ECE8001 : Cognitive Computer Vision

Professor Dr.Tony Han

Final Project: 3-D deformable-model-based Localization&Recognition of road vehicles with License Plate Detection and character recognition

Ning, Guanghan Huang, Chen

1. Introduction

License Plate Detection and Recognition is a traditional computer vision task. In our project, it is achieved with fairly good result and some novelties. In order to make the project more interesting, we made an attempt to get further information of the vehicle besides its plate number: its model type.

The paper[1] from IEEE gives an easy but effective way to judge what kind of vehicle it is given a static picture or a video clip. It is very interesting that the method the paper proposed renders a way to model vehicles, just by setting 15 parameters, which is very easy to accomplish and effective at the same time. Instead of having thousands of vehicle prototypes, the 3-D model adjusts itself (changing parameters) to fit with the vehicle in the picture and converges finally in an optimization framework with evaluation of the fitness good enough. Then this best fitting model will be compared to different vehicle type models; the one with which it gets the minimum square error is the vehicle type it belongs to.

The optimization framework is to calculate the fitness of every individual in a generation. Each generation there will be elites selected and they are used to calculate the density function. And the density function will be used to generate new generation of models. By doing this, the models tend to be better and better with new generations springing up. (If the first generation and its elites are too bad, then this will be false) So the selection of these two parameters is important: The number of individuals in a generation, and the number of elites it selects in a generation.

2. Target Problem:

The first target problem is to detect from a static image(the speed of detection is improved so that it is also fast enough for video clips) the license plate of a vehicle. Once the plate is detected, alignment of the plate is made to ensure the following segmentation and character recognition render good results. Ultimately, the license plate number of the vehicle is derived from the image.

The second target problem is to give a decision on what type a vehicle is. Traditionally, there are features to evaluate. A person who sees a sedan, SUV or a hatchback will consider their features and calculate in their head how much it seems like a sedan or how similar it is to a hatchback. Given a computer, the easy way to do is to have several typical vehicles to be the authority, and the distances between these typical vehicles are different. If a vehicle has the least distance with the typical SUV, then it is reasonable to regard it as a SUV.

The next consideration is how to get the vehicle's distance between authorities in a static or a video clip of surveillance? One way is to build a 3-D model which has lengths of every line. The 3-D model will represent the vehicle.

Then how to get an appropriate 3-D model is important. Using the EMNA algorithm as the optimization framework, the best fitting 3-D model will be found, and this model will represent the vehicle to go and compare with the authorities.

3. Implementation of License Plate Detection and Character Recognition:

3.1 License Plate Extraction

Preparation: Extract HOG features from the training images of license plates. Use SVM to train the model. We trained 3378 license plates.

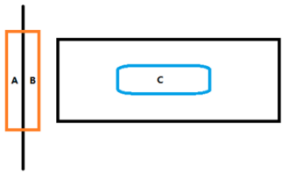
Apply edge detection on the input image first. Scan the whole image with a scanning window, and the regions with edges above a certain ratio will perform SVM classification in the window to decide if this window's score is above the threshold and therefore positive, meaning it contains a plate. So the scanning is not over the whole image, which saves time for detection. The HOG features for the whole image are computed beforehand, instead of calculating that with each scan, so it saves time too. The image will be resized to 909*523, and the scanning window is multiple scales by setting the resized image multiple scales. So for each scale the HOG features will be re-calculated. The nearby positive windows will merge to get the final location using k-means.

(**Fig1**. Edge image(left) and Scanned region(right))

3.2 Alignment

(Fig2. The extracted plate before and after Alignment**)**

(Fig3. The ROI before and after alignment)



(**Fig4.** The model for alignment)

As figure4 shows, for a license plate, a dynamic line will be located as the aligned edge, where the fitness score is maximized. The fitness score comes from:

score_left= dist_RARC- weight_left*dist_RCRB

score_right= dist_RARC- weight_right*dist_RCRB

score_up= dist_RARC- weight_up*dist_RCRB

score_down= dist_RARC- weight_down*dist_RCRB + dist_RARB;

dist_RARC is the HSV histogram distance between region A and Region C. Other notations will be likewise.

