IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2004

Feature-Centric Evaluation for Efficient Cascaded Object Detection

Henry Schneiderman[1]

Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
hws@cs.cmu.edu

Abstract

We describe a cascaded method for object detection. This approach uses a novel organization of the first cascade stage called "feature-centric" evaluation which re-uses feature evaluations across multiple candidate windows. We minimize the cost of this evaluation through several simplifications: (1) localized lighting normalization, (2) representation of the classifier as an additive model and (3) discrete-valued features. Such a method also incorporates a unique feature representation. The early stages in the cascade use simple fast feature evaluations and the later stages use more complex discriminative features. In particular, we propose features based on sparse coding and ordinal relationships among filter responses. This combination of cascaded feature-centric evaluation with features of increasing complexity achieves both computational efficiency and accuracy. We describe object detection experiments on ten objects including faces and automobiles. These results include 97% recognition at equal error rate on the UIUC image database for car detection.

1. Introduction

One of the challenges of object detection is coping with variation in object size and location. An object can appear anywhere in an image and at any size. There are two general approaches to this problem. "Invariant" methods attempt to use features or filters that are invariant to geometry [6][7][25][28] or photometric properties [4][15][22]. The challenge of these approaches is finding features that are simultaneously invariant and discriminative, particularly for natural objects such as faces. On the other hand, "exhaustive-search" methods involve building a classifier that distinguishes between the object and "non-object" (any other scenery) while only having to tolerate limited variation in object location and size. These methods find the object by scanning this classifier over an exhaustive range of possible locations and scales in an image. Figure 1 illustrates this process, where the classifier evaluates all possible "windows" in the image as shown by the rectangles. The drawback of this approach is that such an exhaustive search can be very time consuming.

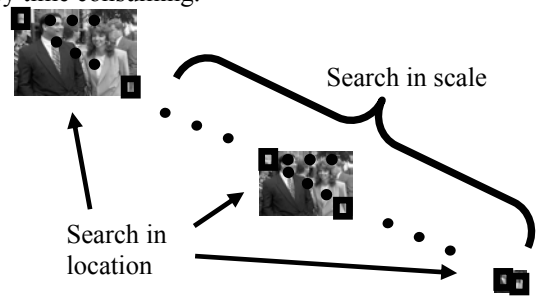
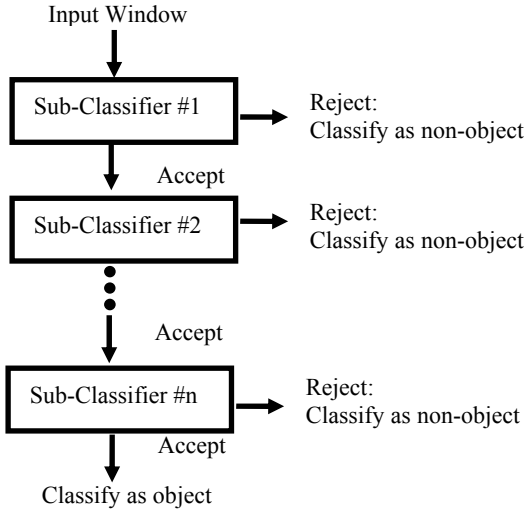


Figure 1. Exhaustive search object detection

One way of reducing the computational cost of exhaustive search is to express the classifier as a cascade of sub-classifiers as shown in Figure 2. Each sub-classifier stage makes a decision to either reject the input, classifying it as non-object or to continue evaluation using the next sub-classifier stage. Those windows that survive the last sub-classifier are classified as the object. This strategy is designed to remove the vast majority of non-object windows with a minimal amount of computation.



[1] This work was supported in part by the Advanced Research and Development Activity (ARDA)'s Video Analysis and Content Extraction Program (VACE) under contract number MDA904-03-C-1789 and the TSWG under contract N41756-03-C-4024.

Figure 2. Cascade of sub-classifiers

The idea of using cascade-like methods has existed for several decades, and, in particular, was used widely in the automatic target recognition techniques of the 1970s and 80s [3]. More recently similar techniques have been used in face detection including [2][9][10][14][17][19][21][23][24][26]. In particular, Viola and Jones [24] and the various enhancements of their method [13][26][27] achieve very fast face detection by using a cascade in conjunction with computationally efficient features based on the integral image.

