Chapter 6

Localization of License Plate for moving Vehicles

6.1 Introduction

Plate localization is a very critical phase of VNPR system. Wrong localization will result into incorrect plate recognition. Trackers are used to limit the search region to certain areas in an image. Arth *et al.* [182] described the method in which license plate is detected using confidence related predictions. However these approaches do not give hundred percent confidences when vehicle is in motion.

In all the stated approaches, license plate localization is the primary task that precedes the reading of plate characters during the important process of license plate recognition [87]. The plate localization algorithm attempts to find the exact location of the boundary of the plate in the input image. From the literature survey, license plate recognition is not only used in intelligent transportation system technology for counting the vehicles and their recognition, but also identifying the uniqueness of vehicles [87]. The survey also lists the other application of license plate recognition that includes traffic management and control, electronic toll collection, law enforcement, counter terrorism, autonomous vehicle navigation, and collision avoidance systems as applications of license plate recognition [87]. Li-

* The entire chapter in the form of paper has been communicated in "Third International Conference on Soft Computing, Artificial Intelligence and Applications (SAI-2014)".

cense plate localization is performed with the aid of features present in the license plates. The license plates are identified by their unique features. The feature extraction depends on the quality of the image. For example, if the license plate is very small, any possibility of distinguishing spaces between characters is also very small. The requirements of features and conditions in the image are presented.

When car is in moving state, complexity to localize the license plate of the car adds further. We therefore focused our research towards proposing an approach to localize the license place when the vehicle is in motion that does not depend on the detailed geometry and can be fitted in the constraint environment.

Our approach is based on decomposing the captured image into small set of characteristic feature based on the eigen value; we call as "eigenilicenseplate", which may be thought of as the principal components of the initial training set of vehicle license plate. Recognition is performed by projecting new image into subspace and then classifies the license plate with the position of known individuals.

6.2 Motion Analysis

Human beings are capable to analyze the information of a moving image. Researchers had tried to build this capability in the computer-based systems. These systems analyze the motion of the image in two phases. In the first phase, the motion of the feature points is computed. If the motion of the image is slow, the instantaneous motion of the feature patterns can be defined. The vector field representing the motion is called as optical flow. When the motion of the image is fast, the displacements of important features in the image can be computed. This is known as correspondence. In the second phase, the motion parameters are estimated using either optical flow or correspondence. It is difficult to estimate the motion parameters due to the following reasons [183]:

1. The relationship between the 3-D motion parameters and the optical flow is nonlinear.

2. The unknowns are in 5-D space.

The useful information from the images can be computed either using global or local methods. As the name represents, local methods use local information: pixel or its neighbourhood while global methods use Hough transform techniques. There are some well known standard image processing techniques that include edge detection, Hough transform, contour following, motion from correspondence, optical flow using the motion constraint equation, pattern recognition [184], least square methods, etc. It has been seen that global methods are more robust to noisy inputs.

There are two methods through which Optical flow can be computed, gradient-based methods and energy model-based methods.

Gradient-based methods [185] were proposed by Horn and Schunck and are based on image in motion constraint equation. These methods search the superlative parametric geometric transform and minimize the square of difference between image intensities over the whole image. Depending on the updated motion parameters, there exist several formulations of gradient methods. The motion parameters are updated by incrementing the motion parameters [186] or by incrementing the wrap matrix.

Let the two images $I_1(x, y)$ and $I_2(x, y)$ are overlapping as shown in figure 6.1(a).

Gradient methods give the estimation of the motion parameters m_p according to the equation 6.1

$$m_p^* = \underset{m_p}{argmin} \sum_{(x_1,y_1)\in S} [I_1(x_i^{(1)}, y_i^{(1)}) - I_2(x_i^{(2)}, y_i^{(2)})]^2 \tag{6.1}$$

where,

$$x_i^{(2)} = f(x_i^{(1)}, y_i^{(1)}, p)$$
$$y_i^{(2)} = g(x_i^{(1)}, y_i^{(1)}, p)$$

S: set of coordinates of pixels that are common to I_1 and I_2 in $I_1's$ coordinates.

p: estimated parameter vector.

