

# Self Lane Assignment Using Egocentric Smart Mobile Camera For Intelligent GPS Navigation

Tianshi Gao, Hamid Aghajan  
Department of Electrical Engineering  
Stanford University, CA 94305, USA  
{tianshig, aghajan}@stanford.edu

## Abstract

*In this paper, we study the self lane assignment problem, i.e. given an image taken inside a vehicle, infer on which lane the image is taken. This problem serves as an example of active egocentric vision application with data fusion. In this application, a camera is mounted inside the vehicle looking outside to the world. Combined with a GPS with a digital map this smart mobile camera is capable of reasoning on which lane the vehicle is. This inference result is then fed back to the GPS to provide the driver with more intelligent navigation instructions. We form the self lane assignment inference problem as a scene classification problem which requires classifying scenes in finer categories than the traditional case. We design the features to represent the image in a holistic way bypassing individual object detection, develop an automatic horizon detection algorithm, and employ and compare three learning algorithms for decision making on the lane number. The experiment results show that our method can achieve the precision and recall rates around or above 90% at the same time.*

## 1. Introduction

Most vision-based applications are developed using a single modality and independent processing of visual data. However, in many practical applications, a vision module alone may not be able to solve the problem effectively or robustly. Fusion of visual data with other sensing mechanisms and prior information can offer a more effective solution in a variety of applications.

In this paper, we study an application developed based on the fusion of visual information with a vehicular navigation system. In this setup, a camera is mounted inside the vehicle and looks outside. Equipped with a smart camera with the capability of not only sensing the world but also reasoning about the world, the vehicle in our application becomes an active embodiment of an egovision concept. From an ego-

centric view, we use this mobile smart camera together with the GPS to serve the driver with more intelligent navigation instructions. To give an example, imagine that you are driving on the highway at 70 mph and trying to figure out which lane you should follow to exit at some distance in front. The precision of the current GPS map is not able to tell which lane you are on, so the instruction given by the GPS is just as simple as “take the exit right”, possibly resulting in the driver’s panic as to how quickly he has to change lanes to exit. However, if we have a smart camera mounted inside the vehicle which is capable of inferring the current lane the vehicle is on and feeding this information into the GPS, then based on this inferred current position and the target position more intelligent instructions like “stay on the current lane” or “change to your next right lane” can be offered to the driver.

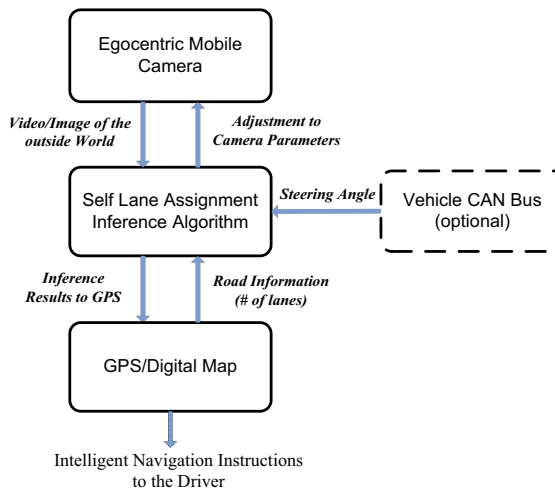


Figure 1. System Diagram

Figure 1 shows the diagram of our system. The self lane assignment inference algorithm plays a central role as the convergence point of data fusion. The videos/images of the

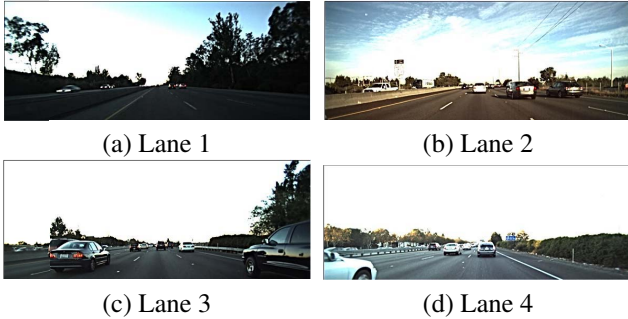


Figure 2. Sample images from four different lanes (classes)

outside world captured by the camera and the road information (number of lanes) provided by the GPS with a digital map are fed into the self lane assignment inference algorithm, and then the inference results are fed back to the GPS to provide the user with more intelligent navigation instructions. In addition, there are two possible extensions. One is that the inference algorithm can instruct the camera to adjust certain parameters based on the image processing results and the vehicle CAN bus can also be synchronized with the camera to provide the steering angle to the inference algorithm to improve the reasoning accuracy. In the current work, we focus on combining the images taken by the camera and the GPS information to infer on which lane the vehicle is.

