Appears in the *Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pp 231–238, Beijing, China 2005.

Using SVM for Efficient Detection of Human Motion

Josef Grahn

School of Computer Science and Communication
KTH (Royal Institute of Technology)
SE–100 44 Stockholm, Sweden
grahn@kth.se

Hedvig Kjellström*

Department of IR Systems
Division of Sensor Technology
Swedish Defence Research Agency
SE–164 90 Stockholm, Sweden
hedvig@foi.se

Abstract

This paper presents a method for detection of humans in video. Detection is here formulated as the problem of classifying the image patterns in a range of windows of different size in a video frame as "human" or "non-human". Computational efficiency is of core importance, which leads us to utilize fast methods for image preprocessing and classification. Linear spatio-temporal difference filters are used to represent motion information in the image. Patterns of spatio-temporal pixel difference is classified using SVM, a classification method proven efficient for problems with high dimensionality and highly non-linear feature spaces. Furthermore, a cascade architecture is employed, to make use of the fact that most windows are easy to classify, while a few are difficult. The detection method shows promising results when tested on images from street scenes with humans of varying sizes and clothing.

1. Introduction

Fast and robust detection of humans in video is made increasingly possible by the recent development in computational power. This technology is interesting for, e.g., automotive applications and automation of visual surveillance.

Our aim in this paper is a method for detection of humans in video, with a low error rate, using standard hardware and software. Furthermore, the detection might be used to initialize a human tracker, which means that the computations must be done in the time interval between two video frames. Surveillance video is usually recorded at a rate of between 10 and 30 Hz.

Detection of humans in a video frame can be reformulated as the problem of classifying a range of windows of different position and scale in the frame as "human" or "non-human". Several thousand windows need to be classified in each frame. Figure 1 shows an example of a detection result.

*Formerly Hedvig Sidenbladh

Figure 1: The result of a detection. The three windows classified as "human" (of about 15000 windows investigated) are marked with rectangles in the image.

To manage this in real-time, three aspects must be taken into regard:

The computational efficiency of the image preprocessing. We employ a combination of spatio-temporal difference images, suggested by Viola et al. [16]. Due to their linearity, these filter responses can be computed in a very efficient manner.

The computational efficiency of the classifier. Since so many windows are classified, the classification must be extremely fast. Furthermore, the patterns to be classified (i.e., the filter response in the windows) are high-dimensional, and the part of feature space in which "human" patterns reside is of non-linear shape. We have chosen to employ support vector machines (SVM) [1, 4], proven efficient for this type of classification problem. Another option, successfully employed by Viola et al. [16], is AdaBoost.

The fact that most windows are easy to classify, while a few are difficult to classify. To make use of this, we em-

ploy a cascade architecture [15]. A very fast but error-prone linear SVM is used to classify all the thousands of windows in the first step, and feed a smaller number of "possible human" classifications forward to the next step, and so forth. In the last step, a slower SVM with high accuracy takes care of the few difficult cases.

The SVM classifiers require large amounts of training data. Accommodating to this, an interactive reinforcement learning procedure was implemented, in which a user marks detections as correct or not correct. This information is then fed into the SVM, which is retrained with the new data. This procedure allows for semi-automatic collection of examples, a tiresome task if performed manually.

The rest of the paper is organized as follows. In Section 2, related work is presented. The detection framework is formalized in Section 3, followed by a description of the classification method in Section 4. The training of the SVM is treated in Section 5. Thereafter, experimental results are presented in Section 6. The paper is concluded with a discussion and suggestions for future work.

2. Related Work

Most methods for human detection aim at detecting human appearance. Cues used are edges [7, 9], wavelet responses [11], color distributions [3], background subtraction [8] or a combination of multiple cues such as depth information, color and neural-net models of face patterns [5]. The detection is often used as an initialization step to tracking [3, 5, 8]. However, if it is possible to estimate image motion (which is most probably not the case if the camera is mounted on a car [7], for instance), this is a very powerful cue to human presence in the image.