Most cascades, such as the Viola and Jones[2] [24], use what we describe as "window-centric" evaluation. These approaches compute lighting correction and feature evaluation separately for each window. In this paper, we present an alternative method of "feature-centric" evaluation which re-uses feature evaluations among overlapping windows. Such a sub-classifier can then use more information without the cost of additional feature evaluation. As the first stage in a cascade, such a sub-classifier often reduces the initial number of candidate windows by more than 99%. We show how this can be done with computational efficiency under a few reasonable assumptions.

We use this initial feature-centric sub-classifier stage as part of a complete system for object detection. The early stages in the cascade use simple and fast feature evaluations and the later stages use more complex and discriminative features. In particular, we propose novel features based on sparse coding and ordinal relationships among filter responses. Finally, we show how we learn the classifier cascade using Adaboost with confidence weighted predictions [20]. We demonstrate computational efficiency and accuracy for object detection and show the generality of the approach to ten objects including faces and automobiles.

2. Feature-Centric Evaluation

Most cascade based classification algorithms use what we call a "window-centric" evaluation. In this method all evaluations are with respect to a classification window. This can be stated formally as[3]:

$$r_{k_o}[k] = l(I[k] * w[k - k_o])$$
$$w[k] = 1, \ 0 \le k < m$$

where $r_{ko}[k]$ represents the rectangular window centered at location k_o selected by the window function $w[k]$ from the input image, $I[k]$. $l(\bullet)$computes a lighting correction over this windowed region.

The feature values for each window are then treated independently as shown below:

$$f_1 = f(r_1[k]) \rightarrow c[1] = c(f_1)$$
$$f_2 = f(r_2[k]) \rightarrow c[2] = c(f_2)$$
$$\vdots$$
$$f_n = f(r_n[k]) \rightarrow c[n] = c(f_n)$$

Figure 3. Window-centric evaluation in 1-D. f_{ko} represents a feature evaluation with respect to a window centered at location k_o. $c(\bullet)$ is a classification function. It outputs a continuous range of values, where a positive value indicates the presence of the object and a negative value indicates its absence. $c[k_o]$ represents the classifier output for the window at location k_o.

In this section we describe an alternative "feature-centric" evaluation strategy. This method shares feature values among overlapping windows. The first step computes feature values over a regular grid in the image:

$$f[k_o] = f(I[k - k_o])$$

where $f[k_o]$ represents the feature evaluation at location k_o.

A second step classifies each window using all the feature values that fall within the window's region of support:

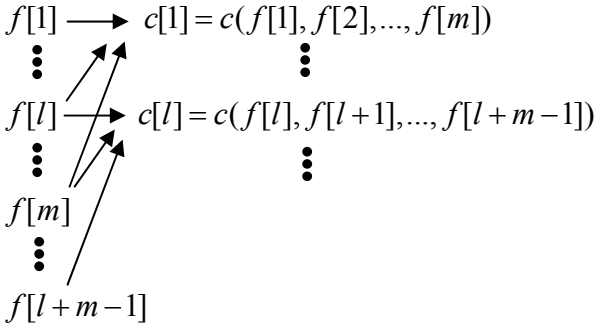


Figure 4. Feature-centric evaluation in 1-D. The region of support of the window spans m feature locations.

The methods in Figures 3 and 4 use the same amount of feature computation, but each feature-centric classification function uses more features than its

[2] All operations after the integral image are window-centric.

[3] All figures and equations in this section are expressed with respect to 1-D signals in the interest of notational simplicity. There are direct counterparts for 2-D signals/images.

window-centric counterpart. This additional information is an advantage in the first stage of a detection cascade. There may be no single feature that is reliable enough to detect all object instances and reject a significant proportion of non-object instances. By using multiple feature values, feature-centric evaluation will lead to more robust performance.

The feature-centric method incurs additional computation cost only in a more complex classification function, $c(\bullet)$. This cost can be minimized under several reasonable simplifications. The first simplification is to write the classification function as an additive model where each term depends on a single feature value:

$$\begin{aligned} c[l] &= c(f[l], f[l+1], ..., f[l+m-1]) \\ &= h(f[l], 0) + ... + h(f[l+m-1], m-1) \end{aligned} \quad (1)$$

Note that the second argument to $h(\bullet)$ specifies the feature's position within the window; that is, even though the same feature function is computed at all locations, its value is interpreted differently by $h(\bullet)$ at each position within the window. This representation preserves geometric structure.