We form the self lane assignment inference problem as a scene classification problem and discuss feature design and representation, horizon detection and learning algorithms in section 2. In section 3, we present experiment results and analyze the effectiveness of our methods. Finally, section 4 concludes the paper.

## 2. Approach

### 2.1. Problem Formation

In this work, we focus on the highway situation where lanes exhibit low curvature and the number of lanes is assumed as a prior information. As shown in Figure 1, this prior information comes from the digital map used by the navigation system.

Figure 2 shows four sample images from different lanes. To infer on which lane the image is taken, we can think of this problem as a scene classification problem in which given an image we want to classify it into four categories corresponding to four scenes seen on lanes 1, 2, 3, and 4, respectively (lane 1 corresponds to the leftmost lane, and others are indexed from left to right).

There are two potential paradigms for solving the scene classification problem. The first one is based on individual object detection after which the inference is done in a higher layer. However, the detection of lane markers and vehicles are not reliable in such a dynamic environment seen by a

moving camera. Due to the occlusion, perspective projection and constantly changing environment, the appearance of the lane markers and vehicles varies in a large range and the lane markers on the side lanes are sometimes even unobservable. Therefore, some previous work on lane detection only focused on the detection of a single lane in front of the vehicle or detection of multiple lanes without occlusions from other vehicles [1, 2]. For vehicle detection, most existing methods (a survey at [3]) only focus on mid-range vehicle detection. Moreover, multiple tasks of detecting different individual objects can cost more processing time and accumulation of detection errors which will adversely affect the inference in the later stage.

To avoid these problems, the second paradigm is to describe the scene in a holistic way. For example, [4, 5, 6, 7] successfully developed holistic representation of an image either in a supervised or unsupervised fashion to classify images into very broad categories such as office, highway, forest, mountain, and so on. However, methods in [4, 5, 6] do not keep the spatial information of the features which is critical in our case as discussed in the next section. Although the method in [7] keeps the spatial information, if the same clustering based method is applied to our problem, it is very likely that the lane marker information cannot be extracted as an independent cluster due to the sparsity of this feature. Moreover, all these methods are developed to classify classes with relatively large differences while our problem is more like to distinguish sub-classes within a single class which requires understanding the scenes in a finer granularity. So the existing methods cannot be used to effectively solve our problem. In the following sections, we discuss our feature design and representation based on low-level features in a global way to bypass explicit individual object detection while achieving the goal of classifying images into finer categories.

### 2.2. Features

#### 2.2.1 Feature Design

Based on how the humans distinguish different lane classes, there are three desired types of information that a good feature design needs to capture:

- *Lane markers.* Lanes are separated and defined by lane markers, so they are the most discriminative information.
- *Vehicles.* If there are multiple vehicles on your right, then it is less likely that you are on the rightmost lane, and similarly for the left lane.
- *Spatial distribution.* In addition to the presence of lane markers and vehicles, the more important information is how they are spatially distributed on the image plane.

To capture the information above, as shown in Figure 3(a), a filter bank consisting of oriented steerable filters with even and odd phases [8] is used to implicitly capture the textures for both lane markers and vehicles. Due to the perspective projection, the scale of edges at different positions on the image plane varies, so we used two sets of 12 filters which are at two different scales. To keep the spatial information, the image is partitioned into multiple cells below the horizon as shown in Figure 3(b). The upper part above the horizon is discarded as this area is usually the sky and gives no discriminative information about different classes.

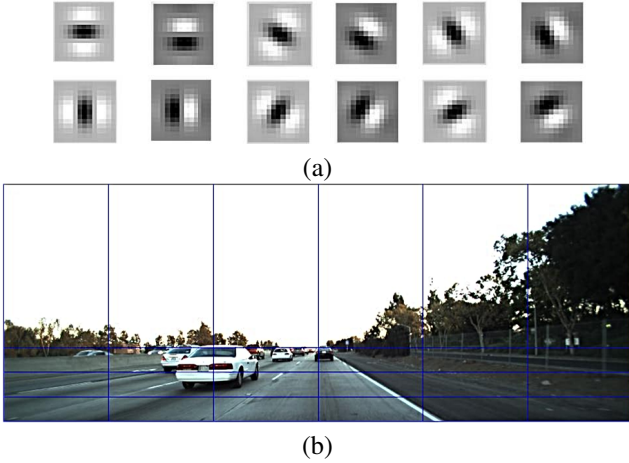


Figure 3. (a) A filter bank consisting of 12 oriented steerable filters per 30 degrees with even and odd phases (b) The spatial partition below the horizon.