An approach based on motion is presented by Song et al. [14]. Here, feature points from two consecutive images in a sequence are compared to the corresponding points on a 2D kinematic model of a human. This approach does not entirely rely on motion information since there is an underlying assumption that one can find features corresponding to specific positions on the body. Viola et al. [16] use instead a filter-based approach to motion pattern recogntion. Using five filters for motion in different directions they are able to detect walking humans in low image resolution with a very low error rate. We use the spatio-temporal filters of Viola et al. in our detection, but SVM for classifying the filter responses, instead of AdaBoost as they do. Optical flow is another representation of image motion. Sidenbladh [13] uses SVM and optical flow patterns for detection. The model-based method of Fablet and Black [6] instead compares dense flow patterns with a generative model of human flow appearance. The method recovers both pose, orientation and position in the image.

SVM has proven to be an effective method for non-linear classification in high-dimensional spaces such as ours. The method has previously been used for face detection [12] and gender classification [10]. Pedestrian detection using SVM from appearance cues such as edge segments in the image [9], optical flow [13] or wavelet responses [11] has also been reported.

3. Detection

Informally, our method is supposed to find rectangular areas, windows, of different sizes containing humans in an image from a sequence. This can be expressed more formally as follows.

3.1. Formalization of the Detection Problem

The input to the method is a number of consecutive frames from an image sequence of images. Let the variable A_t denote such a set of frames, which we henceforth will call a *snapshot*. For our purposes, the snapshot A_t consists of the two frames at time t and $t+1$, but it could well be a longer sequence of images.

If we assume a fixed aspect of the image windows to be tested, any window can be described using three parameters: horizontal position x, vertical position y and scale s. We then have a space K containing all candidates (x, y, s) that may mark a human.

The system can thus be seen as to, given a snapshot A_t, produce an indicator function $g : K \rightarrow \pm 1$, which is an approximation of some true function g^T. The system is hence in itself a function $f(A_t)$ with the set of all possible functions $g(x, y, s)$ as its range. In other words, $f(A_t)$ is the function that generates positions and scales for all detected humans in the snapshot A_t, while g is a function that returns 1 if there is a human at position (x, y, s), -1 otherwise.

3.2. Requirements

The problem has now become to find a suitable function f. A good f should have the following properties.

1. The average time it takes to evaluate f should be low.
2. The average time it takes to evaluate g should be low.
3. The probability for a false positive classification should be low. That is, $P(g(x, y, s) = 1 \,|\, g^T(x, y, s) = -1)$ should be low.
4. The probability for a false negative classification should be low. That is, $P(g(x, y, s) = -1 \,|\, g^T(x, y, s) = 1)$ should be low.

The exact probabilities in the third and fourth requirement are determined by the application. Generally, the false positive and false negative ratio are inversely correlated. If

the system is otherwise tuned to its best accuracy, modifying it to be more restrictive towards signaling a hit, will produce fewer false alarms, but will make it miss more true occurrences.

3.3. Cascade Detection

The first and second requirement above may seem redundant. However, the execution time of g varies significantly depending on the complexity of the classifier used. Simple, e.g. linear, classifiers take less time, but display a higher false positive ratio than more complex classifiers.

A cascade architecture [15] is therefore utilized to classify samples. Several classifiers with increasing complexity constitutes the cascade, where each stage classifies the samples as to be either "not human" or "possibly human". If a sample is determined not to contain a human being, it is thrown out immediately, saving valuable execution time for other samples. Samples that pass through all stages are considered to contain humans.

This design exploits the fact that the vast majority of samples are negative, and that most of these can easily be determined to be that. Thus, the cost of classifying the fewer difficult samples is amortized over the entire set of samples.

4. Classification

We now treat the classification of a particular window (x, y, s) in the snapshot A_t. This is equivalent to the function g as defined in Section 3.1.

4.1. Image Preprocessing

Firstly, to achieve scale invariance, Laplacian pyramids are constructed from both frames in the snapshot A_t. The scale parameter s indicates from which pyramid level the image window (x, y, s) should be extracted.

From now on, the following terminology will be used [16]. Let I_t be the image window in position and scale (x, y, s) at frame t. The same windows translated slightly to the right, to the left, up, and down are then denoted $I_t^{\rightarrow}$, $I_t^{\leftarrow}$, $I_t^{\uparrow}$, and $I_t^{\downarrow}$, respectively.

