A Time-Efficient Cascade for Real-Time Object Detection: With applications for the visually impaired.

X. Chen and A.L. Yuille
Dept. Statistics, UCLA.
xrchen@stat.ucla.edu, yuille@stat.ucla.edu

In: 1st International Workshop on Computer Vision Applications for the Visually Impaired (CVACVI). Workshop in association with CVPR 2005. June 20. San Diego. Organizers: J. Coughlan (SKERI) and R. Manduchi (UC Santa Cruz).

Abstract

Real-time object detection is essential for many computer vision applications. Many rapid detection algorithms are based on using cascades of tests. But existing design criteria for cascades either ignore the time complexity of the tests or make over-simplified assumptions about them. This paper gives a criterion for designing a time-efficient cascade that explicitly takes into account the time complexity of tests (as evaluated by computer run time) including the time for pre-processing. We design a greedy algorithm to minimize this criterion (noting that the full problem is NP-complete). Finally, we illustrate our method on the task of text detection in city scenes. This gives a text detection algorithm that runs at 0.025 seconds per 320×240 image, which is equivalent to 40 frames per second. This is a speed up factor of 2.5 compared to our previous text detector. It gives a real-time system which can be used for applications to help the blind and visually impaired.

1. Introduction

Real-time object detection is essential for many real world vision applications. Recent work shows that certain objects, such as faces [1,2,3,4] and text [5,6,7,8,9], can be detected reliably and fairly quickly. But the detection speed is not always fast enough for applications. The goal of this paper is to provide a technique, time-efficient cascades, that can speed up object detection.

We are particularly interested in designing computer vision systems to help the blind and visually impaired. In particular, we want algorithms which detect text in city scenes so that the text can be enhanced for visually impaired subjects, or read aloud to blind subjects. These applications require real-time search through large images (e.g. 1,600 by 1,200) with high quality performance (as measured by false positive and false negative rates). Our previous work gave acceptable performance [9] but its speed was not quite fast enough when evaluated by visually impaired subjects. The techniques described in this paper enable us to obtain real-time detection.

Arguably the most efficient way to build a real-time object detection system is to use the hierarchical structure known as a *cascade* [10]. A cascade is a degenerate decision tree and a typical example is shown in Figure 1. Each level (or layer) of a cascade is implemented by a classifier evaluated on image features.

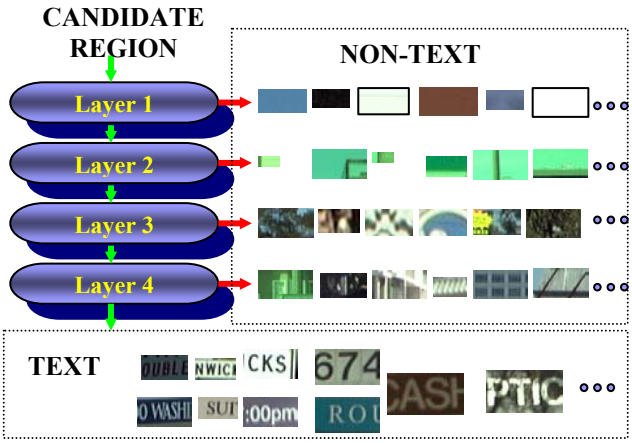


Figure 1. A cascade detector for text detection.

A few layers of classifiers are often sufficient to rapidly reject many candidate regions.

The success of cascades for object detection tasks, such as face detection [4] or text detection [9], is based on the fact that most parts of an image contain no objects of interest. The key insight of cascades is to use simple, rapidly computable, classifiers to reject the parts of the image which do not contain objects while preserving those parts of the image which contain the objects. Then more complex, and time consuming, classifiers need only be applied to limited parts of the image.

Intuitively, the cascade is like carving a sculpture. At the beginning, large cuts are made to quickly give the

rough shape. Then smaller, more precise, cuts are made to refine the details.

The design of efficient cascades requires a trade-off between the time-complexity of the classifiers and performance factors such as false positive rates. But previous work on cascades, and more generally on decision trees, has ignored the time-complexity of classifiers. This is reasonable if the classifiers used at different levels of the cascade are of similar time complexity. But it can be highly suboptimal if, as often happens, the classifiers in the cascade have different time complexity.