### 2.2.2 Feature Representation

Denote the filters as  $F_{k,m}$  where  $k = 1, 2$  corresponds to two sets of filters with different scales and  $m = 1, 2, \dots, 12$  corresponds to each filter turned to different angles with even or odd phases. In addition, the image is partitioned into  $M \times N$  cells, where  $M = 3$  and  $N = 6$ . Let the grayscale image be  $I$  and then the response of the image at a specific cell to each of the filters is incorporated into:

$$x_{i,j,k,m} = \sum_{(x,y) \in C_{i,j}} |(I * F_{k,m})(x,y)| \quad (1)$$

where  $C_{i,j}$  corresponds to the cell at the  $i$ th row and the  $j$ th column with  $i = 1, \dots, M$  and  $j = 1, \dots, N$ . In order to make the response less sensitive to the illumination and contrast of the image, for each cell the 12-dimensional vector  $\mathbf{x}_{i,j,k,\cdot} = [x_{i,j,k,1}, x_{i,j,k,2}, \dots, x_{i,j,k,12}]^T$  is normalized to have energy 1, i.e.,  $\|\mathbf{x}_{i,j,k,\cdot}\|_2 = 1$ . But if the energy of  $\mathbf{x}_{i,j,k,\cdot}$  is too small, the vector is not normalized so that uniform road regions can be captured. Furthermore, three statistics of the responses for a set of 12 filters within

one cell are also incorporated into the feature vector. These three statistics include mean, argmax and max–median of the components of  $\mathbf{x}_{i,j,k,\cdot}$ . So each cell is associated with a  $(12 + 3) \times 2$  dimensional descriptor, and all the descriptors from  $3 \times 6$  cells are stacked into a single  $15 \times 2 \times 3 \times 6 = 540$  dimensional feature vector as a representation for each image.

### 2.3. Horizon Detection

As mentioned earlier, the image is partitioned into multiple cells below horizon. The horizon is obtained by detecting the vanishing point in two steps as follows:

1. *Lone line detection.* We used the method developed in [9] to detect lone lines in the image and filtered out those nearly vertical or horizontal lines to reduce outliers. Denote the number of lone lines after filtering as  $L$  and these lines are parameterized by  $\theta_i$  and  $r_i$  as follows:

$$x \sin \theta_i + y \cos \theta_i = r_i \quad i = 1, 2, \dots, L \quad (2)$$

2. *Robust fitting in the Hough domain.* Since the vanishing point is located at the intersection of the  $L$  lines, we estimate the vanishing point by minimizing the norm of the residual:

$$\text{minimize } \|A\mathbf{x} - \mathbf{r}\|_1 \quad (3)$$

where,

$$A = \begin{pmatrix} \sin \theta_1 & \cos \theta_1 \\ \sin \theta_2 & \cos \theta_2 \\ \vdots & \vdots \\ \sin \theta_L & \cos \theta_L \end{pmatrix} \quad \mathbf{r} = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_L \end{pmatrix}$$

We used the 1-norm instead of the ordinary least square to make the estimation less sensitive to outliers. This convex optimization problem 3 is solved using CVX [11]. Figure 4 shows some detection results.

### 2.4. Learning Algorithm

Different dimensions in the feature vector do not provide equal information. For example, the features from the cell that is in front of the vehicle contain little information (which are almost the same for all classes). To have better generalization error, it is necessary to reduce the model complexity to avoid overfitting, so three different learning algorithms are chosen as follows:

1. *Adaboost with decision trees.* We used the logistic regression version of Adaboost [12, 10] with weak learners based on decision trees. Decision trees make good

**Input:**

- Training dataset:  $D = \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \dots, (\mathbf{x}^{(m)}, y^{(m)})\}$ , where  $y^{(i)} \in \{-1, +1\}$
- Initial weights:  $\mathbf{w}^1 = \{w_1^1, w_2^1, \dots, w_m^1\}$
- Number of nodes per decision tree:  $N_n$
- Number of weak learner decision trees:  $N_T$

**Procedure:**

For  $t = 1, \dots, N_T$ :