Spatiotemporal difference filters can now be defined from these images as

$$F_t = |I_t - I_{t+1}| \,, \tag{1}$$
$$F_t^{\uparrow} = |I_t - I_{t+1}^{\uparrow}| \,, \tag{2}$$
$$F_t^{\downarrow} = |I_t - I_{t+1}^{\downarrow}| \,, \tag{3}$$
$$F_t^{\leftarrow} = |I_t - I_{t+1}^{\leftarrow}| \,, \tag{4}$$
$$F_t^{\rightarrow} = |I_t - I_{t+1}^{\rightarrow}| \,. \tag{5}$$

These filters contain information about the motion in the image. An alternative representation could be optical flow. Flow would represent the motion in a more compact way in terms of number of dimensions in the feature space, leading to an easier classification problem [13]. However, besides being more computationally demanding, the optical flow computation may also introduce errors which would inflict on the classification [16].

The function g could be defined from any combination G_t of these filters. Here, we investigate two combinations.

Difference image. The simplest combination filter is the temporal difference only, i.e.,

$$G_t = F_t \,. \tag{6}$$

Thus, the resulting image will contain information about how much the intensity has changed between the two frames. Examples of responses from this filter are shown in Figure 2a.

The drawback of this combination is that it does not contain very much information. On the positive side, an SVM classifier using this filter will be fast, due to the low number of dimensions in the feature space (which is the window height times the window width). This filter is thus suitable for the first steps in the classification cascade.

Integral difference image. Viola et al. [16] and have successfully used a combination of all difference filters in Equations (1)-(5), which is inspired by the *integral image* of Viola and Jones [15]. Connecting to this terminology, we call this combination,

$$G_t = [F_t, F_t^{\uparrow}, F_t^{\downarrow}, F_t^{\leftarrow}, F_t^{\rightarrow}] \,, \tag{7}$$

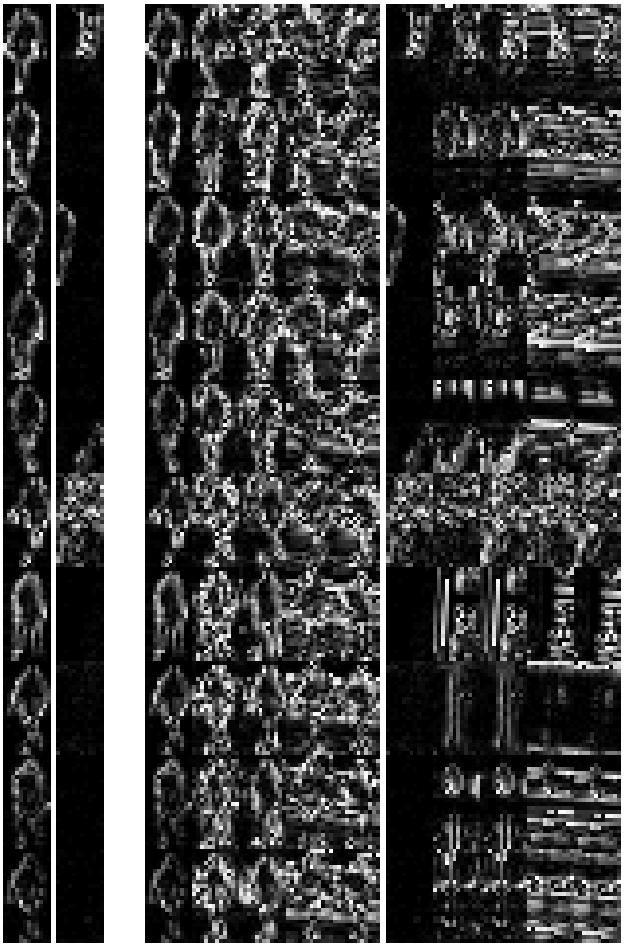
integral difference image, which besides information about intensity change contains information about motion in four directions. Figure 2b shows responses from this filter.

A disadvantage of this filter is that it has many sampling points, hence many dimensions in feature space. This will make the SVM classification more computationally expensive. However, since the filter is a more informative description of the image motion, the classification will be more accurate than with the difference filter. Thus, the integral difference filter is suitable for the last stages in the classification cascade.