For example, Viola and Jones [4,12] designed a cascade for face detection using a criterion based only on the maximum acceptable false positive rates and the minimum acceptable detection rates per cascade layer. The classifiers for their cascade were strong classifiers constructed by AdaBoost [11,12] in terms of weak classifiers based on Haar basis functions. These weak classifiers are of roughly similar time complexity, but the strong classifiers could consist of variable numbers of weak classifiers and so had different time complexity.

The classifiers were even less likely to have similar time complexity in our previous work [9]. Our classifiers were also learnt by AdaBoost from weak classifiers. But our weak classifiers were based on a variety of different image features and so had variable time complexity. Some weak classifiers were based on simple texture features and could be computed rapidly. Other weak classifiers were based on spatial properties, such as spatial relations between edges, and required far more computation.

Classifiers are even more likely to have variable time complexity if we expand the vocabulary of classifiers to include other techniques such as Support Vector Machines [15]. Moreover, even algorithms like Viola and Jones [4] and our own work [9] included pre-processing stages which should also be included in time complexity calculations. We conclude that a well designed cascade must take time-complexity into account.

Somewhat similar arguments have recently been given by Geman and Blanchard for designing decision trees for scene understanding [13]. They give a mathematical analysis of this problem and prove the advantages of a coarse to fine processing strategy. But their approach is very different to ours and it is unclear how to apply their results to the problems which we wish to solve.

In this paper, we give an optimization criterion for designing time-efficient cascades which takes into account the average processing time of each cascade layer as well as the false positive rate (we require that the false negative rate is zero on the training dataset). It is known that cascade design is a NP-hard problem [14] and cannot be solved by any known polynomial time algorithm. Instead we design a greedy algorithm which finds a solution to our criteria. We test this approach on a text detection problem where we can compare this time-efficient classifier to our previous classifier (obtained using Viola and Jones' classifier design [4]). We show that for this problem the time-efficient classifier performs 2.5 times as fast as the our previous classifier [9]. This speed up is sufficient to make our text detection algorithm work in real-time and hence be useful for applications to assist the visually impaired.

This paper is organized as follows. Section 2 presents the optimization criterion for designing time-efficient cascades and describes the greedy algorithm to obtain the cascade. In section 3 we apply the method in a real application of detecting street signs from city scene. We conclude in section 4 and briefly describe future work.

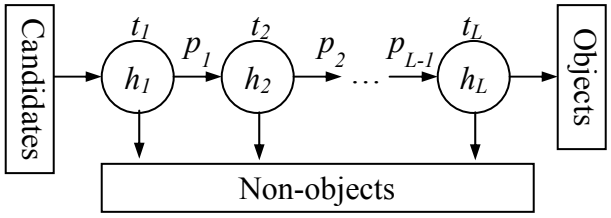


Figure 2. Diagram of a cascade
h_i is the classifier of the I layer, t_i is the average processing time, p_i is the positive rate.

2. Time efficient design of cascade

In this section we give an optimization criterion for designing a time-efficient cascade for object detection. We note that finding a global optimal cascade is an NP-hard problem. Then we define a greedy algorithm to search for minima of the criterion.

2.1. Formulation

The basic idea of a cascade is illustrated in Figure 2. The cascade is run on a set of candidate regions in the input image, which are typically image windows at a variety of scales and positions. The cascade consists of a sequence of binary-valued classifiers h_l. The classifiers are designed so that they have no false negatives (at least on the training data). A candidate region is rejected if it fails the classifier test at any level, in which case it is not necessary to apply the classifiers of the remaining levels.

The time complexity of the cascade depends on the time t_l taken by the classifiers h_l and by the number of regions that the classifier needs to process. Classifiers at

early levels of the cascade will typically be run on many image regions, and so will need to be fast. Slower, and more sophisticated, classifiers can be used at later levels of the cascade.