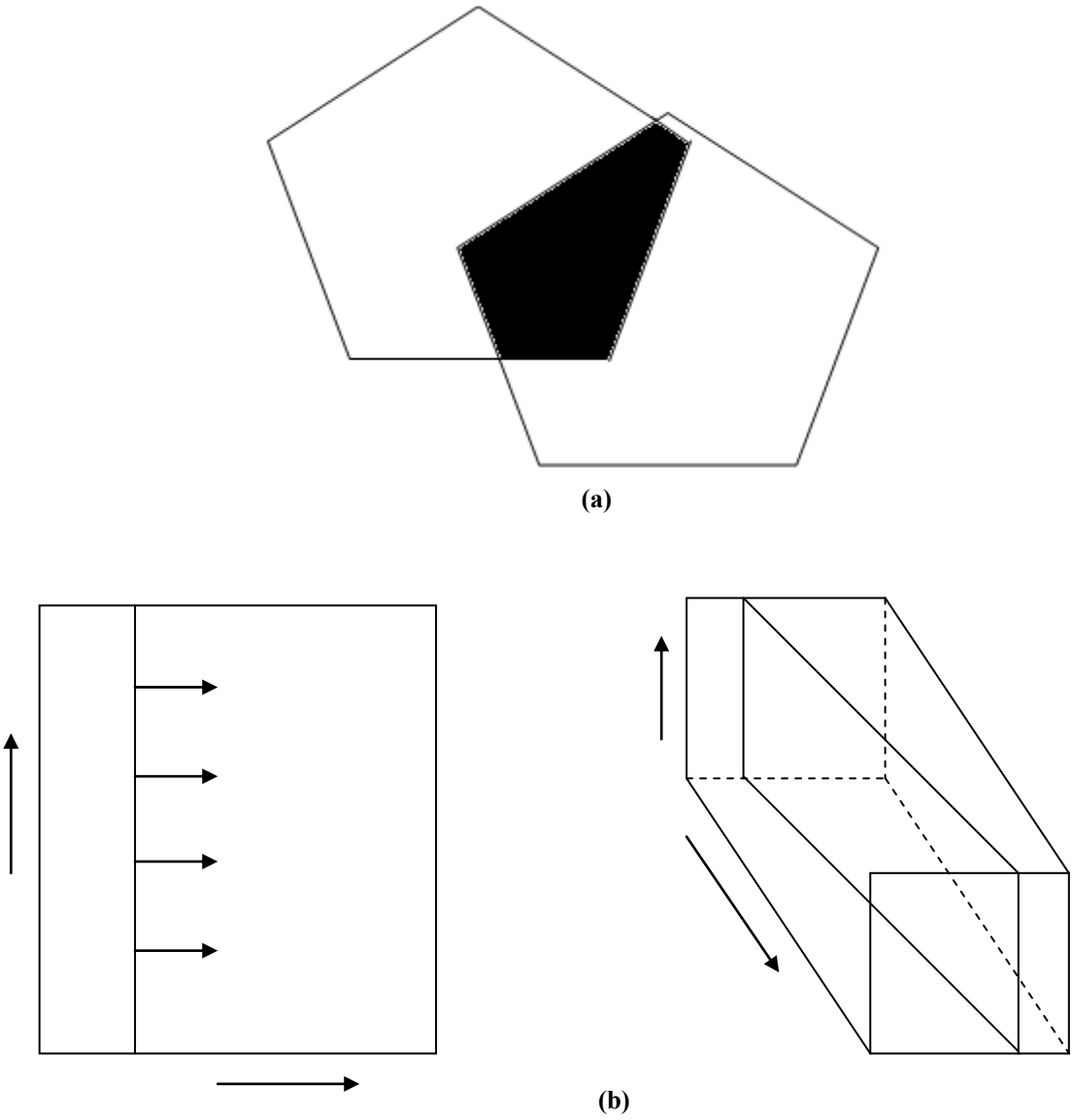

Figure 6.1: (a) Translation of Image. (b) orientation in (x,y,t) space.

The comparison of various optical flow computation techniques are given in [187]. Energy-based model methods are capable to overcome the spatiotemporally oriented filters. Translation appears as orientation in the x-y-t domain [188], also known as spatiotemporal. This translation can be detected.