The idea is to make use of the color information to align. The region A should be very different from B and C, and region B should be very similar to region C, in terms of color.

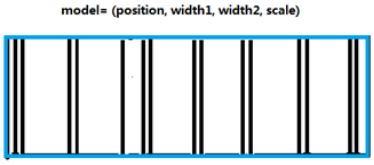
3.3 Segmentation

Based on accurate alignment, we can use k-means to derive two clusters from the license plate. One is the characters, and one is the background of the plate. Set the major cluster with more points white and the minor cluster black so the characters will be black. Segmentation is based on the binary license plate.

(**Fig5**. Aligned binary license plate)

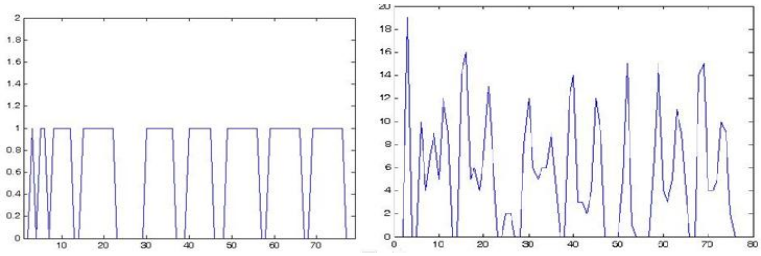
(**Fig6**. Segmented license plate)

The segmentation is done with a model. The model is shown in Fig7. It is from prior knowledge that we know the number of characters from Chinese license plates is seven, and that there is a dot between the second and the third character.



(**Fig7**. Segmentation model)

The model is deformable with its position and width and scale. Scan vertically at each x axis of the plate, the 255 to 0 inversion contributes to the score. In fig8, the y axis shows the score at the corresponding x axis of the plate.



(**Fig8**. the analysis of the best fit model. left figure is with threshold; right figure is without threshold.)

3.4 Character Recognition

Character recognition is a 1 vs N classification problem. We use SVM to classify multiple times until classification result is positive. The extracted features we use for character recognition are:

0 to 1 changes vertically and horizontally, black/white ratio horizontally and vertically, and the raw feature.

There are 5 set of features in all. They are concatenated and the overall feature length is: 96.

3.5 Dataset and Results:

The plate detection is tested on two datasets, one is daylight and contains 43 images, the other one is night and contains 62 images.

The detection rate is both 100% and there is no false positives. All of them are well aligned.

For night dataset, the speed for detection is around 200ms on average with 3 scanning scales and the stepwidth being 8. For the daylight dataset, the plates are fairly small, in order to ensure 100% detection rate, the scales are 3, but the stepwidth to scan is set down to 2. And the average detection time is around 1000ms. If we tolerate some miss, the time could be around 200ms too.

The Segmentation is 100% correct.

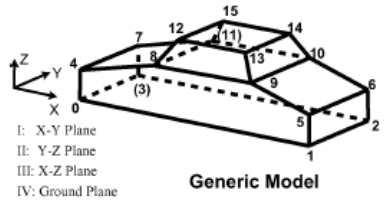
We Trained 400 characters, and tested 100 characters. The testing characters are derived from the segmentation output patches. The character recognition rate for segmented single characters is 99%. (The demo is attached in the e-mail)

4. Implementation of Recognition and classification of Vehicle based on 3-D deformable model:

4.1 Build the Model:

The reference paper introduced its way of having 12 shape parameters to exact the shape of the 3-D deformable model and 3 pose parameters to localize the model. So in the 15 dimensional space the model will be able to fit on to the image.

The 12 shape parameters are defined like this.



