of all h models. Whenever accuracy jth is $false$ then the procedure of removing accuracy $jth\ C$ is active. After all iteration are finished, then fusion the best model and existing model is processing by arrange the selected model in iteration $q = 2 < q < h$. Each of iteration q calculates the $Brierscore(q)$ and $Brierscore(q-1)$. Function $Brierscore$ of q and $q-1$ are score for validation for having q model and not having q model. Best model based upon $Brierscore$ then is chosen.

4. Experiments

Three datasets from [11] were randomly selected, they are Highway, Streetlight, and TwoPositionPTZcam datasets. Datasets were chosen based on highway characteristics of surveillance camera. Fig. 6 shows three example of datasets. Image size of this video will be standardize at 320x240 pixels. It also required transformation into grayscale, because the original input is in RGB format. For the training data, crop the object using blob analyzer and label it manually. As much as 3,447 sample data from this process were collected. In order to get higher results, features from Highway and TwoPositionPTZcam dataset were gathered for analyzing and creating threshold. Then, choose the strongest corner features and use its information for detecting object in StreetLight dataset. Finally, these features with StreetLight dataset were trained with portion 30% for training and 70% of for testing. A combination of several classification methods was used for benchmarking purposes. In our proposed method, the dataset used three labels; they are Sedan, Suv, and Truck, since these vehicles are all exist in chosen dataset [11]

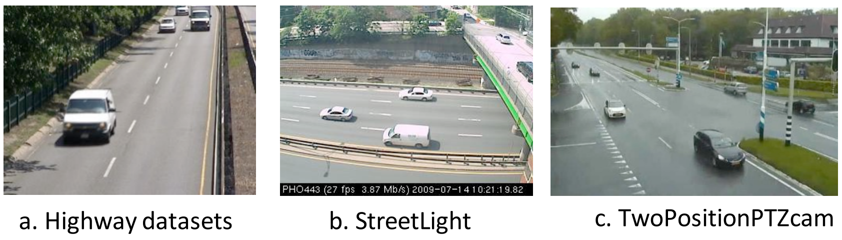


Fig. 6. Sample of three datasets

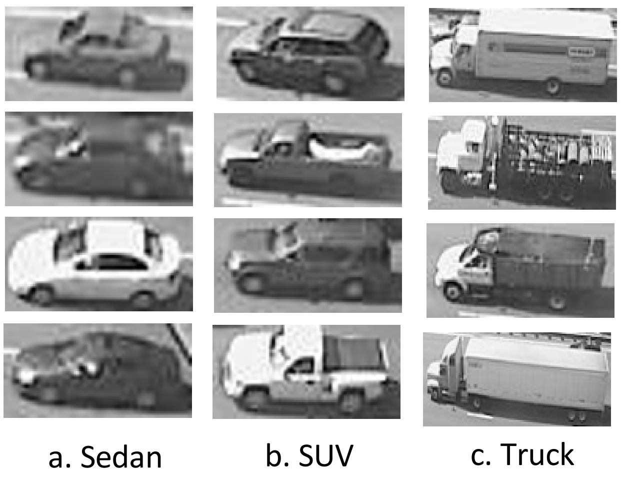


Fig. 7. Sample of cropping and labeling result

Table 1 Confusion Matrix for Ensemble: subspace KNN

True Class	**SUV**	1188(96.1%)	14(1.1%)	34(2.8%)
	Sedan	30(5.5%)	510(92.7%)	10(1.8%)
	Truck	29(1.7%)	2(0.1%)	1630(98.1%)
		SUV	**Sedan**	**Truck**
	Predicted Class			

The algorithm was compared to 19 other methods which is shown in Table 2. Therefore, based on all features, the best classification accuracy rates for three classes are our Ensemble subspace KNN are 96.5%. Which is corresponding to confusion matrix in Table 1. From it, one can see that there are 30 vehicles of predicted class SUV is false classified as Sedan and 29 vehicles of predicted class SUV is false classified as Truck. There are 14 vehicles of predicted class Sedan are false classified as SUV and 2 vehicles of predicted class Sedan false classified as Truck. There are 34 vehicles of class Truck that are false classified as SUV and 10 vehicles of class Truck that are false classified as Sedan.

