An Unsupervised, Online Learning Framework for Moving Object Detection

Vinod Nair James J. Clark
Centre for Intelligent Machines
McGill University, Montreal, QC, Canada H3A 2A7
{vnair, clark}@cim.mcgill.ca

Abstract

Object detection with a learned classifier has been applied successfully to difficult tasks such as detecting faces and pedestrians. Systems using this approach usually learn the classifier offline with manually labeled training data. We present a framework that learns the classifier online with automatically labeled data for the specific case of detecting moving objects from video. Motion information is used to automatically label training examples collected directly from the live detection task video. An online learner based on the Winnow algorithm incrementally trains a task-specific classifier with these examples. Since learning occurs online and without manual help, it can continue in parallel with detection and adapt the classifier over time. The framework is demonstrated on a person detection task for an office corridor scene. In this task, we use background subtraction to automatically label training examples. After the initial manual effort of implementing the labeling method, the framework runs by itself on the scene video stream to gradually train an accurate detector.

1. Introduction

Visual object detection using a learned classifier has become popular in recent years. Working systems for detecting faces [11], [12], [16], handwritten characters [5], and pedestrians [10], [17], have convincingly demonstrated the approach's effectiveness. All these systems use a machine learning algorithm to train a classifier that can accurately distinguish tightly cropped images of the object of interest from all other images. The classifier is essentially a learned model of the object. This model can be used to detect an object instance appearing with arbitrary position and scale in an image by "scanning" the image with the classifier. Scanning is done by shifting a window of interest over the image and labeling each sub-image defined by the window as either 'object' or 'non-object' with the classifier. This is repeated at various window scales. The output of the detector is a bounding box for each object instance in the image.

Currently the popular method for training the image classifier is to use a supervised learning algorithm (e.g. AdaBoost [16], [17], neural networks [5], [12], support vector machines [10]) with a large, hand-labeled set of object and non-object images. This approach has two main drawbacks:
1) *Manual labeling*: Labeling a large training set by hand can be time-consuming and tedious. If the above detection strategy is to scale up well for problems that require a very large training set (more than a few thousand images), it is important to keep the human supervisory effort to a practical level. For example, when training a detector for use in a large-scale vision system, such as a video surveillance network with cameras deployed in a few hundred scenes, it would be highly undesirable to collect and label positive and negative training images from each scene by hand.
2) *Offline learning*: All learning occurs offline, i.e. before the classifier is used for the detection task. So after the initial training, the classifier remains fixed, and any further training "on the job" is not possible. However, in many detection tasks, incremental, online learning is desirable because then the classifier needs to know only what is actually necessary for the specific task. It also enables the classifier to be time-adaptive since online training can continue as long as the task is performed.

The main contribution of this paper is a detection framework that overcomes the above two limitations for the important special case of detecting moving objects from video. The basic idea is to design a substitute for manual labeling — an 'automatic labeler' — that uses motion information to supply labeled training examples (object and non-object images) directly from the video used for the detection task. Then the task video itself becomes the source for a stream of training examples that an online learner can use to incrementally train a classifier, as shown in figure 1. There is no need to strictly separate the training phase and the subsequent working phase as in the previous approaches.

Of course, the labeler itself is not a highly accurate classifier (otherwise there is no need to learn another one). It may fail often because of random sources of motion in

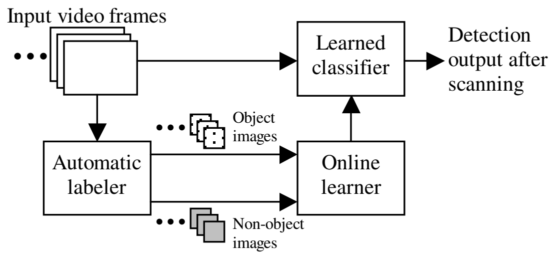


Figure 1. The proposed framework allows learning and detection to go on simultaneously.

the video. But if it can automatically recognize when the motion information is temporarily unreliable for collecting training data, then the labeling and the online learning can be stopped during such times. Therefore, accurate learning is possible even if the labeler fails often, as long as the training examples are labeled with reasonable accuracy when the labeler does work. We argue that it is possible to design such a labeler for moving object detection tasks based on existing techniques such as background subtraction. After the initial manual effort of designing the labeler, the system runs on its own and over time trains an accurate, task-specific detector.