(**Fig9**. The generic model)

Parameters	Descriptions	Value Range
$W1$	Distance from 1 to 2	$[1500, 2100]$
$H1$	Distance from 1 to 5	$[400, 800]$
$H2$	Distance from 0 to 4	$[400, 800]$
L	Distance from 0 to 1	$[3200, 4000]$
$H3$	Distance from 8 to I	$[\max(H1, H2), 900]$
$X1$	Distance from 8 to II	$[0, L/2]$
$X2$	Distance from 9 to II	$[\max(X1, L/2), L - 200]$
$X3$	Distance from 12 to II	$[X1, \min(X2 - 1300, L/2)]$
$X4$	Distance from 13 to II	$[X3 + 1000, X2 - 200]$
$W2$	Distance from 13 to 14	$[1000, W1]$
$H4$	Distance from 13 to I	$[1.5 \times H3, 1400]$
Δ	Distance from I to IV	$[150, 250]$

(**Fig10**. 12 Shape parameters)

Except from the shape parameters, pose parameters are harder to get because for static images, it is harder to get the location X,Y of the vehicle in the image without knowing any changing pixels. The paper used a bounding box to get the X and Y and used HOG to get the third parameter seita. We implemented the HOG but after all the initial X and Y are achieved by hand. The emphasis of this paper is not the algorithm to get a bounding box from a static image, so we skip this process and directly get the pose parameters given a particular image. Actually the detection of the vehicle can also be achieved by HOG with SVM as explained in License Plate Detection; the only difference is the training images will be the vehicle itself rather than its license plates.

4.2 Projection to 2-D image

The 3-D model will be projected into 2-D space so that it could match the vehicle in the image.

M is the projection matrix from MCS (model coordinate system) to WCS (world coordinate system). It includes intrinsic and extrinsic parameters and is decided by the camera calibration. The surveillance image we get from the paper's reference website has a unique camera calibration, and from the website of the author of this paper, we got the M for this picture's camera shown below.

```
M= [0.77900 -0.18953 0.59769 0.00000;
      -0.62703 -0.23676 0.74215 0.00000;
      0.00084 -0.95290 -0.30327 0.00000;
    -3479.05000 -849.37500 33783.70000 1.00000];
```

We also got the focal length, which correlates to the scale.

```
focal_length= 1360;
```

T is the transform matrix.

```
 T = [ cos(seita) sin(seita)  0 0;
-sin(seita) cos(seita) 0 0;
      0            0     1 0;
      X            Y     0 1];
```

The relationship between the coordinate (x, y, z) of 3-D model and the coordinate (x', y') of 2-D model is like this:

```
(x1,y1,c1,q1)= [x,y,z,1]*T*M;
 Y'= (focal_length*y1)/c1+Y;
 X'= (focal_length*x1)/c1+X;
```

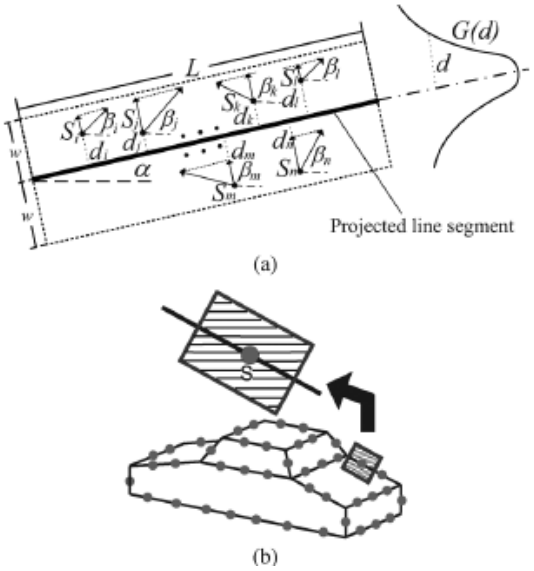
4.3 Evaluation of fitness

The way to calculate the fitness of a model to the vehicle in the image is the most important part of this project. A good fitness function would render smooth surface and be easier to get the global minimum.

Since the wire-frame 3-D model already projected onto the image plane, it forms a series of visible projected line segments. For every visible projected line segment whose direction is assigned as alpha with length of L in the image plane, we have 2alpha*L virtual rectangle. (Fig11.(a))

For pixel Si within the rectangle, calculate its gradient magnitude m(x,y) and orientation beta(x,y) from pixel differences as follows:

$$\begin{cases} A(x,y) = I(x+1,y) - I(x-1,y) \\ B(x,y) = I(x,y+1) - I(x,y-1) \\ m(x,y) = \sqrt{A(x,y)^2 + B(x,y)^2} \\ \beta(x,y) = \tan^{-1}\left(B(x,y)/A(x,y)\right). \end{cases}$$



(**Fig11**. The fitness evaluation principle) (a). Regions around a visible projected line. (b).Regions around sampled points.