From Table 1, one can see that there are correct classification of class SUV, Sedan and Truck as 1188 or 96.1%, 510 or 92.7%, and 1630 or 98.1% respectively. Automatically, ROC curve of our proposed method will stand the highest place with area under curve (AUC) all classes above 0.99, it shows at Fig. 10 (a), (b), and (c).

Table 2. Result of classifications using various methods

Method	Sub-name	Accuracy
SVM	Fine Gaussian	59.00%
SVM	Cubic SVM	91.80%
SVM	Quadratic SVM	85.50%
SVM	Linear SVM	66.80%
SVM	Coarse Gaussian SVM	65.10%
SVM	Medium Gaussian SVM	88.00%
Tree	Simple	59.50%
Tree	Medium	64.60%
Tree	Complex	68.90%
KNN	Fine KNN	95.50%
KNN	Medium KNN	91.20%
KNN	Coarse KNN	72.20%
KNN	Cosine KNN	88.90%
KNN	Cubic KNN	91.00%
KNN	Weighted KNN	93.20%
Ensemble	Boosted Tree	58.70%
Ensemble	Bagged Tree	85.80%
Ensemble	Subspace Discriminant	64.70%
Ensemble	RU Boasted Trees	40.60%
Ensemble	Subspace KNN	**96.50%**

Our comparison methods were divided into four main methods, they are SVM, Tree, KNN, and Ensemble methods. For Cubic SVM, Medium KNN, Weighed KNN and Cubic KNN were having more than 90% accuracy results. The worst accuracy was resulted by RU Boasted Trees classifier.

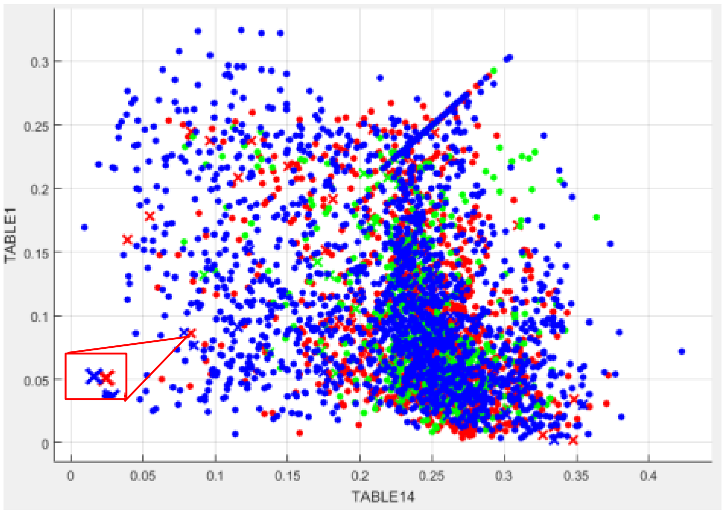


Fig. 8. Sample of scattered data from features in 1 and features 14 of HOG features. Red is SUV, Green is Sedan, and Blue is Truck. Cross mark are wrong classification.

From scattered matrix of HOG features in Fig. 8, it can be seen that data are scattered in a way that their centroids are equal with their mean. Because of this characteristics multi-dimension of data feature's distribution that are having unique stationary pattern in every dimension, our ensemble subspace KNN algorithm are chosen as classifier. This hypothesis is proven by our outstanding result at 96.5% above all methods.

