An audio-visual sensor fusion approach for feature based vehicle identification

Andreas Klausner Allan Tengg Christian Leistner Stefan Erb Bernhard Rinner [*]

Graz University of Technology
Institute for Technical Informatics
Inffeldgasse 16/1, Austria, Graz, 8010
{klausner,tengg,leistner,erb,rinner}@iti.tugraz.at

Abstract

In this article we present our software framework for embedded online data fusion, called I-SENSE. We discuss the fusion model and the decision modeling approach using Support Vector Machines. Due to the system complexity and the genetic approach a data oriented model is introduced. The main focus of the article is targeted at our techniques for extracting features of acoustic- and visual-data. Experimental results of our "traffic surveillance" case study demonstrate the feasibility of our multi-level data fusion approach.

1 Introduction

Currently there is a strong trend towards integration of sensor, computing and communication technology into everydays life. The ultimate goal here is to provide as much support as possible while concealing the computing devices from the users. Our I-SENSE project [6] demonstrates the potential of combining the scientific research areas multi-sensor data fusion and pervasive embedded computing. *I-SENSE* is used as an acronym for intelligent sensing and our project is targeted at various applications. Classification of vehicles is one of the most important tasks in traffic surveillance systems and therefore, we demonstrate the feasibility of our fusion approach in a classification case study. The aim of our multi-sensor fusion framework is to create a synergistic process in which the consolidation of individual data creates a combined resource with a productive value greater than the sum of its parts.

To achieve this aim, our approach is to perform multi-level data fusion by combining data from different sensors at different levels of abstraction. The I-SENSE fusion-framework supports three basic levels of data fusion. These fusion levels are differentiated according to the amount of information they provide. The most basic level is called *raw-data fusion*. At this level, only raw, uncorrelated data is provided to the user. In comparison, level two data fusion provides a higher level of inference and delivers additional interpretive meaning suggested from the raw data and data will be fused based on extracted features. Therefore this level is called *feature-level fusion*. Level three data fusion is designed to make assessments and provide recommendations to the user and is called *decision fusion*.

The remainder of the paper is organized as follows: Section 2 gives an short review about related activities. Section 3 discusses our I-SENSE project as a framework for multi-sensor data fusion. Section 4 deals with the visual feature extraction tasks while section 5 treats the acoustic feature extraction tasks. In section 6 we discuss the decision modeling process. In section 7 we present some experimental results of our approach, and section 8 concludes the paper with a short discussion of our approach.

2 Related work

Our idea of developing a high-performance data fusion architecture originates from the SmartCam project [1]. In the I-SENSE research project we extend the SmartCam to distributed embedded sensor nodes (consisting of a network processor and various digital signal processors) capable of fusing data from various heterogeneous sensors, ranging from simple sensors such as light barriers and induction loops over audio sensors to different video sensors. There exists a large variety of multi sensor fusion applications, but our proposed approach is different in several ways to these applications.

"Project Correlation", funded by the U.S. Air Force, was the first approach to step back from the many application-specific and system-specific solutions and developed a set of generic/reusable engineering guidelines for an effective data fusion-problem solution. A methodology for fusion software development, based on the C4ISR architecture [4] is given. However, this architecture has too much overhead and is, therefore, not suitable for embedded systems.

[*]This project has been partially supported by the Austrian Research Promotion Agency.