- Fit a  $N_n$ -node decision tree  $T_t$  to the weighted dataset with weight distribution  $\mathbf{w}^t$
- Assign to each terminal node  $T_{t,k}$  a weight:  $f_{t,k} = \frac{1}{2} \log \frac{\sum_{i:y^{(i)}=1, (\mathbf{x}^{(i)}, y^{(i)}) \in T_{t,k}} w_i^t}{\sum_{i:y^{(i)}=-1, (\mathbf{x}^{(i)}, y^{(i)}) \in T_{t,k}} w_i^t}$
- Update weights:  $w_i^{t+1} = \frac{1}{1 + \exp(y^{(i)} \sum_{t'=1}^t f_{t',k_{t'}})}$  with  $k_{t'} : (\mathbf{x}^{(i)}, y^{(i)}) \in T_{t',k_{t'}}$

**Output:**

- A series of decision trees:  $T_1, T_2, \dots, T_{N_T}$
- Weighted log-ratio for each node of each tree:  $f_{1,1}, \dots, f_{N_T, N_n}$

Figure 5. Boosted decision tree using logistic regression version of Adaboost

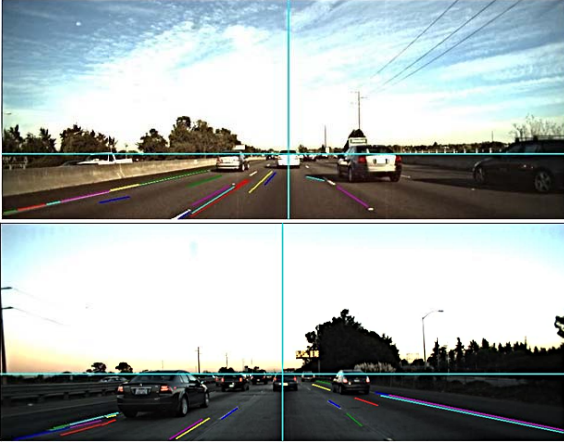


Figure 4. Sample horizon detection results. Color lines are the detected long lines and the cross is the detected vanishing point.

weak learners, since they provide explicit feature selection and limited modeling of the joint statistics of features, and boosting gives better results than a single tree in general. The training procedure of boosted decision tree using logistic regression version of Adaboost is described in Figure 5.

2. *Bayesian logistic regression.* By assuming a prior on the coefficients in logistic regression, Bayesian logistic regression is capable of shrinking the coefficients to avoid overfitting [13]. To be more specific, the poste-

rior is:

$$p(\boldsymbol{\theta}|D) \propto \left( \prod_{i=1}^m \frac{1}{1 + \exp(-\boldsymbol{\theta}^T \mathbf{x}^{(i)} y^{(i)})} \right) p(\boldsymbol{\theta}) \quad (4)$$

where  $D$  is the training dataset  $\{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \dots, (\mathbf{x}^{(m)}, y^{(m)})\}$ , and the prior of  $\boldsymbol{\theta}$  is a multivariate Gaussian with mean 0 and covariance  $\sigma^2 I$ , i.e.  $\boldsymbol{\theta} \sim \mathcal{N}(0, \sigma^2 I)$ . Then the log posterior over  $\boldsymbol{\theta}$  (ignoring the normalization constant) is :

$$l(\boldsymbol{\theta}) = - \sum_{i=1}^m \log(1 + \exp(-\boldsymbol{\theta}^T \mathbf{x}^{(i)} y^{(i)})) - \sum_{j=1}^n \left( \log \sigma + \frac{\log 2\pi}{2} + \frac{\theta_j^2}{2\sigma^2} \right) \quad (5)$$

where  $n$  is the dimension of the feature vector  $\mathbf{x}$ . Finally, the single point estimation of  $\boldsymbol{\theta}$  is:

$$\boldsymbol{\theta}^* = \operatorname{argmax} l(\boldsymbol{\theta}) \quad (6)$$

Since the negated log-posterior is convex, we can use gradient descent to solve the MAP problem 6. We used the implementation from [14].

3. *SVM.* Although SVM doesn't provide explicit feature selection or shrinkage, it is still possible to have relatively low error rate. We used the implementation from [15] with linear kernel.