Fig. 9 shows that TPR/FNR rates of our methods was lowest at 3.9% of SUV, Fine KNN method was the lowest at 7.1 % of Sedan, and Fine Gaussian SVM was the lowest at1.9 % of Truck. Our classifier pick the highest Brier score from ensemble process and the optimal model will best fit the distribution data. However, there were outliers that miss classified as other class. It sees at Fig. 8 (red box) there were miss classified datum of Blue class and Red class that exist in same distance from class's KNN centroid. The highest TPR/FNR rate goes to RU Boasted trees with 100% rate of SUV, Coarse Gaussian SVM with 100% rate of Sedan, and RU Boasted trees with 45.5% rate of truck. Truck is the easiest object to classify than the others, it is because it has distinct box shape. SUV and Sedan have almost the same shape in term of appearance. Thus, SUV and sedan have higher difficulty than Truck. KNN family classifiers were being credited as best classifier among other methods, it shows that in average this classifiers got 88.67%.

Table 3 Legend of TPR/FNR chart

1	2	3	4	5
Subspace KNN	Boosted Tree	Bagged Tree	Subspace discrimin ant	RU boasted trees
6	7	8	9	10
fine KNN	medium KNN	Coarse KNN	Cosine KNN	Cubic KNN
11	12	13	14	15
Weighted KNN	Tree Complex	Tree Medium	Tree Simple	medium Gaussian SVM
16	17	18	19	20
Coarse Gaussian SVM	Linear SVM	Quadratic SVM	Cubic SVM	Fine Gaussian SVM

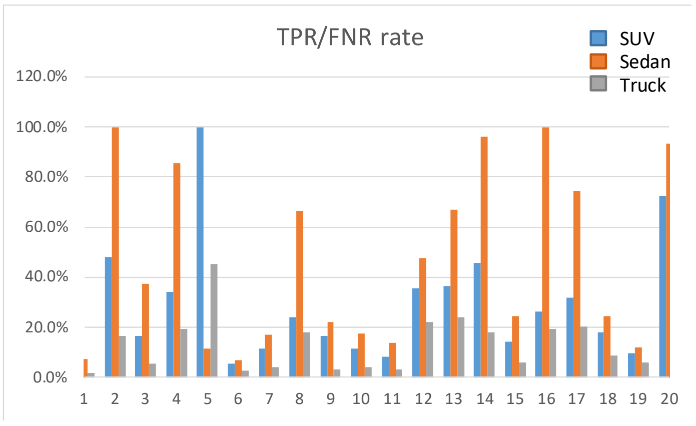


Fig. 9. Chart of TPR/FNR rates

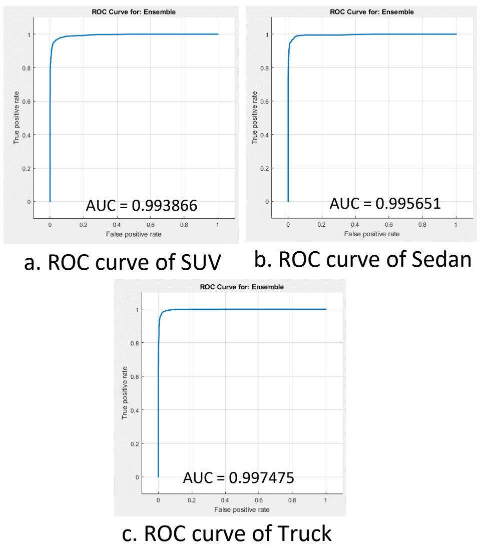


Fig. 10. ROC curve of SUV (a), Sedan (b), and Truck (c)

Our algorithm is currently trained and evaluated with different condition of dataset. Fig. 6 (a), (b) and (c) show the difference of vehicle orientation. The success of our algorithm lies in the feature extraction method. It uses corner detection HOG instead of color or textures. Because of corner detection invariant from illumination and change, the classifier easily process the classification.