Let a line AB be moving along x axis. The orientation in (x, y, t) space is shown in figure 6.1(b). The velocity is inversely proportional to the slope of the plane. There are filters to sense the orientation; hence the velocity can be computed. To construct simple neural mechanisms and to process the images

on a machine in simple manner, such filters must be spatiotemporally separable. But spatiotemporally filters are phase sensitive, which can be overcome with the combination of more filters like a set of oriented Gabor filters one with cosine phase and the other with sine phase. A one-dimensional sine phase Gabor filter is given by the equation 6.2.

$$g(t) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{t^2}{2\sigma^2}} sin(2\pi\omega t) \tag{6.2}$$

For computing the correspondence, many algorithms have been developed. An algorithm to match features at multiple resolutions using Laplacian pyramid [189] is proposed. In [190], a gradient based method and feature based method has been proposed.

It is possible to track each pixel from frame to frame, for every pixel in a blob. Some authors have proposed a method to detect the blobs in a video stream [192]. This method subtracts the background using a dynamic updating background model. The method is as follows.

1. Do smoothening of each frame with a 3×3 Gaussian filter to remove the video noise.

2. $B_0 = I_0$, initialize background model $B_n(x)$.

3. Using equation 6.3, generate a binary motion mask image $M_n(x)$, for each frame.

$$M_n(x) = \begin{cases} 1, & |I_n(x) - B_{n-1}(x)| > T \\ 0, & |I_n(x) - B_{n-1}(x)| \leq T \end{cases} \tag{6.3}$$

   Here T is a suitable threshold.

4. Update non moving pixels using Infinite Impulse Response(IIR) filter to update the view, using equation 6.4.

$$B_n(x) = \begin{cases} B_{n-1}(x), & M_n(x) = 1 \\ \alpha I_n(x) + (1-\alpha) B_{n-1}(x), & M_n(x) = 0 \end{cases} \tag{6.4}$$

where, α is the filter's time constant parameter. Moving pixels are aggregated using a connected component approach to extract the individual blob.

6.3 Computing Eigen License Plate

A 2-D image can be treated as vector by concatenating the rows of the image matrix, shown in equation 6.5. Equation 6.5(a) represents an $n \times m$ image, $p_(i, j)$ is pixel information. Equation 6.5(b) represents a $n \times m$ vector formed by concatenating the rows of image together.

$$I = \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,m} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n,1} & p_{n,2} & \cdots & p_{n,m} \end{bmatrix} \tag{6.5a}$$

$$I = \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,m} & p_{2,1} & \cdots & p_{n,m} \end{bmatrix} \tag{6.5b}$$

The instances of an image can be represented by an nm-dimensional vector X.

$$X = VY, where$$

$$X = [x_1 \quad x_2 \ldots x_n]^T$$

$$V = \begin{bmatrix} V_{1,1} & V_{1,2} & \cdots & V_{1,n} \\ V_{2,1} & V_{2,2} & \cdots & V_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ V_{n,1} & V_{n,2} & \cdots & V_{n,n} \end{bmatrix}$$

$$Y = [y_1 \quad y_2 \ldots y_n]^T$$

Here n is very large even for small images but since a small number of features are enough to characterize a set of images, it is efficient to approximate X using $m(< n)$ columns of V to give:

$$\hat{X}(m) = \sum_{i=1}^{m} y_i v_i, \quad v_i's \text{ are the column vector of V}$$

The mean square error is given by $||X - \hat{X}(m)||^2$. We know that the best vectors $v_1, v_2, \ldots, v_m$ are unit eigenvectors associated with m largest eigenvalues of the covariant matrix of X, which is given by:

$$\sum_X = [(X - E(X))(X - E(X))^T]$$

Then the features $f_1, f_2, \ldots, f_m$ can be easily computed from

$$f_i = v_i^T(X - E(X)), \quad i = 1, 2, \ldots, m$$

This projection is called as Karhunen-Loeve projection [112] and PCA, has been used to represent and recognize face images and v_i is the ith coordinate of the image in the new space, which came to be the principal component. To determine m, the number of features to use, we first rank the eigenvalues of $\sum_X, \lambda_1, \lambda_2, \ldots, \lambda_m$ in non-increasing order. If m features are used ($m < n$), the mean-square error is simply the sum of eigenvalues not used,