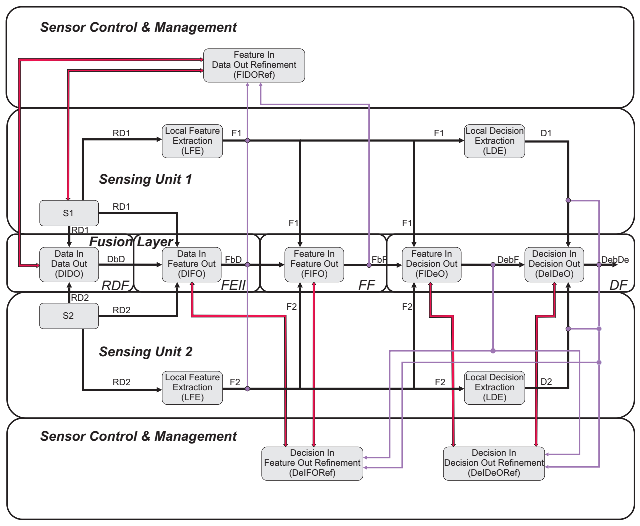

Sensor Control & Management
Feature In Data Out Refinement (FIDORef)
RD1
Local Feature Extraction (LFE)
F1
Local Decision Extraction (LDE)
D1
Sensing Unit 1
S1
Fusion Layer
Data In Data Out (DIDO)
DbD
Data In Feature Out (DIFO)
FbD
Feature In Feature Out (FIFO)
FbF
Feature In Decision Out (FIDeO)
DebF
Decision In Decision Out (DeIDeO)
DebDe
RDF
FEII
FF
DF
S2
RD2
F2
Sensing Unit 2
D2
Sensor Control & Management
Decision In Feature Out Refinement (DeIFORef)
Decision In Decision Out Refinement (DeIDeORef)

Figure 1: Detailed data oriented Fusion Model characterized by input- and output data

3 Fusion Model

Figure 1 presents the detailed, data oriented software fusion model in our I-SENSE approach [7]. This model consists basically of three levels and functions. Namely, the *Sensor Control & Management Unit*, the *Sensing Unit* and the *Fusion Layer*. The figure presents an example of two physical sensors, labelled with $S1$ and $S2$ (i.e. audio sensor and visual sensor).

The *Sensor Control & Management Unit* is responsible for the sensor identification as well as providing the interface to other sensing nodes, human observers and actuators. Furthermore, this unit controls the overall fusion process and provides access to a database where resource requirements for the different fusion tasks are stored. Three essential functional blocks are identified in this layer to allow an online refinement of the overall fusion process based on (i) the generated output decisions of *FIDeO* and *DeIDeO* (*DeIDeORef, Decision in decision out refinement* and *DeIFORef, Decision in feature out refinement*) and (ii) generated output features of *FIFO* and *DIFO* (*FIDORef, Feature in data out refinement*).

The *Sensing Unit* represents the intelligent sensor which consists of the physical sensor itself and a suitable data preprocessor (e.g. resolution based down-sampling, automatic gain control, . . .). A *Local feature extraction Unit (LFE)* is used to extract a single-source feature vector based on color information, structural information, spectral information or acoustic information of an observed object. This means, that each sensor provides an estimate of the position of an object with extracted features, based only on its own single source data. These individual feature vectors are input to a data fusion process, namely the *Feature in feature out (FIFO)* process, in order to achieve a joint feature vector estimate based on multiple sensors. A *Local decision extraction Unit (LDE)* is used to extract local decision from individual objectives features (e.g. classification of objectives identity).

The heart of the framework is the *Fusion Layer* including the following five functional units:

DIDO: *Data in data out unit*; This functional unit is also called *Raw-Data Fusion unit (RDF)*, and raw uncorrelated data will be fused from different and/or similar numerous sensors there. This raw data streams are labelled with *RDx*. In our framework this method is provided for similar sensor types by using wavelet based image fusion techniques of images from visual sensor & infrared spectral camera.

DIFO: *Data in feature out unit*; this is our so called *Feature extraction II unit (FEII)*, where raw data from the individual sensors and/or fused raw-data (i.e. *DbD: Raw data based on raw data*) is used to extract suitable features of the individual tracked objects. These features are found by experimental analyses and/or physical modeling and described in more detail in sections 4 and 5. The output data are feature vectors for each detected object in the observed area.

FIFO: *Feature in feature out unit*; this is our so called *Feature fusion unit (FF)*, where features will be fused to a resulting overall feature vector based on individual objects. Corresponding objects are found by simple object overlapping calculation for similar sensor types and time stamping for different sensor spaces. The output data of this fusion process are fused feature vectors based on features (*FbF*) extracted by the *LFE* unit or features extracted by the *DIFO* unit.

FIDeO: *Feature in decision out unit*; this functional unit is a part of our *decision fusion unit (DF)*, where, a classifier, based on Support Vector Machines (SVM, cp. section 6), is trained with previously recorded and classified sequences. In the next step this SVM is used as a classifier to derive classification decisions based on previously extracted single source feature vectors ore joint feature vectors (*FbF*) from the *FIFO* unit. Outputs are decisions based on Features (*DbF*), and a probability interval of this decision.

DeIDeO: *Decision in decision out unit*; this functional unit is the second part of our *decision fusion unit*, where extracted decisions (*Dx*) will be fused from multiple sensors from the *LDE* unit with fused data from *FIDeO*, based on statistical methods (i.e. Dempster-Shafer methods [2]). Outputs are decisions, based on multiple decisions.