### 3. Experiment and Result

#### 3.1. Data

To prepare the data, two segments on the highway are selected. An IP camera is mounted at the position of the rear mirror inside the vehicle facing the front towards outside. The camera is synchronized with the GPS data through the CAN bus in the vehicle [16]. Eight short video sequences have been taken repeatedly on these two segments at different daytime hours. We randomly sampled from these sequences to get images from different lanes. The numbers of images collected for lane 1 to lane 4 are 112, 500, 750 and 750 respectively. The images are sampled sparsely to reduce the correlation between consecutive frames. Since the dataset is unbalanced, in the later training phase, we set the initial weights for Adaboost according to the proportion of the number of images for each class, and also adjust the penalty parameters of relaxation for different classes in SVM according to the ratio between different classes.

#### 3.2. Horizon Detection

We estimated the horizon positions for all the 2112 images, and the results are summarized in Table 1 and Figure 6. Around 98% of the detected horizon lies in the approximate 5% relative error band. The mean of the relative error is only 1.33% which is 3 pixels in our case.

mean of the relative error	std of the relative error
1.33%	3.85%

Table 1. Horizon Detection Error Rate

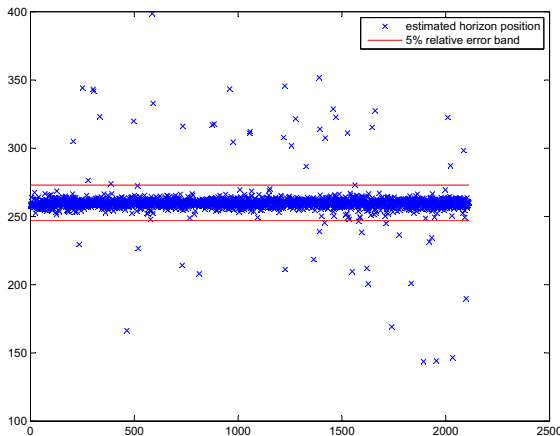


Figure 6. Estimated horizon positions. The x-axis is the index of each image, and the y-axis is the y coordinate of the estimated horizon.

	Lane 1	Lane 2	Lane 3	Lane 4	Recall
Lane1	109	3			97.32%
Lane2	2	435	53	10	87.00%
Lane3		36	688	26	91.73%
Lane4		8	43	699	93.20%
Precision	98.20%	90.25%	87.76%	95.10%	

Table 2. Confusion Table for Adaboost with Decision Trees

	Lane 1	Lane 2	Lane 3	Lane 4	Recall
Lane 1	110	2			98.21%
Lane 2	3	464	30	3	92.80%
Lane 3		23	689	38	91.87%
Lane 4		3	40	707	94.27%
Precision	97.35%	94.31%	90.78%	94.52%	

Table 3. Confusion Table for Bayesian Logistic Regression

	Lane 1	Lane 2	Lane 3	Lane 4	Recall
Lane 1	110	2			98.21%
Lane 2	2	453	37	8	90.60%
Lane 3		46	670	33	89.33%
Lane 4		6	44	700	93.33%
Precision	98.21%	89.35%	89.21%	94.47%	

Table 4. Confusion Table for SVM

#### 3.3. Classification

The classifier for each class is trained in one vs. all fashion. At the test stage, the classification result is chosen as the one with the highest probability. We used 3-fold cross-validation to both choose parameters associated with the algorithm and evaluate the performance of the three different methods. The fold number is 3 instead of the more common 5 or 10 because we want to reduce the correlation between the training and test data.

The confusion table for each algorithm is shown in Tables 2, 3 and 4. All three methods give comparable results, and Bayesian logistic regression has slightly better accuracy for lane 2 and lane 3. In general, the precision and recall rate for each class is around or above 90%. In addition, the results are consistent with the intuitions in terms of: i) lane 1 and lane 4 have higher accuracy than lane 2 and lane 3, since lane 1 and lane 4 are less likely to be confused by other classes; ii) lane 2 and lane 3 are more likely to be confused by each other than to be confused by lane 1 or lane 4; iii) lane 1 is more likely to be confused by lane 2 than lanes 3 or 4, and lane 4 is more likely to be confused by lane 3 than lanes 1 or 2.