$$\sum_{i=m+1}^{n} \lambda_i$$

This is a criterion to determine how many features are needed to sufficiently represent a license plate. We can choose m such that sum of these unused eigen values is less than some fixed percentage P of the sum of the entire set. So satisfying m,

$$\frac{\sum_{i=m+1}^{n} \lambda_i}{\sum_{i=1}^{n} \lambda_i} < P$$

If $P = 4 - 5\%$, a good reduction of features is obtained. Also let $\Omega = [v_1 v_2 \ldots v_m]^T$. So Ω describes the contribution of each eigenlicenseplate in representing the image by treating the eigenlicenseplate as a basis set for image. The simplest method for determining the licenseplate class that provides the best de-

scription of an input image is to find the licenseplate class k that minimizes the Euclidean distance

$$\epsilon_k = ||\Omega - \Omega_k||$$

Ω_k: vector describing kth licenseplate class. If $\epsilon_k < \theta_\epsilon$ (threshold), licenseplate belongs to class k.

We can approximate $\sum_X$ with sample scattered matrix $S = uu^T, u = [u_1 u_2 \dots u_k] \& u_i = X_i - \bar{X}_i$, for k training images. S is of order $n \times n$. If $k < n, S$ degenerates. However we can find the eigensystem of the $k \times k$ matrix $u^T u$.

$$u^T u w_i = \lambda_i w_i$$

λ_i- eigenvalue; w_i- associated eigenvector.

Pre-multiplying by u,

$$u u^T u w_i = \lambda_i u w_i$$

Hence uw_i is the eigenvector of S with eigenvalue λ_i. If number of samples available is more than the image dimensions, then eigensystem of S can be computed directly.

The eigenvectors corresponding to non-zero eigenvalues of covariance matrix produce an orthonormal basis for the subspace within which most image data can be represented with a small amount of error. The eigenvalues are sorted from high to low according to their corresponding eigenvalues. The eigenvector associated with the largest eigenvalue reflects the greatest variance in the image i.e. roughly 90% of the total variance is contained in the first $5 - 10\%$ of the dimensions.

6.4 Locating License Plate

Vehicles are constantly moving on the roads, in such case we device five steps to track the moving vehicle in a static environment, given in figure 6.2.

6.4.1 Step1- Capturing image

In this step the image of the car along with static view is taken.

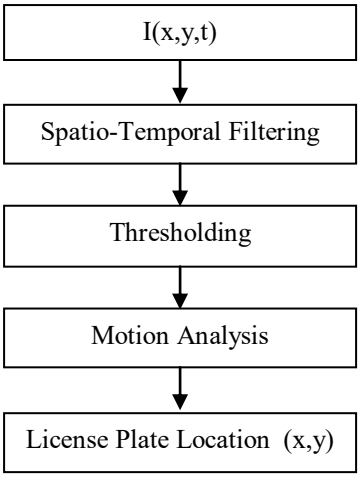

Figure 6.2: Vehicle License Plate Localization

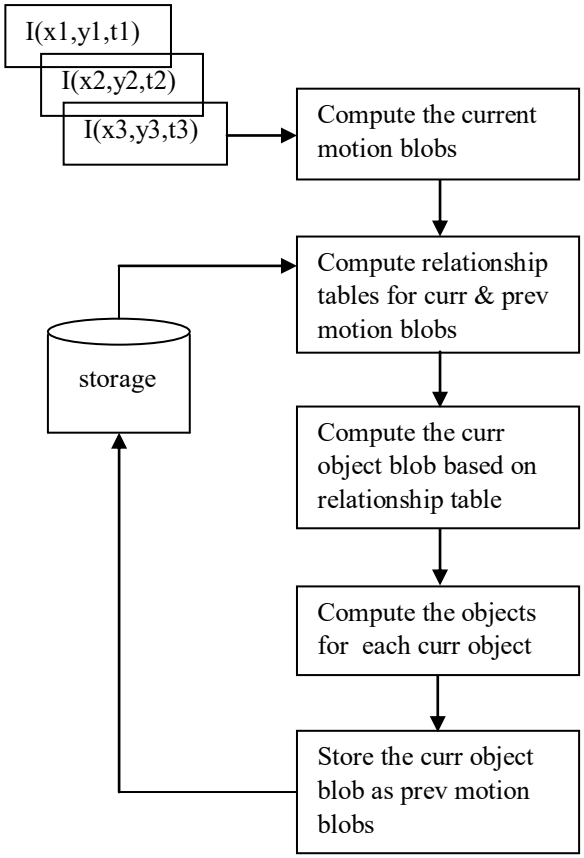

Figure 6.3: Motion detection and tracking blobs

6.4.2 Step2- Spatio-Temporal Filtering

It brings out image locations that change with time, so 'moving cars' light up in the filtered image. The spatiotemporally-filter at a distance x and time step t is