4 Visual Features Extraction

In this section the feature extraction task based on visual features is explained. For our feature extractor we adopted the ideas of Viola and Jones [15] to build a multi-class classificator and improved it using *RealBoost* [11]. The feature set depicted in figure 2 and additional gradient-based information is used in order to generate robust features and calculate them in real-time on an embedded platform. The boosting approach mainly is used to extract the most powerful features while during feature regression phase the overall interpretation of their values could be left to the fusion system (cp. section 6).

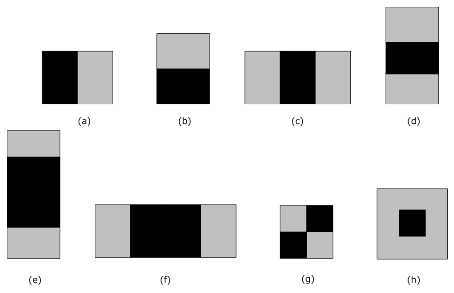


Figure 2: Simple *Haar-like* (a-g) and *center-surround* (h) used in the implementation. Each feature is composed by two or more ”subrectangles” within light rectangles having positive and dark rectangles negative weight.

4.1 Histogram Features

As simple Haar-features perform well in describing global illumination changes and symmetries, they have problems in describing oval shapes and do not use edge information. Yet, these drawbacks can be improved by expanding the feature set with local edge orientation features (EOHs) [8] which store the gradient information in an image in a histogram (cp. figure 3a) depending on the orientation of each edge (cp. figure 3b).

Shortly, in order to use EOHs in an image it has to be preprocessed by an edge detector, e.g. Sobel. The strength of the edge in (x, y) is denoted by $G(x, y)$. Furthermore, a threshold has to be calculated to ignore noise

$$G'(x,y) = \begin{cases} G(x,y) & if\ G(x,y) \geq T \\ 0 & otherwise. \end{cases} \tag{1}$$

Once we have received an edge its orientation can be calculated as

$$\Phi(x,y) = arctan\Big(\frac{G_y(x,y)}{G_x(x,y)}\Big) \tag{2}$$

and the edges are added to the correct K bins[1] in the histogram. The value of the k-th bin is denoted as

$$\psi(x,y) = G'(x,y) \cdot (1-d) \tag{3}$$

Each entry into a bin is multiplied by a weight of $(1 - d)$, where $d \in [0, 1]$ is the distance of the sample from the central value of the bin as measured in units of the histogram bin size. This interpolation has to be done due to boundary effects which may happen if a descriptor or edge shifts smoothly from being in one histogram bin to another or from one orientation to another.

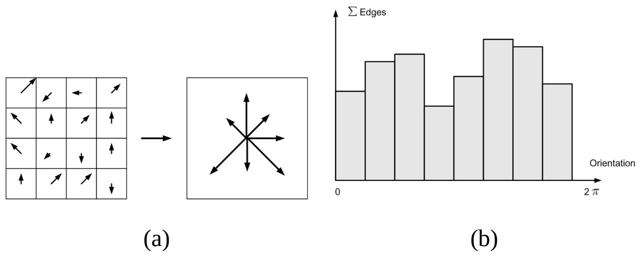


Figure 3: Edge orientation features: (a) First all gradient magnitudes and orientations in a sub-window are calculated and then accumulated into orientation histograms into K orientation groups between 0 and 2π. (b) The typical histogram representation of summed gradient magnitudes classified into eight different orientation groups between 0 and 2π.

4.1.1 Orientation Histogram Features

Once the image has been preprocessed and all edges are added to certain bins depending on their orientation, features can be created which either describe the ratio between two different orientations or the dominant orientation of edges describing an object. The first histogram feature type simply compares two certain bins in the histogram. Equation 4 shows the summation of edges in a sub-window R in the image to their corresponding bin.

$$E_k(R) = \sum_{(x,y)\in R} \psi_k(x,y) \tag{4}$$

Note, that this calculation can be performed very fast with four table lookups by using again integral images as described [3]. The first set of features is then defined as:

$$A_{k_1,k_2}(R) = \frac{E_{k_1}(R)}{E_{k_2}(R)}, \tag{5}$$

which represents the relation between two orientations in the sub-window.

[1]In practice it has been shown that 4, 8 and 16 bins are sufficient in most cases.