Formally, we define a set of classifiers H and a training set S of images hand segmented into regions of objects and non-objects (in our experiments, the objects are text). A cascade c is a set of classifiers $h_1,\ldots,h_L$ where the number L of layers is a variable. Then the average processing time for each region is:

$$t_c = t_1 + \sum_{l=2}^{L}\left(t_l \prod_{i=1}^{l-1} p_i \right) \qquad (1)$$
$$t_l \square t(h_l; h_{l-1},..,h_1)$$
$$p_i \square p(h_i; h_{i-1},..,h_1)$$

where h_l is the classifier in the l^{th} layer, p_i is positive rate of the i^{th} layer (the sum of the false positives and true positives) and t_l is the processing time of the l^{th} layer. t_l consists of both the feature calculation time and the classifier decision time (it is calculated empirically). Note that t_l and p_i depend on the classifiers selected in all the previous layers.

The classifier h_l is trained only on image regions which have been evaluated as objects by the previous layers. The classifiers are constrained to have no false negatives on the training set.

If the classifiers all take equal time, then equation (1) reduces to a formula for the average number of features used for each image region, see Viola and Jones [4]. But they used this formula only to compute the number of features used and not for classifier design.

Now we define an optimization criterion for designing a time-efficient cascade. The goal is to find the cascade c^* which minimizes the average processing time t_c but maintains a low false positive rate and has no false negatives (on the training set). Formally, we seek

$$c^* = \arg\min_{c}(t_c)$$
$$\prod d_l > D_T, \prod f_l < F_T \qquad (2)$$

where F_T and D_T are the preset threshold of overall false positive rate and detection rate. f_l and d_l are, respectively, the false positive rate and the detection rate of the l^{th} layer in the cascade c^*.

In this paper, we require d_l to equal 1, (this is possible by using classifiers trained by asymmetric AdaBoost [12]). Hence the requirement on $\prod d_l > D_T$ is automatically satisfied (with D_T equal to 1).

Solving equation (2) for c^* is non-trivial because it is known that designing binary decision trees is an NP-complete problem [6]. This means that no known polynomial time algorithm can solve the decision problem:

Given t_0, is there a c that $t_c < t_0$? (3).

Instead we design a greedy algorithm which attempts to find a solution to the optimization criterion in (2).

2.2. A Greedy Algorithm

Our greedy algorithm is motivated by the decision problem (3). We initialize t_0 and define a greedy algorithm to see if we can solve the decision problem. If we cannot, then we raise t_0 until we can. If we can solve the decision problem, we lower the value of t_0 until we cannot solve it. This gives a solution to the optimization criterion given in (2).

We now define the greedy algorithm for fixed t_0. Let

$$t_0^1 = t_0, \;\; t_0^l = \frac{t_0^{l-1} - t_{l-1}}{p_{l-1}}, \qquad (4)$$

with t_{l-1} and p_{l-1} defined by equation (1), then we select

$$h_l = \arg\max_{\substack{h \in H \\ t_0^l > t(h; h_{l-1},\ldots,h_1)}} \left(\frac{t_0^l - t(h; h_{l-1},\ldots,h_1)}{p(h; h_{l-1},\ldots,h_1)} \right) \qquad (5)$$

as the classifier in l^{th} layer of the cascade. A mathematical explanation for the algorithm is given in Section 2.3. Roughly speaking, we pick the l^{th} classifier to maximize the expected time remaining normalized by the expected number of regions remaining to be rejected.

The algorithm defined by equation (5) has the intuitive property that it encourages fast classifiers with small false positive rates. Empirically, t_0^l usually increases as l increases. This means that simple features and classifiers are preferred in the early layers and more complex features and classifiers are discouraged until t_0^l is large.

We apply equation (5) iteratively to construct the cascade. The algorithm terminates with *SUCCESS* when we have achieved the desired false positive rate F_T, or with *FAIL* if we cannot achieve this rate within time t_0. Algorithm 1 shows the details of the algorithm.