Fitness score of contributed by Si is measured by the vertical component of its gradient magnitude along the line.

$$E_{S_i} = m(x,y) \cdot \sin(\beta(x,y) - \alpha)$$

For those pixels that are closer to the line within the rectangle would have higher weight and contribute more to the FES. Make di the distance between a pixel to the line. Then FES contributed by the line will be:

$$E_l = \sum_{S_i} [E_{S_i} \cdot G_{0,w}(d_i)]$$

Then we have all the lines contribution add together, resulting in the final FES.

$$E = \sum_{l} [\log(E_l)].$$

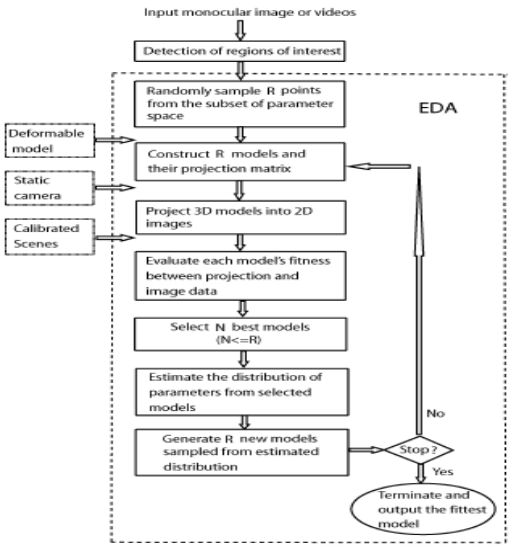
In order to decrease the amount of calculation, the paper suggests sampling some numbers of points from each line based on the length of it, and only calculate the FES of the rectangle surrounding these sampled points.

If length of a line is L, and the longest of all lines is Ll and the shortest line's length is Ls. Then the number of sampling points should be:

$$n = 6 \times ((L - L_s)/(L_l - L_s)) + 1.$$

4.4 Optimization framework

Being able to calculate the FES of one model, the last step would be to generate many models and find the best which is the most fitting to the vehicle in the image.



(**Fig.12** the flowchart of the algorithm)

Then the optimization framework will be responsible for this task.

Selecting elites from each generation and depend on them to calculate density function, with which new generations can be generated. Thus, the offspring will be more and more like the vehicle in the image and finally it will hopefully converge and reach the global minimum.

This optimization method is called EDA(Estimation of Distribution). And the particular kind of EDA we are using is EMNA(Estimation of Multivariate Normal Algorithm- global). It is a good evolutionary computing framework for this kind of problem because the model is multi-dimensional.

Algorithm 2 $\text{EMNA}_{\text{global}}$ for deformable models

1: BEGIN

2: $D_0 \longleftarrow$ Sample R individuals randomly from the 15-D parameter space; $g \longleftarrow 1$

3: **while** the difference between two generations of average fitness is less than a threshold **do**

4: Calculate the evaluation score of each individual using FES and sort them.

5: $D_{g-1}^N \longleftarrow$ Select $N \leq R$ individuals which have higher scores from D_{g-1}

6: $f_g(x) = f(x|D_{g-1}^N) = \mathcal{N}(x; \mu_g, \Sigma_g) \longleftarrow$ Estimate the multivariate normal density function using D_{g-1}^N

7: $D_g \longleftarrow$ Sample R individuals (the new populations) from $f_g(x)$

8: $g \longleftarrow g + 1$

9: **end while**

10: END

(**Fig.13** EMNA as optimization framework)

The means and the variance-covariance are calculated as below:

$$\begin{cases} \hat{\mu}_{i,g} = \frac{1}{N}\sum_{r=1}^{N} x_{i,r}^{g}, \\ \hat{\sigma}_{i,g}^{2} = \frac{1}{N}\sum_{r=1}^{N}\left(x_{i,r}^{g} - \hat{\mu}_{i,g}\right)^{2} \\ \hat{\sigma}_{i,j,g}^{2} = \frac{1}{N}\sum_{r=1}^{N}\left(x_{i,r}^{g} - \hat{\mu}_{i,g}\right)\left(x_{j,r}^{g} - \hat{\mu}_{j,g}\right), \end{cases}$$

4.5 Final Decision

With the 15 parameters of the best model known, calculation of the distance between the optimal model and different typical models will be done. Using MES method, the result is easily figured out.