The framework is described in more detail in section 3 using the example of a person detection task for an office corridor scene. Since the success of our approach largely depends on having an effective labeler, we discuss the general requirements for such a labeler in any detection task. We satisfy these requirements for the person detection task by designing a labeler based on background subtraction. For online learning we use Winnow [7], which learns a linear classifier in a Boolean vector space.

Each training image is mapped from its raw pixel form to a high-dimensional Boolean vector representation that is suitable for learning with Winnow. This representation is based on the type of overcomplete, local image feature set used by Viola and Jones [16]. We use these features because they enable effective learning and rapid detection. The expanded Boolean representation converts many nonlinear functions in the original pixel representation into linear ones, thus making them learnable with Winnow.

We take advantage of Winnow's ability to ignore irrelevant and redundant features to learn accurately in this large space. Training is made computationally efficient with the virtual weight algorithm [9], which allows Winnow to learn without explicitly computing a training image's high-dimensional Boolean vector representation.

2. Relation to existing approaches

Many approaches to moving object detection use motion as their primary source of information. An example is background subtraction [3], [4], [13], [15], which uses the difference between a model of the scene background and the current image of the scene to classify pixels as foreground or background. Connected foreground pixels are then grouped into blobs, and a blob-based object model is used for detection. This strategy of local motion measurement followed by a grouping operation to find moving objects is typical of motion-based methods, e.g. detection using optic flow. But key issues such as how to distinguish object motion from other sources of motion, and how to group local measurements, tend to be dealt with in an ad hoc manner that may not work well. As a result, in the case of background subtraction, detection can fail due to sudden background changes, slow-moving objects, shadows, and 'holes' in parts of the foreground that are similar to the background.

The strength of the learned classifier-based detection approach is that it selects the object model using a learning algorithm, based explicitly on the model's ability to discriminate between object and non-object training examples. Such a model is often better than a handcrafted, blob-based one at capturing the non-intuitive aspects of the object that can be important for high detection accuracy.

Our framework essentially combines the above two approaches to overcome the limitations of both. We use a hand-designed, motion-based detector to automatically collect labeled training examples that are then used to learn an accurate classifier. The work of Levin *et al.* [6] is similar to ours in that they apply motion information to reduce the hand-labeling effort. They initially train two classifiers — one on background difference images and the other on intensity images — with a small number of hand-labeled examples. Then the two classifiers use unlabeled data to iteratively improve each other. This algorithm differs from ours in three ways: 1) manual labeling is still necessary (albeit a small amount), 2) learning is done in an offline, batch manner, and 3) training is conceivably computationally expensive since two classifiers have to be repeatedly learned over multiple iterations. On the other hand, Winnow allows fast and efficient learning, as the results in section 5 show.

3. Unsupervised, online learning framework

In this section we further detail our framework with the example task of detecting people in an office corridor.

3.1. Task description

The corridor (see figure 2) is monitored with a network camera in fixed position. The detector's task is to

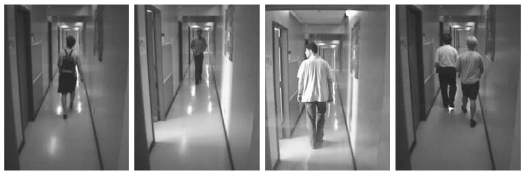


Figure 2. Example frames used for the detection task.

compute a bounding box for each fully visible person in a grayscale frame. Network and image compression latencies restrict the frame rate to about 1Hz. This task is challenging for simple motion-based detection methods. Large, abrupt lighting changes occur when an office door opens. People cast strong shadows on the walls. Detection fails when blobs of nearby people merge. Instantaneous motion is difficult to measure with optic flow because of the low frame rate.

3.2. Designing an automatic labeler

Now we explain how the automatic labeler is designed for this task. First, consider the two requirements that a labeler should satisfy:
1) *Automatic failure recognition*: The labeler must be able to automatically recognize when the motion information is not reliable enough to obtain correctly labeled training data. During such times, it can admit failure and temporarily stop. So it needs to supply accurate training data only when it can. By satisfying this condition, an effective labeler need not always be highly accurate.
2) *Unbiased labeling*: The labeler should not introduce any biases into the training data it provides that would mislead the learner. For example, if the labeler systematically fails to collect a certain type of training data, such a bias may cause the learner to develop misconceptions about the task and train an inaccurate classifier.