given by

$$f_{ST}(x,t) = \phi.f_S(x,t) + (1-\phi).f_T(x,t)$$

where the falloff parameter ϕ specifies the trade-off between spatial filtering f_S and temporal filtering f_T, with $f_T(x,t)$ propagating filtered distances from the previous time step $t-1$ to the current time step using motion compensation [191].

6.4.3 Step3- Thresholding

It produces the binary image.

6.4.4 Step4- Motion Analysis

In this step, we analyze motion blobs for tracking as shown in figure 6.3 [192].

6.4.5 Step5- License Plate Localization/Recognition

To localize the license plate simple rules are applied like "the license plate is of rectangular size and is located in the down side," and is small upper blob above two wheels, which are rotating with respect to the car itself. We can make use of the license plate space also to locate license plate in image. Images of license plate region do not change radically when projected into the license plate as shown in figure 6.4(a).

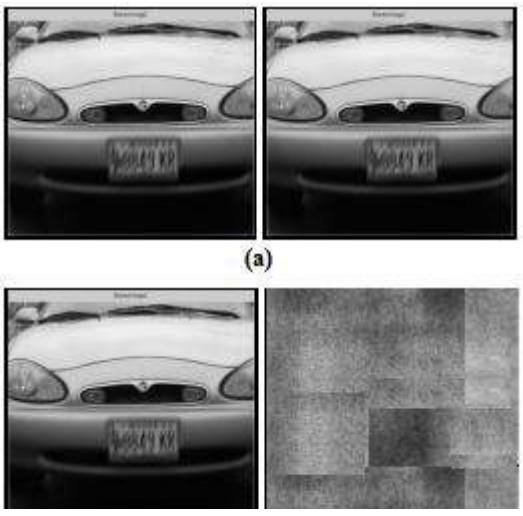

Figure 6.4: (a) Projections on licenseplate space (b) Original image and $\epsilon(x,y)$

This helps to detect the presence of license plate in the vehicle. At every location in the image, calculate the distance ϵ between the local subimage and license plate image. This distance notifies the presence of license plate. Let $\epsilon(x, y)$ represents the distance from license plate region at every point in the image. Figure 6.4(b) shows an image and its $\epsilon(x, y)$- dark area indicate the presence of a license plate (as license plate localizes in the pit).

Also, direct application of Euclidean distance is expensive, hence computing $\epsilon(x, y)$ can be done efficiently as follows:

We need to project the subimage to compute the $\epsilon(x, y)$ at every pixel of an image I and subtract the projection from original. Let training set of license plate images be $\zeta_1, \zeta_2, \ldots \zeta_m$.

The average license plate of the set is defined by

$$\kappa = \frac{1}{m} \sum_{n=1}^{m} \zeta_n$$

So, each license plate differs from average by vector:

$$\Omega_i = \zeta_i - \kappa$$

Hence to project a subimage ζ onto license plate space:

$$\Omega = \zeta - \kappa$$

Using

$$\begin{aligned}
\epsilon_k &= ||\Omega - \Omega_k|| \\
{\epsilon_k}^2 &= ||\Omega - \Omega_k||^2 \\
&= (\Omega - \Omega_k)^T (\Omega - \Omega_k) \\
&= \Omega^T \Omega - \Omega^T \Omega_k + \Omega_k^T (\Omega - \Omega_k) \\
&= \Omega^T \Omega - \Omega_k^T \Omega_k \\
&\because \Omega_k \perp (\Omega - \Omega_k)
\end{aligned}$$

A new license image (ζ) can be transformed into its eigenlicenseplate by: $\omega_k = u_k^T(\Omega), u_k$ are set of orthonormal vectors

$$\Omega_k = \sum_{i=1}^{L} \omega_i u_i$$

Eigenlicenseplates are orthonormal vectors

$$\Omega_k^T \Omega_k = \sum_{i=1}^{L} \omega_i^2$$

and

$$\epsilon^2(x,y) = \Omega^T(x,y)\Omega(x,y) - \sum_{i=1}^{L} \omega_i^2(x,y) \tag{6.6}$$

Here $\epsilon(x,y)$ and $\omega_i(x,y)$ are scalar functions of image location and $\Omega(x,y)$ is a vector function of image location.