Algorithm 1. Solving the Decision Problem

- F_T = overall required false positive rate
- S = set of training images
- P = set of positive examples

- N = set of negative examples
- H = set of classifiers
- $l = 0$
- $F_l = 1$; $t_0^{l+1} = t_0$
- While $F_l > F_T$
  - $l \leftarrow l + 1$
  - Train H, $\forall\ h \in H$, calculate t_h, f_h, p_h using S
  - If $\forall\ h \in H$, $t_h \geq t$, then return *FAIL*
  - Select h_l according to (5)
  - $F_l \leftarrow F_{l-1} * f_h$
  - Calculate t_0^{l+1} according to (4)
  - Update N using current cascade
- Return *SUCCESS* and get t_c according to (1)

Note that the positive examples P are all the object region examples in the training set S of images. The negative examples N are a subset of the non-object regions in the training set (there are too many non-object regions to use).

If algorithm 1 results in *SUCCESS*, then we know that there is an algorithm that can obtain the desired false positive rate in time smaller than t_0. But there may be an even faster algorithm. We therefore replace t_0 by t_c and reapply algorithm 1. This is illustrated by Algorithm 2.

The training time for this algorithm is much longer than previous methods, e.g. Viola and Jones [4]. But this is affordable since the training process is performed off-line.

Algorithm 2. Training a cascade detector

- Set time t_0
- Set overall false positive rate F_T
- While Algorithm1 is *SUCCESS*
  - $t_0 \leftarrow t_c$
- Return the last successful cascade and its t_c

2.3. Justification for the Greedy Algorithm

In this section we give a mathematical justification of the greedy algorithm.

First, the time remaining after first m-1 layers is

$$t_0 - (t_1 + \sum_{l=2}^{m-1}\left(t_l \prod_{i=1}^{l-1} p_i \right))$$

The remaining layers will take time cost

$$t_c^m \prod_{i=1}^{m-1} p_i$$

where,

$$t_c^m = t_m + \sum_{l=m+1}^{L}\left(t_l \prod_{i=m}^{l-1} p_i \right)$$

We can solve the decision problem provided we can satisfy the following condition for all layers.

$$t_c^m \prod_{i=1}^{m-1} p_i < t_0 - (t_1 + \sum_{l=2}^{m-1}\left(t_l \prod_{i=1}^{l-1} p_i \right))$$

This is equivalent to finding $h_m,\ldots,h_L$ s.t.

$$t_c^m(h_m,...,h_L) < \frac{t_0^{m-1} - t_{m-1}}{p_{m-1}},$$

where t_0^m is defined in sub-section 2.1. The algorithm proceeds by finding classifiers iteratively. We select classifier h_{m-1} to make $\frac{t_0^{m-1} - t_{m-1}}{p_{m-1}}$ as big as possible. This maximizes the bound for t_c^m and so maximizes the time available for the remaining layers.

2.4. Preprocessing

The time cost of a classifier consists of the feature calculation time and the decision time of the classifier. Many applications employ some kind of preprocessing to save time for feature calculation, for example, building integral images for Harr basis features [4], or calculating filter response maps for differential features. We provide a way to incorporate preprocessing time into our algorithm for cascade design.

Suppose $n = |I|$ is the size of a image I. $T_{pre}(n)$ is the preprocessing time for a set of features X. $K(n)$ is the number of all candidate regions in the image I. Note for a feature set X, the preprocessing time is only counted once, when the set of feature is first used in the cascade. Suppose the lth layer is the first layer that uses features from X, then the average preprocessing time per candidate is

$$t_{pre}^{l} = \frac{T_{pre}(n)}{K(n)\prod\prod_{i<l} p_i} \quad (6)$$

If n is fixed, e.g. fixed image resolution, or K(n) $\propto$ n and $T_{pre}(n) \propto$ n, e.g. building the integral image and calculating differential filter response map, then t_{pre}^{l} only depends on the positive rates of previous layers. So it can be directly added to the corresponding t_h in the algorithm.

Equation (6) implies that when l is big (i.e. a feature set is first employed in a late layer), preprocessing may not save time, since t_{pre}^{l} will always be very big. In such cases we should not use preprocessing.