Moreover, top features selected by Adaboost with decision trees are shown in Figure 7. The oriented filters at different cells correspond to the selected features. Interestingly, these top selected features have good interpretations: the positions and orientations of these filters are more or less consistent with the positions and orientations of the lane markers or the vehicles. For example, for the

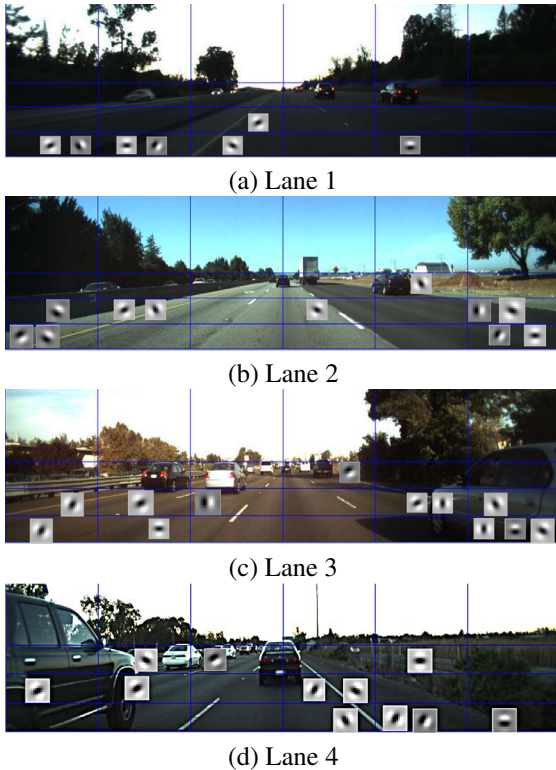


Figure 7. Selected top features by Adaboost with decision trees for different classes.

lane 2 classifier, most of the filters are at the positions of the lane boundaries and the orientations are either parallel (strongest response) or orthogonal (weakest response) to the lane markers. Another example is the vertical filter at [2nd row, 3rd column] and the horizontal filter at [3rd row, 2nd column] for the lane 3 classifier, since the vertical and horizontal edges are probably generated by vehicles.

#### 4. Conclusion and Discussion

In this paper, we studied the self lane assignment inference problem as an example of active egocentric vision application with data fusion. We proposed a novel and effective algorithm to infer the lane number from a single image based on a holistic representation from low-level features. The experiment results showed that the proposed method achieves high accuracy with precision and recall rates around or above 90%. Some further improvements could include incorporating temporal dimension using CRF (Conditional Random Fields) and making spatial partitioning more robust, perhaps by drawing rays from the vanishing point.

#### 5. Acknowledgments

The authors wish to thank Jaime Camhi from Electronics Research Laboratory of Volkswagen Group of America,

Inc. for providing the videos used in this paper.

#### References

- [1] Z. Kim. Robust lane detection and tracking in challenging scenarios. *IEEE Trans. on Intelligent Transportation Systems*, vol. 9, no. 1, pp. 16–26, 2008.
- [2] Y. Wang, E.K. Teoh, and D. Shen. Lane detection and tracking using B-Snake. *Image and Vision Computing*, pp. 269–280, 2004.
- [3] Z. Sun, G. Bebis, and R. Miller. On-road vehicle detection: a review. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 28, no. 5, 2006.
- [4] A. Oliva and A. Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *IJCV*, 42(3):145–175, 2001.
- [5] A. Torralba, K.P. Murphy, W.T. Freeman and M.A. Rubin. Context-based vision system for place and object recognition. In *Proc. ICCV*, 2003.
- [6] F.F. Li and P. Perona. A Bayesian hierarchical model for learning natural scene categories. In *Proc. CVPR*, 2005.
- [7] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proc. CVPR*, 2006.
- [8] E.P. Simoncelli and W.T. Freeman. The steerable pyramid: A flexible architecture for multi-scale derivative computation. In *Proc. ICIP*, 1995.
- [9] J. Kosecka and W. Zhang. Video compass. In *Proc. ECCV*, 2002.
- [10] D. Hoiem, A. Efros, and M. Herbert. Recovering Surface Layout from an Image. In *IJCV*, 2007.
- [11] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming (web page and software). <http://stanford.edu/boyd/cvx>, December 2008.
- [12] M. Collins, R. Schapire, and Y. Singer. Logistic regression, adaboost and bregman distances. *Machine Learning*, vol. 48, no. 1–3, 2002
- [13] A. Genkin, D.D. Lewis and D. Madigan. Large-Scale Bayesian Logistic Regression for Text Categorization. *Technometrics*, 2006.
- [14] A. Genkin, D.D. Lewis and D. Madigan. BBR: Bayesian Logistic Regression Software. <http://www.stat.rutgers.edu/madigan/BBR/>.
- [15] C. Chang and C. Lin. LIBSVM – A Library for Support Vector Machines. <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [16] N. Hu, T. Gao, J. Camhi, C. Lee, D. Rosario and H. Aghajan. Smart Node: Intelligent Traffic Interpretation. *ITS World Congress*, Nov. 2008.