The additive model assumption reduces dimensionality from m dimensions to m models of one dimension. This greatly reduces representational cost and computational needs. However, this restriction to an additive model is not a severe one. Graphical probability models such as Bayesian networks and Markov Random Fields can be expressed as additive sums. As well, additive models produced by boosting and bagging can be quite powerful.

The second simplification represents the features, $f[\bullet]$, as discrete-valued. Under this assumption the function $h(\bullet)$ is a table indexed by feature value and position within the window. The advantage of a discrete representation is that the classifier only needs to perform table look-ups (memory accesses) and additions.

The classification function of each window can be computed separately:

$$c[l] = \sum_{k \le l < k+m} h(f[k], l-k)$$

In this form, the classifier function makes nm look-ups into random (not necessarily sequential) memory and nm additions over n windows where each window's region of support spans m features. However, we can re-organize this computation such that most of these memory accesses will be sequential. In particular, each value of $f[k]$ will contribute to the classification of m windows. We can therefore, store the m values, $h(f[k], \bullet)$, in sequential memory locations. Each of these values can then be added in sequence to running sums of the classification values for the appropriate m windows. Sequential access of these values reduces computational time by at least 50% in our implementation.

Feature-centric evaluation is most useful in the early stages of a cascade where the goal is to quickly remove as many windows as possible. However, once this is accomplished, several restrictions of feature-centric evaluation make it less suitable for later stages in a cascade. In particular, it computes the same feature function, $f(\bullet)$, at all positions within the window, whereas it may be advantageous to compute different types of features at different positions. For example, on a face it may be advantageous to compute horizontal features around the eyes and vertical features around the nose. Also, lighting correction must be computed with respect to a local region about each feature value, as we will describe in the next section.

Window-centric evaluation provides a more general form of equation (1) that is more suitable for later stages in the cascade:

$$\begin{aligned} c[l] &= c(f_{l,1}, ..., f_{l,m}) \\ &= h_1(f_{l,1}) + ... + h_m(f_{l,m}) \end{aligned} \quad (2)$$

where different features can be computed at different locations in the window, i.e., each $f_j(\bullet)$ may have a different region of support within the classification window:

$$\begin{aligned} f_{l,1} &= f_1(r_l[k]) \\ f_{l,2} &= f_2(r_l[k]) \\ &\vdots \\ f_{l,m} &= f_m(r_l[k]) \end{aligned}$$

In practice, our detection cascade usually uses a single feature-centric stage followed by multiple window-centric stages. This ordering achieves good performance in terms of speed and accuracy for detection of many objects. In particular, the feature-centric stage usually removes more than 99% of the non-object windows from consideration.

3. Components of a Cascade Stage

This section describes the components of an individual cascade stage including the sub-classifier structure, the feature representation, the method of lighting correction, and the method of estimating the sub-classifier's model.

3.1. Sub-Classifier Structure

Each sub-classifier stage is represented as a semi-naïve Bayes classifier [11]. A semi-naïve Bayes classifier decomposes the input variables into subsets. The classifier represents statistical dependency within each subset, while treating the subsets as statistically independent. Below we show the semi-naïve Bayes classifier as a log-likelihood test:

$$H(X_1,...,X_n) = \log\frac{P(S_1|\omega_1)}{P(S_1|\omega_2)} + \log\frac{P(S_2|\omega_1)}{P(S_2|\omega_2)} + ... + \log\frac{P(S_m|\omega_1)}{P(S_m|\omega_2)} > \lambda$$
$$S_1,...,S_m \subset \{X_1,...,X_n\} \qquad (3)$$

where $X_1. \ . \ . \ X_n$ are the input variables within the classification window, $S_1. \ . \ . \ S_r$ are subsets of these variables, and ω_1 and ω_2 indicate the two classes. For the problem of object detection, the classes are the object and non-object. For example, ω_1 may correspond to face and ω_2 may correspond to "non-face." In this form, the classifier chooses class ω_1 if $f(X_1,.. .,X_n) > \lambda$. Otherwise, it chooses class ω_2.