3. Application to text detection

In this section, we apply the proposed algorithm to the text detection application described in our previous publication [9] which used Viola and Jones' classifier design principle. Our new time-efficient text detector cascade runs 2.5 times faster and gives small improvement on the detection and false positive rates. (Note that comparison of our method to other text detection methods has been made in [9]). The following experiment focuses only on the improvement of using a time-efficient cascade.

3.1. The datasets

Both the training dataset and testing data set were enlarged significantly compared to our previous work [9]. There are now 15,478 text samples taken from 1,174 text images, which consists of combining our previous text images with a benchmark training set [16]. Negative training samples were taken from 4,000 large images without text.

Overall we have 930 images with resolution 1,600 by 1,200, mostly taken by blind and visually impaired people in a variety of scenes, including offices, supermarkets, streets, bus stops, subway stations, and restaurants. We use 400 images as the training set *S*. The test set contains the rest 530 images.

3.2. Feature set

The feature set, used to construct the classifiers, is the same as in our previous work. For completeness, we briefly summarize them below. Details can be found in [9].

- Gray level means and variances calculated on blocks.
- First order differential features calculated in blocks.
- Histogram features of intensity and gradient.
- Edge linking features.

3.3. Classifier set

We use the same set of classifiers as in our previous work [9], which enables direct comparison of our time-efficient cascade to our previous cascade.

Details are given in [9]. Briefly, the weak classifiers are linear separable planes in one and two dimensional feature space. AdaBoost is at each layer to combine weak classifiers into a strong classifier..

3.4. Results

The training time for the time-efficient cascade was more than ten times longer than our previous method. But this computation is off-line and so is not significant.

Table 1 compares the speed and performance of the time-efficient cascade compared to our previous cascade. Observe that we get a speed-up of about 2.5 and slightly improved performance.

Table 1. Experiment results

	Old	New
Number of layers	4	10
False positive	5×10^{-6}	2×10^{-6}
Detection rate	89%	91%
320*240 image	0.067s	0.025s
640*480 image	0.26s	0.1s
1600*1200 image	1.54s	0.59s

3.5. Implemented systems

The text detector was tested in two systems to help blind and visually impaired people navigate in streets. One system is called the "*Smart Telescope*". It is designed for people with low vision, and is shown in Figure 3. The system consists of three parts: A camera, a portable computer and a micro-screen mounted on eye glasses. The camera captures and transfer 640 by 480 images to the computer in real-time. The computer runs the cascade detection algorithm, finds and marks the regions of text in the image, and shows an enhanced image on the micro-screen. Visually impaired people can select a region using wireless mouse and enhance for reading. As shown in table 1, the text detector can

process 640 by 480 images at 10 frame per second. This is fast enough for practical use.

The other system called "*Signfinder*" is designed for blind people. The user has a hard-head mounted digital camera connected to a laptop computer. Once a picture is taken, it is automatically transferred to the laptop and processed by our text detector. The picture resolution is 1,600 by 1,200. On average the time-efficient cascade detector takes about 0.6 second to process a picture which is fast enough for practical use. Figure 4 shows some of the detection results.

Figure 3. Smart telescope system

4. Conclusion and future work

In this paper, we formulated an optimization criterion for designing a time-efficient binary cascade taking into account the time complexity of classifiers. We developed a greedy algorithm for training efficient cascades motivated by our criterion (noting that the full problem is NP-complete). Our approach can be applied to design cascades using any type of classifiers. The techniques in this paper can be extended to a greater range of problems, including multi-object detection.

We illustrated our approach on the problem of text detection. The time-efficient cascade was shown to be on average 2.5 times fasters than our previous cascade [9] which was designed using the criterion given in [12].

This results in two systems, Smart Telescope and Signfinder, which use text detection to design systems to assist the blind and visually impaired.

References

[1] H. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection", *In IEEE Trans. PAMI*, vol. 20, 1998.

[2] F. Fleuret, and D. Geman, "Coarse-to-Fine face detection", *International Journal of Computer Vision*, June, 2000.

[3] H. Schniederman and T. Kanade, "A Statistical method for 3D object detection applied to faces and cars", *Proc. of Computer Vision and Pattern Recognition*, 2000.