4.6 "Pseudo Code" arrangement

(1). **HOG.m**

Compute best two directions o1 and o2, getting seita.

(2). **Pose.m**

Compute bounding box and its X,Y **(now get them by hand)**

Compute seita using **HOG** which provide o1 and o2.

Compute projection matrix with X, Y, seita and therefore compute model dots in MCS&WCS coordinate systems

(3). **Model.m**

Using generated 12 parameters, it forms a 3-D vehicle model within range

(4).**draw_model.m**

Using **model**, compute the coordinate axis's of every dot of the vehicle model within MCS.

And draw the 3-D model so it is visualized.

(5).**Fitness.m**

Calculate FES which will be used in **EMNA.**

(6). **EMNA.m**

Generate many generations of vehicle models using **model**, and use **fitness** to get FES of each individual of each generation, find the best individual with optimum 12 parameters that fit the vehicle in bounding box best.

(7).**Judge.m**

using **sedan, suv**, and **hatchback**, by comparison, find which type the vehicle is.

(8). **Sedan.m** (9).**suv.m** (10). **Hatchback.m**

Sedan model. With vehicle model as input, calculate the difference between sedan and itself.

Suv model. With vehicle model as input, calculate the difference between suv and itself.

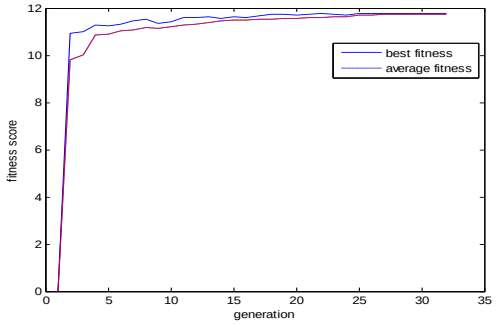
hatchback model. With vehicle model as input, calculate the difference between hatchback and itself.

4.7 Dataset and Results:

The dataset we use is PETS2000. Instead of the video, we choose 10 of the static test images. We are currently setting initial X, Y and seita by hand, so it will be tedious if we test all the 1451 images.

(Fig.14 Test on a sedan**,** Corresponding 3D model of(from left to right): First Generation, Generation 10, Converging Generation. The classification output type is sedan.)

(Fig.15 Test on a SUV**,** Corresponding 3D model of(from left to right): First Generation, Second Generation, Generation 10, Converging Generation. The classification output type is SUV.)



(**Fig16**. Evolutionary curve for red sedan. For this particular trial, it converged at generation 25. Average fitness is the average fitness of all the individuals of the corresponding generation, which is indicated by the x axis. Best fitness is the best fitness of all the individuals of a corresponding generation.)

Since the model is deformable, there are two things we know:

(1). For different images and different vehicles, the model will adapt.

(2). Every time it converges, the model would be different. Therefore the result will be different.

Since the model will adapt, the reference paper only tested three images of three different vehicle types. They further tested different poses.

Since every time the model may be slightly different when it converges, the classification rate in terms of trials should also be evaluated. (# of positive classification/# of trials)The corresponding classification rate is: 70%. Hatchbacks and SUVs can sometimes be mis-classified.

The implementation still need to be improved in that the in-visible lines should not contribute to the score. We need to let the program know the invisible lines automatically for every pose.

5. Reference:

[1] *Three-Dimensional Deformable-Model-Based Localization and Recognition of Road Vehicles, from Zhaoxiang Zhang, Tieniu Tan, Kaiqi Huang, Yunhong Wang*