The additive models given by equations (1) and (2) derive from this semi-naïve Bayes formulation. Each term in an additive model corresponds to a term in the semi-naïve Bayes form, where each feature, $f(\bullet)$, is a function over a particular subset, S_i, of variables. In particular, for feature-centric evaluaton, equation (1), the relationship with the semi-naïve-Bayes form is given by:

$$h(f[l+k],k) = \log\frac{P(f_k = f(S_k)|\omega_1)}{P(f_k = f(S_k)|\omega_2)}$$

For window-centric evaluation, equation (2), the relationship with the semi-naïve-Bayes form is given by:

$$h_k(f_k) = \log\frac{P(f_k = f_k(S_k)|\omega_1)}{P(f_k = f_k(S_k)|\omega_2)}$$

The semi-naïve Bayes formulation takes advantage of the sparse structuring of statistical dependency. Typically, over a specific class, such as faces, any image variable (e.g., a specific pixel corresponding to the eye location on a face) will have strong statistical dependency with a small number of other variables and weak dependency with the remaining ones. Sparse behavior becomes more pronounced under various transforms of the images, such as steerable filters [8] or a wavelet transform [5]. In this paper the image variables are wavelet coefficients generated by a two level linear phase 5-3 filter bank[4] or a two-level 4-4 filter bank[5].

Graphical probability models such as semi-naïve Bayes concisely represent sparse relationships. **Error! Reference source not found.** describes how we assign variables to subsets by minimizing a series of cost functions related to classification error.

3.2. Feature Representation

Computational considerations govern the complexity of the feature functions, $f_i(S_i)$. In early stages, when many windows have to be evaluated, the feature functions are designed to be computationally efficient. In particular, their range must be restricted to a small set of discrete values (e.g., 10^2) such that the tables, $h(f[k],\bullet)$, can be relatively small. In later stages, when there are fewer windows to evaluate, the feature functions are chosen to be more discriminative and powerful. However, their range must still be restricted to a relatively small number of values (e.g., 10^4) to avoid over-fitting the classification function. Below we describe two feature functions corresponding to these two cases respectively.

3.2.1. Linear Projection and Scalar Quantization

This feature function involves steps of linear projection and scalar quantization. The projection is onto a small set of orthogonal basis vectors. These basis vectors come from the subset's principal components over training examples from one or both classes. Stage 1 typically uses projection on at most 2 vectors. Later stages may use 3 or more basis vectors. This feature function also computes an additional coefficient representing the energy orthogonal to the space of projection.

The feature function quantizes each of the coefficients into a small set of possible values (usually, 7 or 9). The quantization boundaries are fixed functions of the coefficient's empirical variance (e.g., +-1.2σ, etc.). The final step re-expresses these quantized values as a single feature value. For example, one may consider a typical feature formed from 3 projection coefficients. If each is discretized to 7 values, we can represent the overall feature as a single value with a range of 7^3 possible values.

3.2.2. Linear Projection, Sparse Coding, and Ordinal Representation

This feature representation consists of linear projection, sparse coding, and representation of projection coefficient polarity and ordering. This representation concisely represents visual information with stability to appearance changes. It is also supported by reconstruction studies and by work in similar representational strategies [18].

The linear projection step is identical to that of section 3.2.1 except that it uses a greater number of basis vectors

[4] The low-pass coefficients are {-1, 2, 6, 2, 1} and the high-pass coefficients are {2, -4, 2}

[5] The low-pass coefficients are {1, 3, 3, 1} and the high-pass coefficients are {-1, -3, 3, 1}

(e.g., 9 to 15 vectors). The resulting projection coefficients are often sparse; for a given input, many of the projection coefficients will be zero or negligibly small. Of course, different subsets of projection coefficients will be non-zero for different inputs. In our representation, we assume that no more than g of the q projection coefficients will be non-zero for any given input. In particular, there are z such possible subsets chosen from the q coefficients. Where,

$$z = \sum_{i=0}^{g} \binom{i}{q}$$

The advantage of making this assumption is that it reduces dimensionality. We no longer have to represent a q dimensional space. The largest space we have to represent is g-dimensional. However, we must represent a collection of z of these smaller subspaces.

This feature representation consists of two parts. First, we need to encode which combination of coefficients is non-zero. As we mentioned there are z such possible combinations. We make this determination by thresholding the absolute values of the q projection coefficients. (The actual threshold is determined by human observation such that values less than the threshold do not have a noticeable impact on appearance.) If more than g of the coefficients are above threshold, the g largest coefficients are chosen.