For the person detection task, we have hand-designed a simple, heuristic labeler based on background subtraction that approximately satisfies the above two requirements. A static, grayscale model of the background is initialized automatically by averaging ten consecutive frames that do not contain significant motion, as measured by the thresholded difference between two consecutive frames. The pixel-wise difference between the current frame and the background model is used to classify each pixel as either foreground or not. The connected foreground pixels are grouped into blobs. Blobs with approximately the correct aspect ratio and size are labeled as 'person.' Image regions that do not contain any foreground are labeled as 'non-person.'

Automatic failure recognition is handled differently for global and local background subtraction failures. When a large fraction of the frame persists as foreground, it is usually indicative of a gross background model failure due to, for example, a drastic change in scene lighting when an office door is suddenly opened. In such cases, both the automatic labeling and the online learning are stopped, and the background model is initialized again as before. Labeling and learning resume once reinitialization is complete.

Local failures typically occur when two nearby blobs merge, or when a person is partially segmented because part of the person and the background have similar intensities. The blobs resulting from these types of failures usually have the wrong aspect ratio and size. As a result, the local image regions containing blobs that do not satisfy the aspect ratio and size requirements are left unlabeled. No training data is obtained from such regions.

The overall accuracy of the labeler is poor because it fails more often than not. But it is still effective because the training data it provides the online learner is reasonably accurate. Figure 3 shows examples of automatically cropped person and non-person images. The labeler has processed the live task video over a period of five weeks to find 1557 person instances. However, almost 11% (168/1557) of these are actually mislabeled non-person instances. Some examples are shown in figure 4. The opposite error of labeling person instances incorrectly as non-person, while possible, has not occurred. Since learning occurs online, it is not practical for a human to inspect the live training data stream and manually remove the mislabeled examples. So the learning algorithm must be able to learn accurately despite the label noise.

The labeler satisfies the unbiased labeling requirement only partially because many of its failures are systematic rather than random. For example, it systematically fails to find people wearing clothes with similar intensity as the background. This bias may affect the learning. On the other hand, if the absolute intensity of people's clothes is irrelevant for learning the person model, then such a bias will not matter. Ultimately, the learned classifier's accuracy on an independent test set is the best measure for determining whether any apparent labeling bias really matters. As the results show (section 5), the classifier has good accuracy

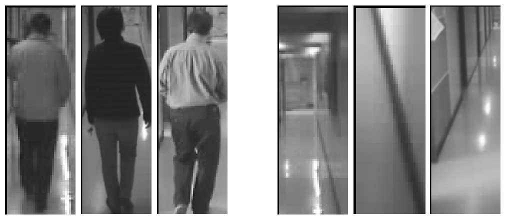


Figure 3. Examples of person and non-person images automatically cropped and correctly labeled by the labeler.

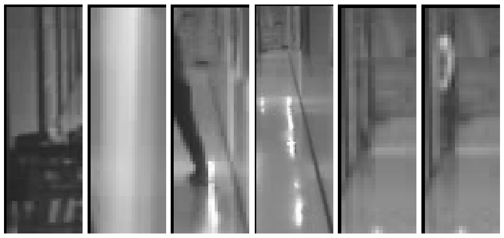


Figure 4. Examples of cropped images incorrectly labeled as 'person' by the labeler. Such errors occur when a partly segmented person, lighting changes, or shadows momentarily create a person-like blob.

and therefore, the labeler's biases do not significantly affect the learning.

3.3. Online learning

Here we give a high-level overview of the Winnow-based online learning. The details of Winnow, the local image features we use, and the virtual weight algorithm are given in section 4.