Calculating first term of equation 6.6,

$$\begin{aligned}\Omega^T(x,y)\Omega(x,y) &= [\zeta(x,y) - \kappa]^T[\zeta(x,y) - \kappa] \\ &= \zeta^T(x,y)\zeta(x,y) - 2\kappa^T\zeta(x,y) + \kappa^T\kappa \\ &= \zeta^T(x,y)\zeta(x,y) - 2\zeta(x,y) \otimes \kappa + \kappa^T\kappa\end{aligned}$$

Calculating second term of equation 6.6,

$$\begin{aligned}\sum_{i=1}^{L} \omega_i^2(x,y) &= \sum_{i=1}^{L} \Omega^T(x,y)u_i \\ &= \sum_{i=1}^{L} [\zeta(x,y) - \kappa]^T u_i \\ &= \sum_{i=1}^{L} [\zeta^T(x,y)u_i - \kappa^T u_i] \\ &= \sum_{i=1}^{L} [I(x,y) \otimes u_i - \kappa^T u_i]\end{aligned}$$

Therefore, from equation 6.6:

$$\epsilon^2(x,y) = \zeta^T(x,y)\zeta(x,y) - 2\zeta(x,y) \otimes \kappa + \kappa^T\kappa + \sum_{i=1}^{L}[I(x,y) \otimes u_i - \kappa^T u_i]$$

$$= \epsilon^2(x,y) = \zeta^T(x,y)\zeta(x,y) - 2\zeta(x,y) \otimes \kappa + \kappa^T\kappa + \sum_{i=1}^{L}[\zeta(x,y) \otimes u_i - \kappa \otimes u_i]$$

Because κ and u_i are fixed, $\kappa^T\kappa$ and $\kappa \otimes u_i$ can be computed ahead of time using neural networks.

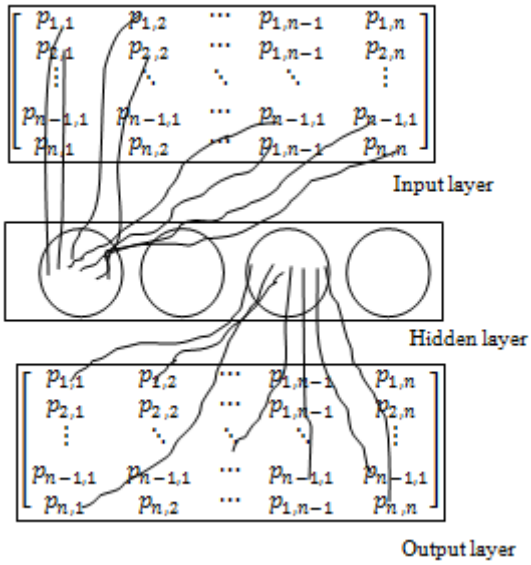

Figure 6.5: ANN for eigenlicenseplate computation

The eigenlicenseplate can be implemented using artificial neural networks also. Figure 6.5 shows a 3 layer, fully connected linear network that implements this step. Normalized vehicle image is given to the input layer, with one element per image pixel or N elements. The weights from input to hidden layer are eigenlicenseplate and hence value of each hidden unit: $\omega_i = \Omega^T u_i$. So hidden units have weight vector $\Omega^T = [\omega_1, \omega_2, \ldots, \omega_L]$. Output layer produces the licenseplate projection of the input image when output weights also correspond to the eigenlicenseplates.

6.5 Experimental Results

We have performed several experiments to assess the feasibility of our approach with a large database of vehicle images, under light variations and different scales and locations of license plates. Over 230 images had been taken, on each of which six level Gaussian pyramid was constructed to convert 512×512 pixels to 16×16 pixels. Out of 230 images, we made 23 groups of 10 images with 2 varying orientations (locations) and 2 lighting conditions, for each image. Figure 6.6 shows for one of the license image under varying lighting and orientation. The effects of varying orientations and lighting have been studied on all 230 database images. Then we classified all the images as being one of these 10 individuals. 4 eigenlicenseplates were used in the process of classification.