Secondly, the feature encodes the coefficients within the chosen subset. A direct representation would be to quantize each of the projection coefficients into r values. Such a quantization would represent the subset by r^g possible values over a g dimensional space. However, a more compact representation is desirable. We take advantage of the observation that the polarity and ordinal relationships among these g projection coefficients are more important for human perception than their absolute values [18]. In particular, we only represent the polarity (i.e., the sign) of each coefficient and the index of the largest coefficient. If there are i coefficients in the subset, then this encoding involves, $2^i i$ possible values.

Overall, such a feature function encodes S_i, a subset of wavelets, into v possible values:

$$v = \sum_{i=0}^{g} \binom{i}{q} 2^i i$$

For example, a feature based on $q = 9$ and $g = 4$, has $v =$ 10,387 possible values.

3.3. Lighting Correction

Feature-centric lighting correction is computed in terms of local neighborhoods. In particular, we normalize each wavelet coefficient as a function of its neighbors. For a high-pass coefficient, we use the following normalization:

$$v(x,y,b,s) = \frac{w(x,y,b,s)}{\delta + \sum_{b' \neq ll} \sum_{i=-1,j=-1}^{i=1,j=1} abs(w(x+i,j+j,b',s))}$$

where, x,y is the position of the coefficient, b is the band of the coefficient, s is the scale, and δ is a small positive constant to avoid division by zero.

For coefficients that are low-pass in both directions ($b = ll$) we use the following normalization:

$$v(x,y,b=ll,s) = \frac{w(x,y,b=ll,s)}{\delta + \sum_{i=-1,j=-1}^{i=1,j=1} w(x+i,j+j,b=ll,s)}$$

All feature-centric features are then computed using these normalized coefficients.

Lighting correction for window-centric evaluation can be computed as a direct function of the pixels within the window. In general, there are no perfect methods to adjust for all possible forms of lighting variation. Our approach is to use multiple methods (usually a different one for each stage) such that overall result will not be critically sensitive to the failure of any individual method. These methods include (A) a scalar shift and scalar scale factor such that the input pixels have a zero mean and unit variance, (B) a scalar scale factor such that the largest gray level is a fixed value, and (C) multiple evaluation over several scalar scale factors and selection of the largest response of equation (3). We also consider multiple regions of support for the first two methods.

3.4. Estimation of the Classifier

Using the feature representation of section 3.2, each sub-classifier is re-written as:

$$H(f_1,...,f_m) = \log \frac{P(f_1|\omega_1)}{P(f_1|\omega_2)} + \log \frac{P(f_2|\omega_1)}{P(f_2|\omega_2)} + ... + \log \frac{P(f_m|\omega_1)}{P(f_m|\omega_2)} > \lambda$$

We estimate the parameters of this model from labeled training examples. The most direct method is to estimate the individual probability distributions separately. Since each f is discrete-valued, this simply involves counting the occurrences of each value of f. Such a method assumes each training example is of equal weight. However, different weightings of the training data may lead to lower classification error for the overall classification function $H(f_1,\ldots,f_m)$. Therefore, it is more desirable to estimate the overall $H(f_1,\ldots,f_m)$ such that we directly minimize classification error. One way of doing this is to use Adaboost with confidence weighted predictions [20] with $H(f_1,\ldots,f_m)$ as the base classifier. Adaboost is guaranteed to minimize error on the training set and minimizes a bound on the generalization error. Since our base classifier is a table, the final classifier (a weighted sum of instantiations of the base classifier) can be collapsed back to a table that is the same size as that of

the base classifier. The final result can be interpreted as an estimation of the probability distributions under a re-weighted distribution of the training data.

This procedure can extend across stages. In stage *K*, the previous classifier up to stage *K-1* could be considered as the first hypothesis in the Adaboost procedure generating weights for the K^{th} stage.

The non-object training data for each stage is generated by a boot-strapping procedure [23] where the classifier derived from the previous *K-1* stages is applied to large database of non-object images.

4. Experiments

We trained object detectors for ten classes of objects. Where possible, we have evaluated performance on standard test sets for object detection, such as those available for faces and automobiles. However, no standard test sets exist for the remaining objects, so we created our own test sets, striving to achieve realistic and varied conditions involving lighting, occlusion, and background scenery.

4.1 Automobiles

We trained a detector for detecting side views of passenger cars. This detector operates on a 32x96 input window and can evaluate a 300x200 image in less than 1 second on an Athalon 1.8GHz processor.