The second type of orientation features compares one histogram bin to the summed value of all remaining bins, hence finding the dominant orientation. Equation 6 defines the calculation of the second feature set.

$$B_k(R) = \frac{E_k(R)}{\sum_i E_i(R)} \tag{6}$$

Unlike to simple edge orientation features which only describe single dominant orientations or the relation between two edges, it is possible to define *full histogram features*, where the whole edge information stored in an orientation histogram can be analyzed. These features, though, need some kind of *reference histogram* in order to make the comparison of one and the same sub-window calculation in two different images possible. This histogram, for example, can contain the average histogram values of the positive trainingset or can be a unit vector, if the histograms are considered as vectors, respectively. If one feature has to be evaluated on different samples, all values are calculated with respect to the reference histogram. Due to this, the values in the histogram have to be mapped to a single scalar. This can be done by calculating the Euclid distance, again considering the histograms to be vectors, of the current histogram in a sub-window and the reference histogram as seen in equation 7.

$$F(R) = |A - B| = \sqrt{\sum_k (A_k - B_k)^2}, \tag{7}$$

where $F(R)$ is the scalar gained from the comparison of the current histogram A with the reference histogram B at bin k.

Finally, we use orientation histograms to describe symmetries in a sub-window of an image and also to describe areas where symmetry is missing [12]. The calculation of this symmetry-feature is given in equation 8.

$$Symm(R1, R2) = \frac{\sum_{k\in K} |E_k(R_1) - E_{\#bins-1-k}(R_2)|}{sizeof(R_1)} \tag{8}$$

4.2 Multi-class Feature extraction

Our multi-class feature extractor is build on a tree structure consisting of several binary extractors. While one-vs-all multi-class classifiers sometimes might yield slightly better results we chose the tree scheme to improve computational efficiency.

The single feature extractor of the tree are each trained with positive and negative training examples, where, e.g, in the first stage motor bikes and cars as positive examples are trained against small and large trucks as negative examples. A detailed structure is given in figure 4.

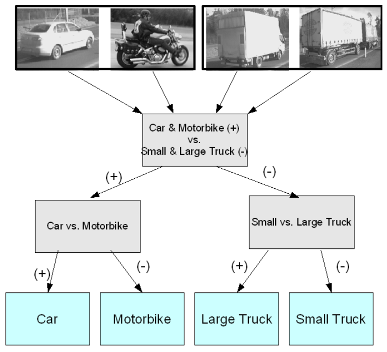


Figure 4: Visual feature extraction tree.

5 Acoustic Feature Extraction

The acoustic feature extractor uses cepstral analysis [10] for feature generation. In the acoustic part of the database noise-induced error handling and modelling of non stationary signal behavior were major issues. Acoustic signatures of moving vehicles mostly behaved non stationary due to variations in engine RPM, gear changes or Doppler effects. Additional wind or environmental noise degraded significantly signal quality. Hence, features from spectral analysis techniques as power spectral estimates [9] or harmonic line association [5], which achieved good classification performance in military approaches, performed poor in our recognition scenario as they assume stationary signal behavior within the analyzed time interval. Cepstral analysis offers some advantages in comparison to these techniques, and it also outperformed other spectral envelope estimation methods like filter bank analysis (Haar transform, channel vocoder) and linear predictive coding (LPC).

5.1 Cepstral Features

In the acoustic feature extractor, cepstral analysis is used for extraction of vehicle characteristic features. Cepstral coefficients (CC) have been popular feature candidates in speech recognition and audio genre classification systems, as they provide very good information packing properties. The cepstrum $c(\tau)$ of a signal $x(t)$ is defined as the inverse Fourier transform of the logarithm of its spectrum:

$$c(\tau) = F^{-1}\left\{\log|F\{x(t)\}|\right\}, \tag{9}$$

where F denotes the Fourier transform. We use low order CCs to capture the slowly varying properties of the spectrum, i.e. the spectral envelope. The whole signal energy, for example, is contained only in the first coefficient (c_0 term), which yields good class discrimination, as trucks usually produce more noise than cars.