The main steps of the learning process are given in figure 5. For each frame, after the automatic labeling step, scanning is done by shifting a window of interest over it at various scales. For each subimage defined by this window at a particular location and scale, the current version of the Winnow-trained classifier is asked to classify it as either 'person' or 'non-person.' This is the *predicted* label. The label assigned to the subimage by the automatic labeler is called the *true label*. If the true label is available (it may be unavailable due to labeler failure), then it and the subimage together constitute a single training example for Winnow. Since Winnow is *conservative*, it learns from the example only if the predicted and true labels disagree.

Note that the learning is done "on the fly" while scanning, and there is no need to crop and store any training images as in the previous approaches. Also, finding representative negative training examples (referred to as "bootstrapping" [12]) is nicely integrated into this learning scheme. In each frame, the learner considers a large number of subimages as potential non-person training examples, but learns from only those that it does not yet know how to classify correctly. Thus it quickly learns to ignore uninformative non-person images.

Temporal adaptation of the classifier can be useful in this task. For example, people's clothing styles are significantly different in the summer (t-shirts, shorts) and the winter (bulky jackets), which affects their appearance. Instead of learning all possible appearances at once, the learner can simply adapt the classifier to them when necessary, thus making more efficient use of the learner's representational power. The practical benefits of such adaptation, as well as its possible transient effects on the accuracy of the classifier, need to be studied in detail.

```
Initialization:
1. Compute background model (section 3.2).
2. Initialize the state of the classifier (section 4.1).

Online learning algorithm:
For each frame
    Automatic labeling: (section 3.2)
    3. Perform background subtraction, group
       foreground pixels into blobs.
    4. Use the output from step 3 to classify all regions
       in the frame as 'person' or 'non-person', or as
       unlabeled.

    Learning: (sections 3.3 and 4)
    5. Scan the frame by shifting a window over
       the frame at various locations and scales.

    For each window location and scale while scanning
        6. Classify the subimage defined by the
           window using the current classifier state.
        7. If the subimage was labeled in step 4, then
           input the subimage, its 'true' label from step
           4, and its 'predicted' label from step 6 to
           Winnow for learning.
```

Figure 5. Summary of the unsupervised, online learning framework.

Some implementation details related to figure 5:

• The dimensions of the frames we use are 288x216. When scanning (step 5), the scan window's size is restricted to be a multiple of 48x16, and the allowed multiples (or scales) go from 1.00 to 5.75 in steps of 0.25. The window's vertical and horizontal shift step sizes are computed as 25% of its height and width, respectively.

• Non-person subimages in a frame outnumber the person subimages usually by a factor of $O(100)$. Such a highly imbalanced dataset is not necessarily a problem because Winnow learns only when the predicted and true labels disagree. So only the number of prediction mistakes per class matters, not the actual number of training examples per class. However, to reduce computation, we use only 50 non-person subimages (randomly selected) per frame.

• When to start using the classifier for detection is flexible. Although the classifier's output from scanning, computed in step 6, is available right from the start of training, it may not be accurate enough for detection purposes until after sufficient training. In practical applications, training may be considered sufficient after, say, the cumulative number of prediction mistakes reaches a predetermined value, or the rate of prediction mistakes becomes small.

4. Winnow, features, and virtual weights

This section describes the three main components of the online learner: 1) Winnow, 2) the local image features, and 3) the virtual weight algorithm.

4.1. Winnow

Winnow is a close relative of the perceptron that learns a two-class linear classifier for Boolean feature vectors. Like the perceptron, Winnow maintains one weight per feature, and classifies a feature vector based on the sign of the dot product of the weight vector and the feature vector. It uses a multiplicative update rule for changing the weights, which allows it to learn quickly in settings with only a few relevant features and many more irrelevant and redundant ones. It can tolerate the occasional mislabeled example because the incorrect updates caused by such errors can be reversed by the subsequent correct examples. For a more detailed description of the algorithm, see [7]. Examples of its practical success are [1], [11].

We use the balanced version of the Winnow algorithm [8], shown in figure 6. It maintains two weights per Boolean feature, one for the feature itself and the other for its complement. The weight update multiplier α is the only parameter to adjust. It controls how quickly the weights are promoted or demoted. We chose α to be 2 for computational efficiency, but other values can be used as well.