We evaluated the accuracy of our car detector on the UIUC Image Database for Car Detection[6]. This test set consists of 170 images containing 200 cars. The table below shows a comparison of our recognition rate to other methods at equal error rate (number of false detections = number of misses). In this experiment we used the evaluation software provided by UIUC to compute the performance.

Ours	Fergus, et. al. [6]	Agarwal and Roth [1]
97.0%	85.5%	79.0%

Table 1. Recognition rate at equal error rate on the UIUC Image Database for Car Detection.

4.2. Frontal Faces

We trained a frontal face detector to operate on a 32x24 input window. This detector evaluates a 300x200 image in under 180 ms on an Athalon 1.8GHz processor. The computation time is over an order magnitude faster than an algorithmically equivalent implementation using window-centric evaluation.

[6] http://l2r.cs.uiuc.edu/~cogcomp/Data/Car/

We evaluated the accuracy of our frontal face detector on the MIT-CMU test set for frontal face detection[7] (excluding five images of hand-drawn faces). Below we compare our performance to others on this test set by indicating the number of false detections at various recognitions rates:

	89.7%	93.1%	94.4%	94.8%	95.7%
Ours	6	19	29	35	46
[24]	31	65	--	--	--
[17]	--	--	--	78	--
[19]	--	--	65	--	--

Table 2. False alarms as a function of recognition rate on the MIT-CMU Test Set for Frontal Face Detection.

4.3. Faces with Out-of-Plane Rotation

We also trained detectors for finding faces from profile view. We built a composite face detector combining two profile detectors (one for left profile and one for right profile) with the frontal detector. The entire detector requires 630 ms for 300x200 image. Below we give our performance on the CMU Face Detection Test Set[8] for Non-Frontal Face Detection:

	85.5%	87.0%
Ours	---	86
Schneiderman and Kanade [19]	91	--

Table 3. False alarms as a function of recognition rate on the CMU Test Set for Non-Frontal Face Detection.

These evaluation rates are comparable, but the computation time for our algorithm is more than an order of magnitude faster than that of [19]

4.4. Other Objects

We trained and tested detectors for various objects including stop signs, tea cup, sugar bowl, thermos, creamer, tea kettle, coffee mug, and push cart. For each of these objects we gathered our own test sets. In Tables 4 – 11 show results of various detection experiments on these objects. Each column corresponds to a different value for the detection threshold. Figure 8 shows some representative examples from these test sets.

[7] http://vasc.ri.cmu.edu/idb/html/face/frontal_images/index.html

[8] http://vasc.ri.cmu.edu/idb/html/face/profile_images/index.html

Recognition rate	89.6%	90.4%	93.4%	93.9%
False detections	2	40	67	453

Table 4. Stop sign detection over 114 images.

Recognition rate	73.0%	80.9%	88.8%	94.4%
False detections	2	9	18	66

Table 5. Tea cup detection over 89 images.

Recognition rate	91.1%	93.3%	96.7%	98.9%
False detections	2	8	23	30

Table 6. Sugar bowl Detection on 90 images.

Recognition rate	61.4%	77.3%	84.1%	88.6%
False detections	15	46	86	247

Table 7. Thermos detection on 44 images.

Recognition rate	81.8%	91.5%	96.8%	98.9%
False detections	7	17	42	94

Table 8. Creamer Detection on 94 images

Recognition rate	80.9%	95.7%	97.9%
False detections	1	19	68

Table 9. Tea kettle Detection on 47 images.

Recognition rate	76.0%	88.4%	94.2%
False detections	14	71	313

Table 10. Mug Detection on 121 images.

Recognition rate	70.4%	90.7%	98.1%
False detections	10	53	95

Table 11. Push Cart Detection on 54 images

5. Conclusion

This paper describes the features representation used for a cascaded object detection system. It presents the "feature-centric" computational strategy allowing for a discriminative first stage with little penalty in computational cost. It also present a novel feature representation using sparse coding and ordinal relationships. This combination allows for efficient and accurate object detection.

References

[1] Agarwal, S. and Roth, D. "Learning a Sparse Representation for Object Detection". ECCV 2002.

[2] Amit, Y. 2000. A Neural Network Architecture for Visual Selection. *Neural Computation*. 12:1059-1089.