CCs are computed either directly using equation 9, or estimated efficiently via linear predictive analysis by converting LPC coefficients into LP based cepstral coefficients. The LPC parameters a_i in an autoregressive (AR) model are directly obtained as system of equations from the autocorrelation function $r(k)$, by solving the so called Yule-Walker equations:

$$\sum_{i=1}^{p} a_i r\left(|k-i|\right) = r(k), \tag{10}$$

where *p* denotes the selected model order. A recursive method for solving these equations is the Levinson-Durbin algorithm [10]. The CCs are then calculated by the following recursion:

$$c_m = \begin{cases} \ln\left(r(0)\right) & \text{for } m = 0 \\ a_m + \sum_{k=1}^{m-1} \left(\frac{k}{m}\right) c_k a_{m-k} & \text{for } m \geq 1 \end{cases} \tag{11}$$

where $a_0 = 1$ and $a_k = 0$ for $k > p$ and *p* denotes the selected model order. This method avoids any signal transformation and thus, the computational effort is highly reduced without significant loss in classification performance. LP based CCs as feature vector for the classifier afford efficient use in real time applications.

5.2 Acoustic Processing

The motor noise of vehicles is mainly limited to the frequency range below $500\ Hz$. In order to suppress tire noise, audio data is lowpass filtered and downsampled to $1\ kHz$. As the vehicles travel quite slowly in our scenario, analysis frames of 2 seconds are used. The input signals are blocked into smaller frames of $N = 256$ samples with 50% overlap. A hamming window is used to reduce leakage effect and the AR modelling order is set to $p = 20$ coefficients. For each block, only the first $p + 1 = 21$ autocorrelation coefficients are calculated and the AR parameter set is achieved by the Levinson-Durbin recursion. The first 16 cepstral coefficients proved best classification results and hence they are converted from the LP parameter set with the above described equations. The mean values over all frames yield the final feature vector for SVM based classification.

6 Decision Modeling

The decision modeling process is provided as a generic software framework which allows online data fusion on a distributed embedded system with limited memory resources. In our multi-level data fusion framework Support Vector Machines (SVM), proposed by Vapnik [14] are used as classification method for decision modeling. For large sets of training data, common SVM learning strategies are not feasible, especially on embedded platforms because of their restricted time and memory resources. Therefore, a modified version of the original SVM, the so called Least Squares Support Vector Machine (LS-SVM) [13] is used for decision modeling in our framework. The main characteristic of LS-SVMs is the lower computational complexity compared with original SVMs, without any quality loss in the classification results.

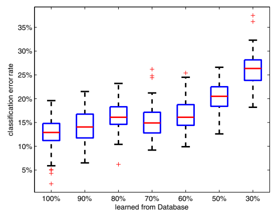
The extraction of support vectors from a given training dataset is comparable with the problem formulation of finding the most significant vectors in a given data set. The optimal solution for solving this task should combine the following features. It should (i) be fast, (ii) lead to a sparse solution (i.e. low number of support vectors) and (iii) produce good classification results.

In [7] we propose a modified nearest neighbor technique for an intelligent preselection of learning data in order to reduce the training set and therefore reduce the number of support vectors which are then used by the LS-SVM classifier. The remaining datasets are used as support vectors for a LS-SVM classifier to find the decision boundary between two classes in the learning process. Using our approach leads to a sparse LS-SVM classifier with good classification results (about 2% higher error rate compared to standard SVM) and lower computational costs(about 70% faster than Standard SVM) and lower memory costs(about 55% less data for storage compared to LS-SVM).

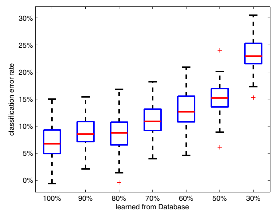
7 Experimental Results

In this section we present the results of our vehicle classification case study. We have implemented a simple multi-class classifier by using the One-against-All technique. The database consists of 3 different classes: (i) large trucks, (ii) small trucks and (iii) cars. We generate a database with about 250 samples for each class – about 200 are used as training samples and about 50 are used as evaluation samples. The samples for training and evaluation are chosen randomly from the database. Each sample is characterized by 4 visual based features (histogram and orientation features) and 16 acoustic based features. For the feature extraction 2 different types of sensors were used: (i) CMOS visual sensors and (ii) acoustic sensors. For classification we use our LS-SVM with support vector preselection for these experiment. For all experiments we used radial basis function (RBF) as the kernel function. The results presented in figure 5 are average values from 100 runs with random selection of training datasets and random selection of the evaluation set (Note: Both sets are disjunctive).