[4] P. Viola and M. Jones, "Robust Real-Time Face Detection", *International Journal of Computer Vision*, 57(2), 137–154, 2004

[5] A. K. Jain and B. Yu, "Automatic text localication in images and video frames", *Pattern Recognition*, 31(12), 1998.

[6] T. Sato, T. Kanade, E. Hughes, and M. Smith, "Video OCR for Digital News Archives," *IEEE Intl. Workshop on Content-Based Access of Image and Video Databases*, Jan, 1998.

[7] H. Li, D. Doermann and O. Kia. "Automatic Text Detection and Tracking in Digital Video". *IEEE Transactions on Image Processing*, 9(1):147–156, 2000.

[8] J. Xi, X. Hua, X. Chen, W. Liu, H.-J. Zhang. "A Video Text Detection and Recognition System". *IEEE International Conference on Multimedia and Expo* (*ICME* 2001), pp 1080-1083. 2001.

[9] Anonymous.

[10] J. Quinlan, Induction of decision trees. *Machine Learning,*1:81–106, 1986.

[11] Y. Freund and R. Schapire, "Experiments with a new boosting algorithm", *Proc. of the Thirteeth Int. Conf. on Machine Learning*, 148–156 (1996).

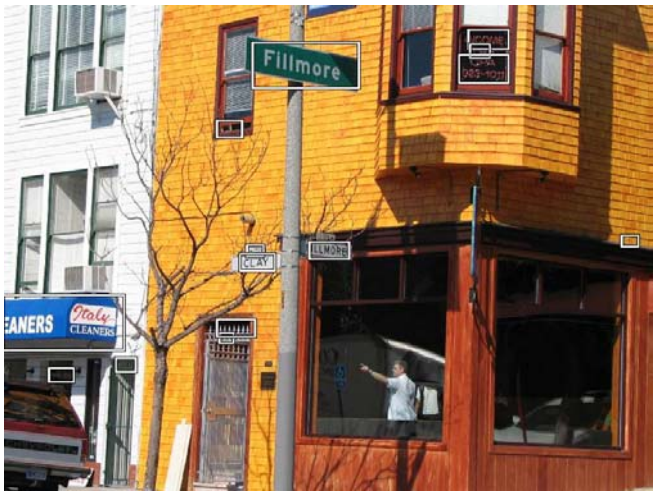
[12] P. Viola and M. Jones, "Fast and Robust Classification using Asymmetric AdaBoost and a Detector Cascade", In *Proc. of NIPS01*, 2001.

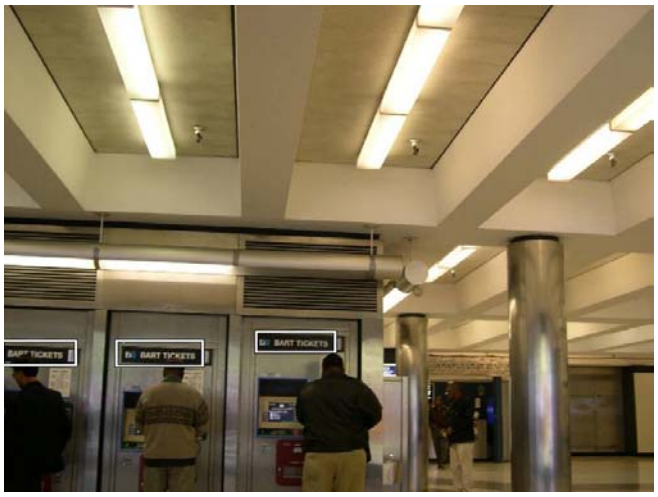
[13] G. Blanchard, D. Geman. "Hierarchical testing designs for pattern recognition." Preprints.

[14] L. Hyal and R. Rivest. Constructing optimal binary decision trees is NP-complete. Information Processing Letters, 35(1):15--17, 1976.

[15] V.N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

[16] S. M. Lucas, A. Panaretos, L. Sosa, A. Tang, S.Wong and R.Young. "ICDAR 2003 Robust Reading Competitions", In 7th International Conference on Document Analysis and Recognition- 2003.