Filter responses as points in feature space. Henceforth, the filter responses G_t are considered to be on vector form; in practice, all columns of all difference images in the filter response are concatenated to one vector. Furthermore, the t subscript is dropped for notational purposes.

4.2. Support Vector Machines

As discussed in Section 3.1, our binary classifier can be expressed as a function $g : \Re^n \rightarrow \pm 1$ that maps patterns G



(a) Difference (b) Integral difference

Figure 2: Filter responses. For each type of filter, the left column contains positive examples and the right negative examples.

onto their correct classification z as $z = g(G)$. In the case of an SVM, the function g takes the form [1, 4]

$$g(G) = \sum_{i=1}^{N} z_i \alpha_i k(G, G_i) + b \,, \tag{8}$$

where N is the number of training patterns, (G_i, z_i) is training pattern i with its classification, α_i and b are learned weights, and $k(.,.)$ is a kernel function. Here, we use both linear functions $k(G, G_i) = G - G_i$ and radial basis functions (RBF) $k(G, G_i) = e^{-\|G-G_i\|/2\sigma^2}$. The patterns for which $\alpha_i > 0$ are denoted *support vectors*.

The surface $g(G) = 0$ defines a hyperplane through the feature space as defined by the kernel $k(.,.)$. The weights α_i and b are selected so that the number of incorrect classifications in the training set is minimized, while the distances from this hyperplane to the support vectors are maximized. This is achieved by solving the optimization problem [1, 4]

$$\textit{Maximize}: \quad L_D \equiv \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} z_i z_j \alpha_i \alpha_j k(G_i, G_j) \tag{9}$$

$$\textit{subject to}: \quad 0 \leq \alpha_i \leq C \,, \quad \sum_{i=1}^{N} z_i \alpha_i = 0 \,. \tag{10}$$

The constant C affects the tolerance to incorrect classifications. Using the optimal parameters α_i, Eq (8) with any support vector (x_i, y_i) as indata can be used to find b.

For a thorough description of the SVM theory, see [1, 4].

SVM parameters. One of the benefits of the SVM, compared to many other inductive learners, is that few parameters must be set manually. Consequently, the parameter space that must be explored to find the optimal set-up is relatively small.

The first parameter is the choice of kernel. Secondly, kernel specific parameters appear. The RBF kernel requires the stiffness σ to be set. Apart from kernel specific parameters, there are the penalties for incorrect classifications of positives C_p and negatives C_n, respectively.

SVM complexity. The computational complexity of an SVM classification step depends on the type of kernel. For a linear kernel, the complexity is $\mathcal{O}(\dim(G))$. It is therefore desirable from an computational efficiency point of view to keep the number of dimensions in the patterns G as low as possible. Classification with a linear kernel is extremely fast but inexact, therefore suitable for early stages in the cascade.

If an RBF kernel is used, the complexity rises to $\mathcal{O}(N_i \dim(G))$ where N_i is the number of support vectors. N_i depends on $\dim(G)$ as well as on the stiffness and penalty paramters in the SVM. To achieve a fast classification, one should strive to keep N_i as low as possible. The RBF kernel requires more computation time than a linear kernel but renders a more exact classification. It is therefore suitable for the final cascade step.

It is important to note that the error rate of the classification, which also depends on $\dim(G)$ and N_i, is of importance as well as speed (Section 3.2). Tradeoffs between computational efficiency and classification accuracy have to be made, both with the choice of kernel and when setting other SVM parameters.

5. Training the SVM

As noted in [13], a large number of training examples are required for an SVM to be able to classify human motion patterns. To allow for efficient gathering of examples, a tool for interactive learning was implemented.

5.1. Acquisition of Training Examples

Obtaining a profusion of video recordings containing humans is a trivial task, having access to a large city and a digital video recorder. The problem is instead to mark the humans in the video material.

Extracting negative examples requires much less effort, although the process cannot necessarily be completely automated. After all, we are interested in negative examples from the environment where we later on intend to use the system to detect positive occurrences, whence we should prepare for the risk of these occurring while we collect the negatives. But still, only a very small amount of human supervision is needed.

Interactive marking tool. To accommodate the goal, an interactive marking tool was constructed. Using this tool, an operator can step through and view a recorded image sequence. Using the mouse, markers can be placed where a training examples should be extracted.