4.2. Local image features

Learning is performed on the type of local image features used by Viola and Jones [16], rather than on the raw pixels. An important advantage of these features is that they can be computed efficiently at any scale using an integral image. So, when scanning a frame with the classifier at various scales, the classifier itself can be 'resized,' just by scaling the features it uses. This makes scanning fast and efficient. Moreover, these features can support effective learning for object detection, as shown in [16], [17].

Figure 7 shows an example for each of the five types of features used in training. The numerical value of a feature is an integer, computed as the absolute difference between the pixel sums of the black and white regions. We use a total of 8294 features for all types together. So every subimage encountered in a scan (figure 5, step 5) can be thought of as an 8294-dimensional integer vector. Next, we explain how we map this vector to a high-dimensional Boolean vector space and use virtual weights with Winnow to learn efficiently in this space.

```
$N$-dimensional Boolean feature vector $x \in \{0,1\}^N$.
$N$-dimensional weight vectors $w^+, w^- \in \Re^N$.
Thresholds $\theta^+, \theta^- \in \Re$.
Weight update multiplier $\alpha > 1$.

Initialization: (fig. 5, step 2)
$w_i^+ \leftarrow 1, \theta^+ \leftarrow 1, w_i^- \leftarrow 1, \theta^- \leftarrow 1$ for all $i$.
Prediction: (fig. 5, step 6)
Predict 1 when $\theta^+ + \sum_{i=1}^{N} w_i^+ x_i > \theta^- + \sum_{i=1}^{N} w_i^- x_i$.
Update: (only for incorrect prediction; fig. 5, step 7)
    If true label is 0:
        $w_i^+ \leftarrow w_i^+ \alpha^{-x_i}, w_i^- \leftarrow w_i^- \alpha^{x_i}$ for all $i$.
        $\theta^+ \leftarrow \theta^+ / \alpha, \theta^- \leftarrow \theta^- \alpha$.
    If true label is 1:
        $w_i^+ \leftarrow w_i^+ \alpha^{x_i}, w_i^- \leftarrow w_i^- \alpha^{-x_i}$ for all $i$.
        $\theta^+ \leftarrow \theta^+ \alpha, \theta^- \leftarrow \theta^- / \alpha$.
```

Figure 6. Summary of the balanced version of the Winnow algorithm (adapted from Littlestone [8]).

4.3. Efficient learning with virtual weights

The virtual weight algorithm maps the integer vector to a Boolean vector representation in such a way that allows Winnow to learn without explicitly computing this representation. Therefore, even if the dimensionality of the Boolean vector is very high, learning is still efficient.

To understand how the mapping is done, first consider a simplified version of the problem with only one integer-valued feature (instead of 8294 of them). Let x denote the value of this feature. Suppose that x is delimited by the

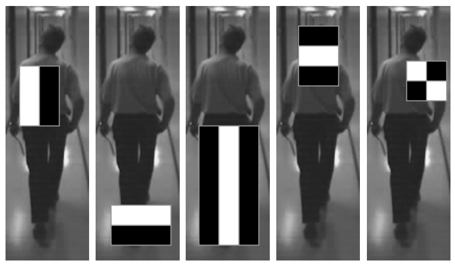


Figure 7. Examples of the five feature types we use. The position and size of the features with respect to the base window size of 48x16 are varied to define an overcomplete set of 8294 features.

interval [0, N], without losing generality. Suppose also that the true image classification function to be learned is linear with respect to terms of the form $x < t$ and $x >= t$, where t is an integer in [1, N]. (For example, a conjunction or a disjunction of such terms is a linear classification function, while an XOR of the terms is not.) One way to learn this function with Winnow is to define a Boolean feature of the form $x < t$ for each possible threshold t. Such a feature's value is 1 if $x < t$, and 0 otherwise. Then learning can be performed on these features and their complements. A naive implementation of this idea is to use two weights per feature with balanced Winnow. However, this requires maintaining $2N$ weights (one weight-pair per threshold), which can be computationally expensive if N is very large.

A more efficient approach is to maintain weights virtually by storing one weight for an entire interval of thresholds that has the same weight. For example, the balanced Winnow algorithm in figure 6 starts off with all the weights set to 1. So, in the simplified version of the problem, the Boolean features with a threshold in the interval [1, N] all have a weight of 1 initially. Instead of storing N identical weight-pairs to represent this situation, the virtual threshold algorithm stores only one weight-pair and the two limits of its associated threshold interval.