Continue on next page

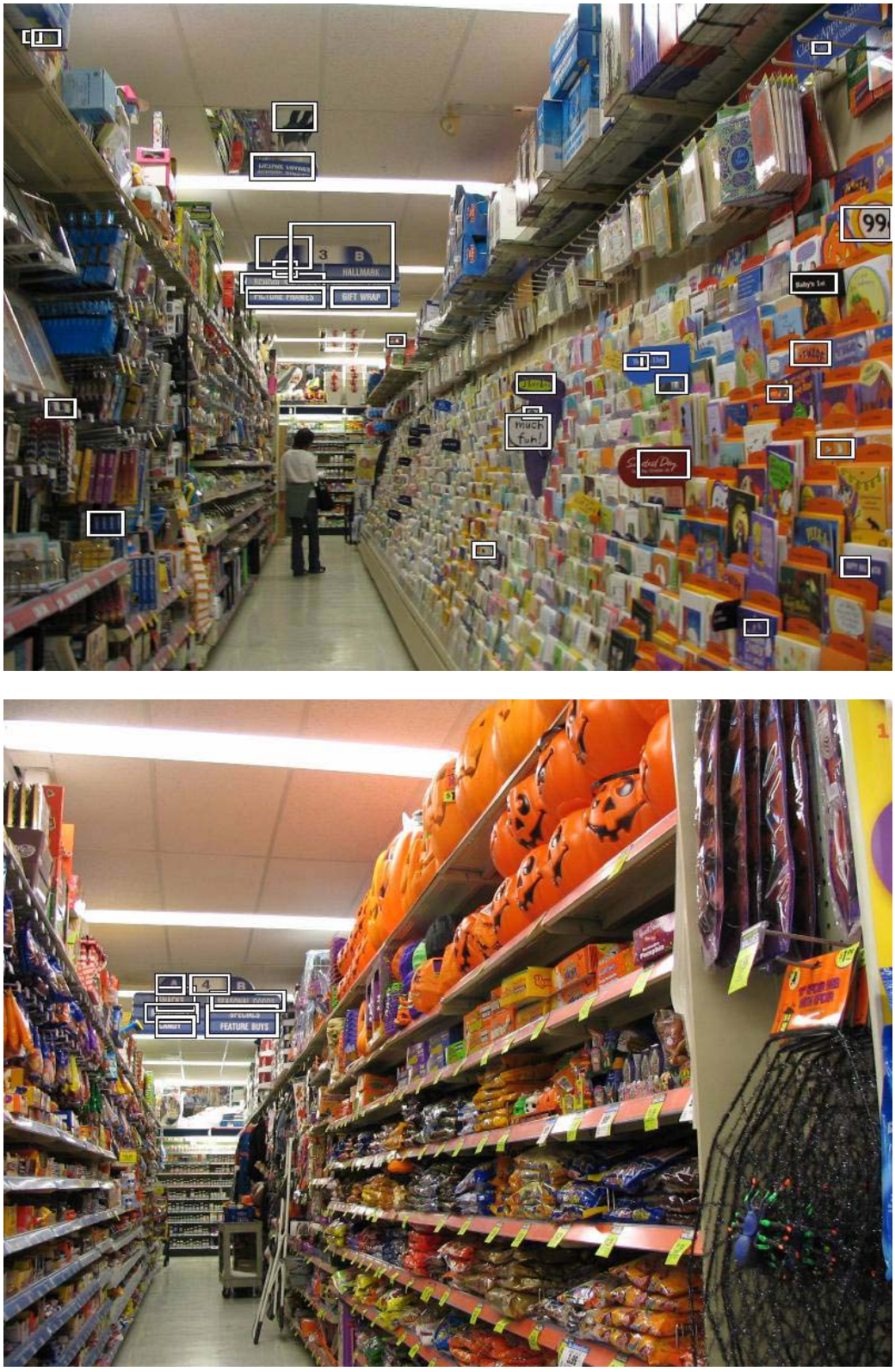


Figure 4. Text detection examples