Classifier feedback. Once an initial collection of examples have been obtained, it is possible to train a SVM which can be used to extract new examples, which are presented to the user for rejection or confirmation.

The procedure has two advantages. First, it quickens the process of collecting positive examples. Secondly, by collecting the samples that are misclassified by the SVM, one can be certain to obtain examples that are useful from a learning point of view. While it is easy to gather abundant negative examples randomly, few of them give the SVM trainer additional knowledge of the shape of the set [12].

Overlearning is of course the great danger with this sort of feedback procedure.

5.2. Training and Test Data

The video material used in the experiments consist of three 30 second 25 Hz sequences recorded in the daytime at three different locations. The camera is located on street level, approximately one metre above ground. The sequences include both distant humans as well as humans right in front of the camera. In many cases, the humans walk towards or away from the camera.

The sequences are split in half, where the first half is used for training and the second for testing. The same person is never present in both test and training sequences.

From the training sequences, 1005 positive and 1434 negative examples are extracted and used to train the SVM.

5.3. Parameter Optimization

During training, the SVM parameter space is explored by a grid search. Initially, a sparse grid over a vast range of values is used. The limits of this grid are chosen based on theoretical arguments; for example, the limits of the kernel

Figure 3: The left image shows all samples considered positive by the stages of the cascade. The right images shows the same image, but with overlapping samples merged.

stiffness are imposed by the high frequency camera noise on one hand, and the extension of the set of all samples on the other. In other words, the kernel needs to be stiff enough, so that the radial basis is significantly larger than the size of typical perturbations induced by the camera noise. The basis should also be small enough not to eclipse the entire set of samples with a single RBF. Promising areas are thereafter investigated further, using a finer grid.

The number of support vectors varies between 500 and 1000 depending on the parameters, with approximately the same number of positive and negative support vectors.

6. Experiments

In this section, experimental results are presented. Different parameter settings, kernels and preprocessing settings are compared in terms of the false positive and false negative ratio. Computation time for the different steps of a cascade is also measured.

6.1. Notes about Measuring Error

It should be noted that, when negative training examples are collected using classifier feedback as decribed in Section 5.1, the resulting set of examples will be harder to classify than an average example generated uniformly over K. For training purposes, this is desired since it reduces the training time by focusing on important examples. However, if these examples were used for validation, they would yield unfair results. To get more accurate results, the system is validated exclusively on randomly generated negative examples.

Furthermore, as a postprocessing step in each level of the cascade, detections with significant overlap are merged

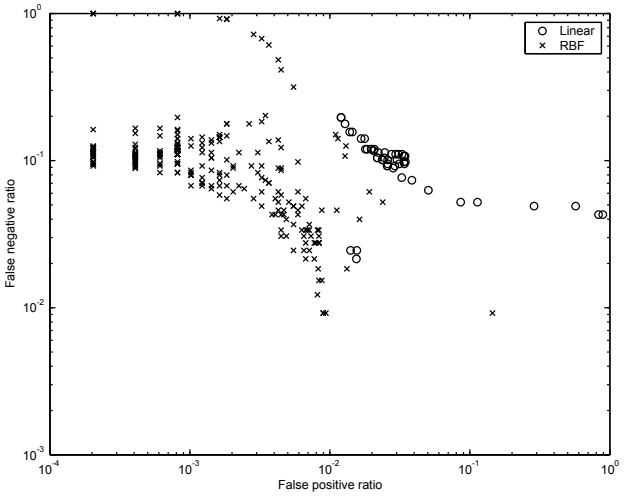


Figure 4: False negative and false positive rate of a range of parameter configurations. Different markings represent different kernels.