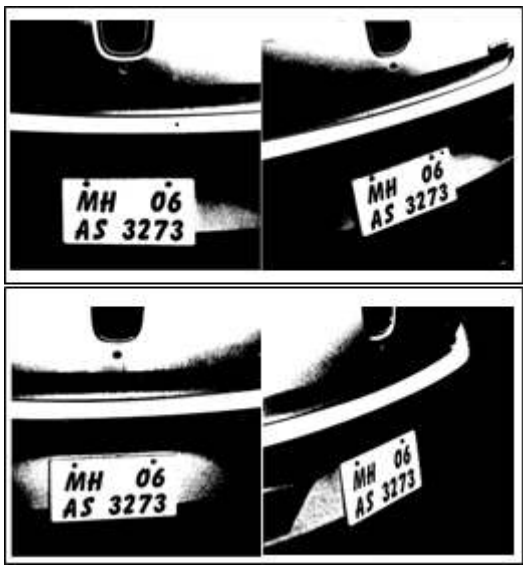

Figure 6.6: Images of one license plate, 2 under varying lighting conditions and 2 under varying orientation

Then we computed the difference between the training conditions and test conditions, the variables were referring to lighting difference and orientation difference. The results are shown in figure 6.7. It shows the number of correct classifications of vehicular license plate for different lighting and orientation conditions. When every license plate is classified as known, we achieved approximately 95% correct classification, in case of different lighting conditions and 87% correct classification, in case of different orientation.

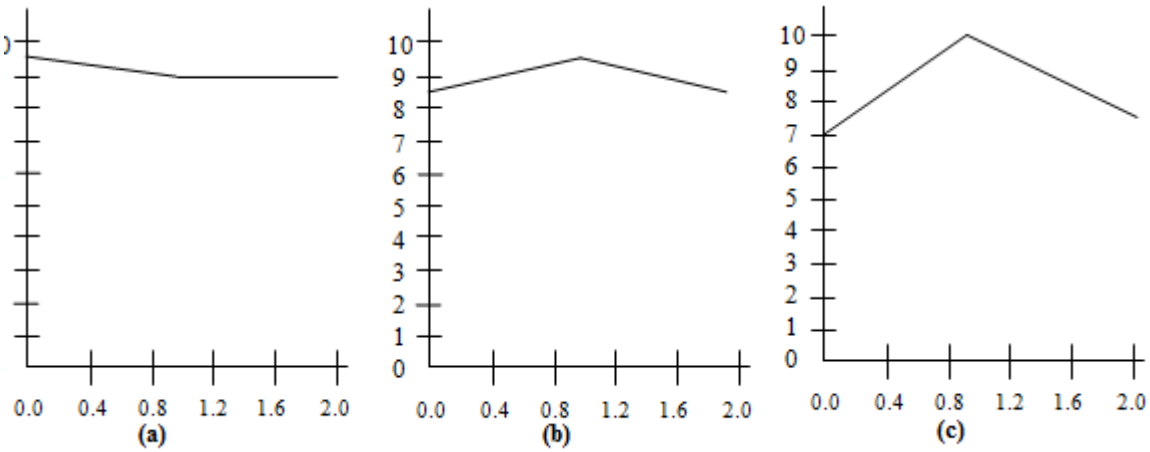

Figure 6.7: Recognition performance of eigenlicense plate with varying (a)lighting (b)orientation (c)varying lighting & orientation

We have found the following noteworthy points:

1. Light variation causes fewer errors because in this case the correlation between the neighbouring pixels is high.

2. Orientation variation causes more errors and degrades the performance substantially. The reason is that in this case the correlation between one image to another is largely lost.

6.6 Conclusion

In this chapter the importance of vehicle identification has been explored. An approach for localization of number plates is presented. In this approach, number plate located at any corner of image can be localized. Number plates having variation such as white background black script, black background white script and yellow background black script can be easily localized. The approach was based on Principal component analysis and contains mainly three stages: Spatio-Temporal filtering, Thresholding and Motion analysis. We have also shown the implementation using artificial neural networks.