[3] Bhanu, B. "Automatic Target Recognition: a State of the Art Survey." IEEE Trans. on Aerospace and Electronic Systems, 22, pp 364-379. 1986

[4] Chen, H., Belhumeur, P. and Jacobs, D. "In search of Illumination Invariants"*, CVPR pp. 254-261, 2000.*

[5] Field, D. J. "Wavelets, Vision, and the Statistics of Natural Scenes." Phil. Trans. R. Soc. London A. v.357, pp.2527-2542, 1999

[6] Fergus, R., Perona, P., Zisserman, A. "Object Class Recognition by Unsupervised Scale-Invariant Learning". CVPR 2003.

[7] Forsyth, D.A., et. al., "Invariant Descriptors for 3D Recognition and Pose," PAMI, 13:10, 1991

[8] Freeman, W. T. and Adelson E. H. "The design and use of steerable filters". *PAMI* 13:9, pp. 891 - 906, 1991.

[9] Geman, D. and Flueret, F. 2001. Coarse-to-fine Face Detection. *IJCV*. 41:85-107.

[10] Heisele, B., Serre, T., Prentice S., Poggio, T. "Hierarchical Classification and Feature Reduction for Fast Face Detection with Support Vector Machines." Pattern Recognition, Vol. 36, No. 9, 2007-2017, 2003.

[11] Kononenko, I. "Semi-Naïve Bayesian Classifier." Sixth European Working Session on Learning. pp. 206-219. 1991

[12] Lakshmi Ratan A., Grimson, W.E.L., Wells, W.M., "Object detection and localization by dynamic template warping, IJCV, 36(2):131-148, 2000

[13] Li, S.Z., Zhang, Z.Q., Shum, H., Zhang, H.J. "FloatBoost Learning for Classification". *NIPS* 16, 2002

[14] Mikolajczyk, K., Choudhury, R. and Schmid, C. "Face detection in a video sequence - a temporal approach." CVPR, vol. 2, 96-101, 2001

[15] Nagao, K., Grimson, W.E.L. "Using Photometric Invariants for 3D Object Recognition,", CVIU,71(1):74-93, 1998

[16] Rajagopalan, A.N. and Chellappa, R., "Higher-Order Statistics- Based Detection of Vehicles in Still Images", Jl. Optical Society of America:A, Vol. 18, pp. 3037-3048, 2001

[17] Roth, D., Yang, M-H., Ahuja, N. A SNoW-Based Face Detector. *NPPS-12*. 1999

[18] Sadr, J., Mukherjee, S., Thoresz, K., Sinha, P. The Fidelity of Local Ordinal Encoding. NIPS *14*, 2002

[19] Schneiderman, H., Kanade, T. "A Statistical Method for 3D Object Detection Applied to Faces and Cars". CVPR, 2000.

[20] Shapire, R. E. and Singer, Y. "Improving Boosting Algorithms Using Confidence-rated Predictions." *Machine Learning* 37(3):297-336.

[21] Sinha, P., Torralba, A. "Detecting Faces in Impoverished Images. MIT AI Memo 2001-028, 2001.

[22] Slater, D. and Healey, G. "The Illumination-Invariant Recognition of 3D Objects Using Local Color Invariants." PAMI. 18:2, pp. 206 – 210. 1996.

[23] Sung, K-K., Poggio, T.. "Example-Based Learning for View-Based Human Face Detection". *PAMI*, 20(1):39-51. 1998

[24] Viola, P. and Jones, M. "Rapid Object Detection Using a Boosted Cascade of Simple Features." CVPR, 2001.

[25] Wood, J. "Invariant Pattern Recognition: A Review." *Pattern Recognition*. 29:1. pp. 1 – 17. 1996

[26] Wu, J., Rehg, J. M. and Mullin, M. D. "Learning a Rare Event Detection Cascade by Direct Feature Selection". *NIPS 2003*.

[27] Zhang, Z.Q., Zhu, L, Li, S.Z., Zhang, H.J.. "Real-Time Multi-view Face Detection". *5th International Conference on Automatic Face and Gesture Recognition*. 2002

[28] Zisserman, A., et. al. "3D Object Recognition Using Invariance". Artificial Intelligence 78(1-2): 239-288, 1995.

[29] Schneiderman, H. "Learning Statistical Structure for Object Detection." *Computer Analysis of Images and Patterns (CAIP), 2003*, Springer-Verlag, August, 2003

Figure 5. Face detection

Figure 6. Car detection

Figure 7. Stop sign detection

Figure 8. Detection of tea cup, tea kettle, coffee mug, sugar bowl, thermos.