5. Conclusion

In this paper, an Ensemble KNN classifier based on IPCP background subtraction was presented. Different from another direct classifier method our method is succeeded to focus on extracting object's corner features and filter undesired pixel which classified as background. Our method is also, can be implemented with a robust scenario, it is proven from the capability of employing IPCP as a real-time background subtraction method combining with Ensemble KNN classifier. Our method achieved the highest accuracy of 96.5% among another method. It is shown that our method can implicitly deal with occlusion since it has distinct vehicles based on HOG features. For the next experiment, one can focus on the occlusion scenario in a traffic jam. In further research that are many areas which can be improved such as vehicle counter, vehicle behavior based on tracking, real-time hardware implementation, and vehicle ID automatic registration.

6. References

[1] S. Kamkar and R. Safabakhsh, "Vehicle detection, counting, and classification in various conditions," *IET Intelligent Transport Systems,* vol. 10, no. 6, pp. 406-413, 2016.

[2] T. Bouwmans, "Traditional and recent approaches in background modeling for foreground detection: An overview," *Computer Science Review,* Vols. 11-12, pp. 31-66, May 2014.

[3] P. Rodriuez and B. Wohlberg, "Incremental Principal Component Pursuit for Video

Background Modeling," *Journal Mathematical Imaging and Vision,* vol. 55, no. 1, pp. 1-18, 2016.

[4] R. Zhao and X. Wang, "Counting Vehicles from Sematic Regions," *IEEE Transactions on Intelligent Transportation Systems,* vol. 14, no. 2, pp. 1016-1022, June 2013.

[5] P. Prommool, S. Auephanwiriyakul and N. Theera-Umpon, "Vision-based Automatic Vehicle Counting System Using Motion Estimation with Taylor Series Approximation," in *6th IEEE International Conference on Control System,* Penang, Malaysia, 2016.

[6] C. Roopashree and T. Sateesh kumar, "Vehicle Detection and Counting," *International Journal of Electrical and Electronics Engineers,* vol. 07, no. 1, pp. 160-166, 2015.

[7] D. Li, B. Liang and W. Zhang, "Real-time Moving Vehicle Detection, Tracking, and Counting System Implemented with OpenCV," in *IEEE conference on Computer Vision,* Chinna, 2014.

[8] L. Unzueta, M. Nieto, A. Cortes, B. Javier, O. Otaegui and P. Sanchez, "Adaptive Multicue Background Subtraction for Robust Vehicle Counting and Classification," *IEEE Transactions on Intelligent Transportation Systems,* vol. 13, no. 2, pp. 527-540, June 2012.

[9] M. Liang, X. Huang, C. Chung-Hao, X. Chen and A. Tokuta, "Counting and Classification of Highway Vehicles by Regression Analysis," *IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS,* vol. 16, no. 5, pp. 2878-2888, 2015.

[10] S. V. Huffel, "partial Singular Value Decomposition Algorithm," *Journal of Computational and Applied Mathematics,* vol. 33, no. 1, pp. 105-112, 1990.

[11] N. Goyette, P.-M. Jodoin, F. Porikli, J. Jonrad and P. Ishwar, "Changedetection.net: A new change detection bechmark dataset," 2014. [Online]. Available: www.changedetection.net. [Accessed 23 April 2017].

[12] E. Rosten and T. Drummond, "Machine Learning for High-Speed Corner Detection," *Computer Vision – ECCV,* vol. 3951, pp. 430-443, 2006.

[13] K. Mizuno, Y. Terachi, K. Takagi, S. Izumi, H. Kawaguci and M. Yoshimoto, "ARCHITECTURAL STUDY OF HOG FEATURE EXTRACTION PROCESSOR FOR REAL-TIME OBJECT DETECTION," in *IEEE Workshop on Signal Processing Systems,* Japan, 2012.

[14] A. Gul, A. Perpereglou, Z. Khan, O. Mahmoud, M. Miftahuddin, W. Adler and B. Lausen, "Ensemble of a subset of kNN classifiers," *Adv Data Anal Classif,* pp. 1-14, 2016.