Now suppose that a training example with the local image feature value $x = a$ is given and Winnow predicts its label incorrectly. Then in the "Update" step in figure 6, a Boolean feature's weight is updated differently depending on whether its threshold is above or below a. Therefore, instead of storing a single weight-pair for the interval [1, N], now two weight-pairs have to be stored, one for the interval [1, a) and the other for [a, N]. So in the worst case, the number of weight-pairs increases only with the number of prediction mistakes. When the number of mistakes is much smaller than N, the savings in memory and computation are significant compared to explicitly storing and updating all the weights.

In the learning framework, we apply the same weight representation technique described for the simple case above to each of the 8294 local image features simultaneously. Training a linear image classifier on Boolean features of the form $x < t$ (and their complements) is a reasonable thing to do because these features are the weak classifiers used successfully by Viola and Jones in [16]. They show that a linear combination of such features (computed by AdaBoost) can be used to build an accurate classifier for face detection.

Note that the Boolean vector representation of a training image never needs to be computed explicitly, which is advantageous because it has an extremely high dimensionality. We maintain weights virtually for 222,955,814 Boolean features and their complements, so the feature vector has a total of 445,911,628 dimensions. However, in our experiments, the number of actual weights maintained during training reaches a maximum of only about 1.7 million. Intuitively, one expects the curse of dimensionality to make learning very difficult in this space. But Winnow's ability to quickly ignore redundant and irrelevant features allows it to learn accurately and overcome the curse. The same counter-intuitive strategy of learning a linear function in a very high dimensional space is used successfully by support vector machines [2] and tile coding-based linear function approximation in reinforcement learning [14].

During training, many of the weights become so small compared to the largest weight that their effect on the output of the classifier becomes negligible. So we prune those weights and their associated Boolean features, just as in [1]. This results in large savings in computation and memory. Pruning essentially integrates feature selection into the learning process, by throwing away features that do not help Winnow learn an accurate classifier. In our experiments, we prune Boolean features with a weight that is 2^{20} times smaller than the current largest weight. This threshold has to be chosen carefully: if it is too small, aggressive pruning may remove many useful features. If it is too high, a large number of low-weight features may slow down training significantly.

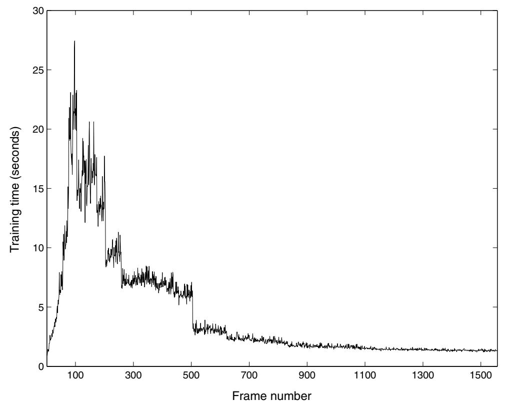
5. Person detection results

5.1. Training

To train the person detector, the algorithm in figure 5 is run on the task video stream for a period of five weeks. During this period, a set of frames containing 1557 person examples and 77850 non-person examples is used for learning (these examples contain the label errors mentioned in section 3.2). Figure 8 plots the learner's time and memory use on a Pentium 4 2.5Ghz PC. The graphs show that the computational requirements of the framework remain reasonable throughout training. Learning is slow early on when a large number of weights are maintained, but it becomes faster as useless Boolean features are pruned away. In the end, Boolean features computed from just 529 out of the initial 8294 local image features remain.

5.2. Testing

We measure the accuracy of the trained detector on a set of manually collected and labeled test frames (i.e., frames not used for training) containing 2890 person instances and 852,001 non-person instances. Figure 9 shows the receiver operating characteristic (ROC) of the detector on this test set. Each frame is scanned the same way as described in section 3.3, except with an initial scale of 2.00. This speeds up the detection significantly for a negligible loss