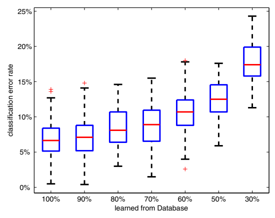
The box plots (lines at the lower, median and upper quartile values; whiskers show the extent of the rest of the data) shown in figure 5 indicate that fusing data from various sensors help to improve the robustness and confidence as well as to reduce ambiguity and uncertainty of the processed vehicle classification. In case we perform our vehicle clas-



(a) visual features



(b) acoustic features



(c) fused feature vector

Figure 5: Experimental results: vehicle classification with SVM based on (a), (b) and (c)

sification based on visual features only (with 100% of the learning database) the mean error rate is about 12.9%. In comparison the same classification based on only acoustic features lead to 6.8% error rate. Fusing data from both information sources decrease the error rate to 6.6%, what is better than the best single source result. The figure also indicates that using smaller sets of training data increases the mean error rate. Using 30% of training samples lead to (i) 26.4% error rate in case based on visual features only, (ii) 22.9% error rate in case based on acoustic features only and (iii) 17.4% error rate in case based on fused features. This fact indicates that fusing data on feature level allows to decrease the number of learning samples in order to gain same classification results than with single source data.

8 Discussion

Vehicle identification is one of the most important tasks in traffic surveillance systems. Our approach is to use different type of sensors in order to fuse the extracted features from the individual sensors. We show that our approach is advantageous in comparison to single source vehicle classification. The results of our experiment demonstrate that the advantage is twofold. Firstly, the classification error rate decreases by using our modified LS-SVM approach. Secondly, the necessary number of training samples can be reduced to obtain quite similar classification results. We use an algorithm for intelligent training data preselection in order to identify a subset of training data which describes the whole dataset best, leading to a reduced number of vectors which have to be stored. These approach makes learning of large training datasets possible even in embedded system with restricted memory and time resources.

We plan to extend our database by a bus class. The decision fusion process based on Demster-Shafer theory and further acoustic feature extraction algorithm will be implemented.

References

[1] M. Bramberger, A. Doblander, A. Maier, B. Rinner, and H. Schwabach. Distributed Embedded Smart Cameras for Surveillance Applications. *Computer*, 39(2):68–75, Feb. 2006.

[2] A. P. Dempster. A generalization of bayesian inference. *Journal of the Royal Statistical Society*, 30:205–247, 1968.

[3] M. Grabner, H. Grabner, and H. Bischof. Fast approximated sift. *In Proceedings of ACCV06*, 2006.

[4] C. A. W. Group. Capstone C4ISR Architecture. http://www.fas.org/irp/program/core/c4isr.htm, 1998.

[5] M. Hohil, J. Heberley, J. Chang, and A. Rotolo. Vehicle counting and classification algorithms for unattended acoustic sensors. *SPIE - The International Society for Optical Engineering*, 5019:99–110, 2003.

[6] A. Klausner, B. Rinner, and A. Tengg. I-SENSE: Intelligent embedded multi-sensor fusion. In *Proceedings of the 4th IEEE International Workshop on Intelligent Solutions in Embedded Systems (WISES)*, pages 105–116, Vienna, Austria, June 2006.

[7] A. Klausner, A. Tengg, and B. Rinner. Enhanced Least Squares Support Vector Machines for Decision Modeling in a Multi-Sensor Fusion Framework. In *Proceedings of the International Conference on Artificial Intelligence and Pattern Recognition (AIPR-07)*, Orlando, US, July 2007.

[8] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[9] E. Munich. Bayesian subspace methods for acoustic signature recognition of vehicles. In *Proceedings of the European Signal Processing Conference (EUSIPCO-04)*, Vienna, Austria, Sept. 2004.

[10] L. Rabiner and B. Juang. *Fundamentals of Speech Recognition*. Prentice hall, Englewood Cliffs, NJ, USA, 1993.

[11] R. E. Shapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning, vol. 37, no. 3, pp. 297-336*, 1999.

[12] C. Sun and D. Si. Fast reflectional symmetry detection using orientation histograms. *Real-Time Imaging*, 5:63–74, 2004.

[13] J. Suykens, P. V. Dooren, B. D. Moor, and J. Vandewalle. Least squares support vector machine classifiers: a large scale algorithm. *European Conference on Circuit Theory and Design (ECCTD'99)*, pages 839–842, 1999.

[14] V. Vapnik. *Statistical Learning Theory*. Wiley, New York, USA, 1998.

[15] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *IEEE,vol. 1, pp. 511-518*, 1(3):511–518, 2001.