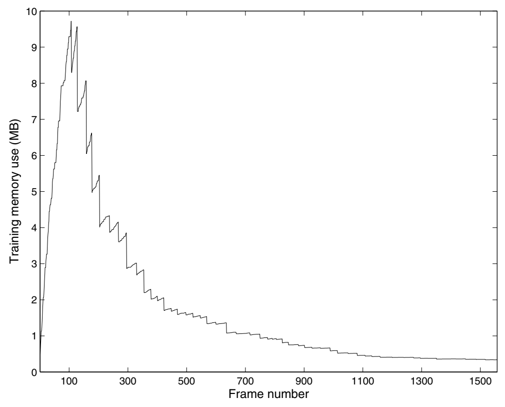


Figure 8. Time and space requirements of the online learning algorithm as training progresses on a Pentium 4 2.5Ghz PC. The number of weights increases rapidly early on when the learner makes a lot of mistakes. Then it starts decreasing as the weights of many features become small and pruning removes those features. Eventually, the number of features stabilizes.

in accuracy. A 288x216 frame contains 785 windows, and is scanned in approximately 1.6 seconds. Note that a person instance is almost always detected more than once, but many false positives are isolated single detections. So single detections are ignored, and multiple detections are combined by averaging significantly overlapping detection windows. Figure 10 shows examples of test frames scanned using the detector.

Although the test results show that reasonable accuracy can be achieved even with an extremely simple labeler, the detector's performance is not entirely satisfactory. For example, at a detection rate of 80%, its false positive rate is 0.2%, or 1/500 (about 3 false positives per two frames). In comparison, Viola *et al.* [17] report a false positive rate of 1/15000 at the same detection rate for a pedestrian detector trained on appearance information with AdaBoost and a hand-labeled dataset. (Since the two detection tasks are not identical, this is only a rough comparison.)

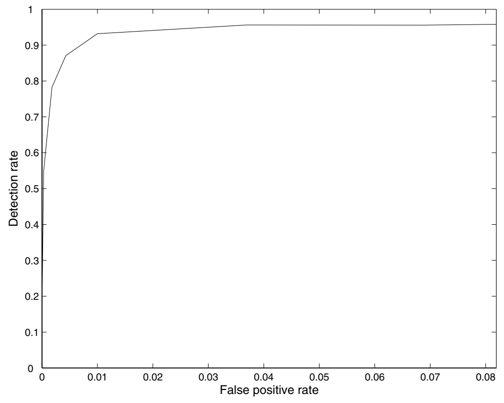


Figure 9. Receiver operating characteristic of the detector on the test set.

The 30-fold higher false positive rate of our detector can be attributed mainly to the large amount of false positive noise generated by the automatic labeler. This problem can be overcome to a significant extent by designing a more accurate labeler. We use a very simple labeler here (in retrospect, perhaps too simple), and it can be improved with a better background subtraction algorithm. With a sharp reduction in the false positive training examples, the accuracy of the detector should improve significantly.

Nevertheless, in many practical applications it may be difficult to design an automatic substitute for hand-labeling that never generates any label noise. As a result, it would not be surprising if a detector trained under our framework is not as accurate as one trained on manually-labeled images with a batch learning algorithm. The loss in accuracy is the price to pay for eliminating hand-labeling. However, we anticipate that for many applications this price will be small compared to the savings in human effort.

6. Conclusions

We have proposed a new detection framework for moving objects based on a learned classifier. It has two main benefits: 1) manual labeling is completely avoided by using motion information to collect labeled training images, and 2) training is performed online, so the classifier needs to learn only what is required specifically for the task, and only when necessary. By making the classifier scene-specific and temporally adaptive, its representational capability is used more efficiently. This is in contrast to the previous approaches that train offline a fixed, scene-independent classifier with a large, hand-labeled dataset. We have also presented an efficient method that enables Winnow-based online learning on the kind of features used by Viola and Jones [16]. The viability of the framework is demonstrated with an example person detection task.

This framework is particularly useful for applications such as wide-area video surveillance systems, which use hundreds of cameras in many different scenes. Instead of training one general-purpose detector for all possible scenes with a large amount of hand-labeled data, each camera can have its own specialized detector automatically trained for its limited operating environment.

References

[1] A. Blum. Empirical Support for Winnow and Weighted-Majority Based Algorithms: Results on a Calendar Scheduling Domain. *Machine Learning* 26, pages 5-23, 1997.

[2] C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pages 121-167, 1998.

[3] A. Elgammal, D. Harwood, and L. Davis. Non-parametric Model for Background Subtraction. In *European Conference on Computer Vision*, 2000.

[4] N. Friedman and S. Russell. Image Segmentation in Video Sequences: A Probabilistic Approach. *Proc. of Conf. on Uncertainty in Artificial Intelligence*, pages 175-181, 1997.

[5] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-Based Learning Applied to Document Recognition. *Proc. of the IEEE*, vol. 86, no. 11, pages 2278-2324, 1998.

[6] A. Levin, P. Viola, and Y. Freund. Unsupervised Improvement of Visual Detectors Using Co-training. *Proc. of ICCV*, pages 626-633, 2003.

[7] N. Littlestone. Learning Quickly When Irrelevant Attributes Abound: A New Linear-threshold Algorithm. *Machine Learning* 2, pages 285-318, 1987.

[8] N. Littlestone. Comparing Several Linear-threshold Learning Algorithms on Tasks Involving Superfluous Attributes. *Proc. of Int. Conf. on Machine Learning*, pages 353-361, 1995.

[9] W. Maass and M. Warmuth. Efficient Learning with Virtual Threshold Gates. *Information and Computation*, 141(1), pages 66-83, 1998.

[10] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, and T. Poggio. Pedestrian Detection Using Wavelet Templates. *Proc. of Computer Vision and Pattern Recognition*, pages 193-199, 1997.

[11] D. Roth, M. Yang, and N. Ahuja. A SNoW-based Face Detector. In *Neural Information Processing Systems 12*, 2000.

[12] H. Rowley, S. Baluja, and T. Kanade. Neural Network-based Face Detection. *IEEE Trans. on PAMI*, vol. 20, no. 1, pages 23-38, 1998.

[13] C. Stauffer and E. Grimson. Adaptive Background Mixture Models for Real-time Tracking. *Proc. of Computer Vision and Pattern Recognition*, pages 246-252, 1999.

[14] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[15] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: Principles and Practice of Background Subtraction. *Proc. of ICCV*, pages 255-261, 1999.

[16] P. Viola and M. Jones. Robust Real-time Object Detection. *Second Int. Workshop on Statistical and Computational Theories of Vision*, 2001.

[17] P. Viola, M. Jones, and D. Snow. Detecting Pedestrians Using Patterns of Motion and Appearance. *Proc. of ICCV*, pages 734-741, 2003.

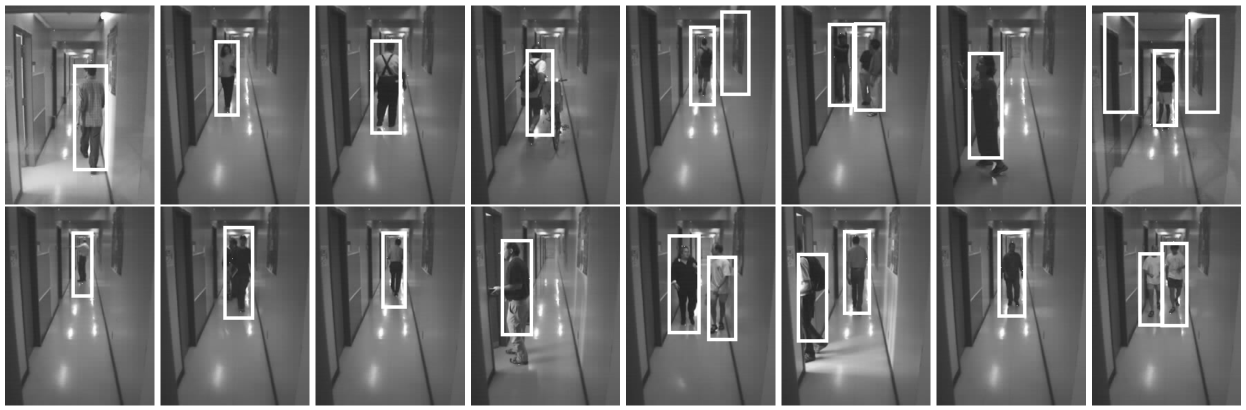


Figure 10. Examples of test frames scanned with the detector.