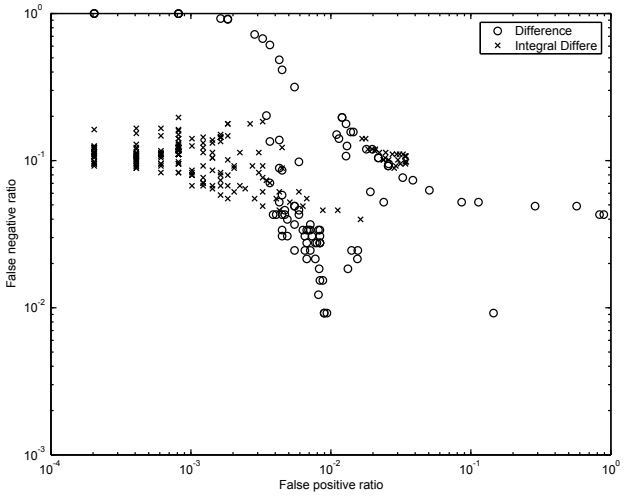


Figure 5: False negative and false positive rate of a range of parameter configurations. Different markings represent different filters.

(Figure 3). During the course of a cascade, 8% of the detections are removed by merging.

6.2. Comparing Kernels

The performance of the linear and RBF kernels are first compared. Figure 4 shows the false positive and false negative ratio, using a number of parameter configurations, filters and filter sizes for each kernel. The shape of the different markers indicate which kernel that was used.[1]

Examining the plot, it is obvious that the RBF kernel is the superior kernel, with regard to accuracy. However, it is more computationally expensive than the linear kernel.

6.3. Comparing Filters

The two filter combinations suggested in Section 4.1 are then compared. Figure 5 presents the different error ratios achieved for different parameter configurations, kernels and filter sizes. Different filters have different markers.

Combining the information in Figures 4 and 5 shows that the lowest false negative ratios are reached using the combination difference filter/RBF kernel, while the lowest false positive ratios are reached with integral difference filter/RBF kernel. The same false negative ratio can be reached with the combination difference filter/linear lernel, which is fast, as with integral difference filter/RBF kernel, which is more computationally demanding.

The conclusions that can be drawn from this is that the difficult cases in the late stages of the cascade should be classified using the combination integral difference filter/RBF kernel which results in very low false negative ratios. However, in the early stages, a difference filter should be used since the classification is faster, and results in the same amount of, or even fewer, missed detections.

[1] Since all parameters are varied, the plot can be percieved as a "ROC cloud", in analog to a ROC curve where one parameter is varied.

6.4. Comparing Filter Resolutions

Figure 6 shows error ratios achieved using two different filter resolutions, 4×8 and 12×24 pixels respectively. For each size, results for different parameter configurations, filters and kernels are shown. Different filter sizes have different markers.

It is obvious from the results that the lower resolution performs worse in terms of error rate. Again, a tradeoff has to be made between performance and computational cost.

6.5. Typical Misclassifications

Incorrect classifications can roughly be divided into four categories, which are described below. It should be noted that the occurrence of misclassification can be limited considerably with interactive learning as described in Section 5.1. Furthermore, the images below should not be seen as representative of the detection method. An example of a successful detection can be seen in Figure 1.

False positives around humans. The invariably most common misclassifications are incorrect positives produced in the vicinity of a correctly classified single human. These exist both as slightly translated and possibly upscaled markers, covering most of the person plus some of the background, or, more frequently, as downscaled markers covering moving body parts. When the pedestrian is close to

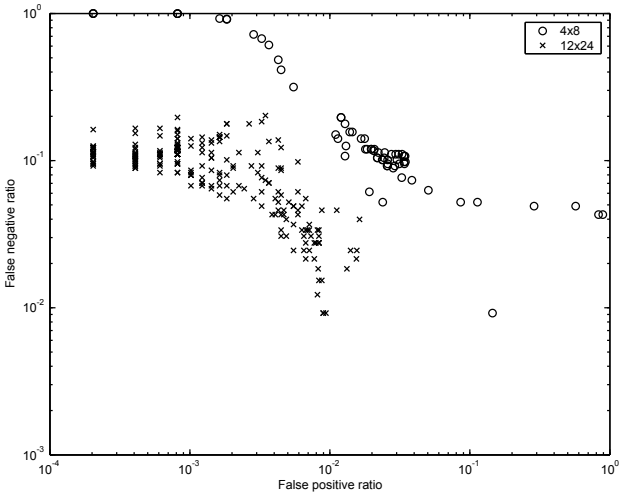


Figure 6: False negative and false positive rate of a range of parameter configurations. Different markings represent different filter resolutions.

the camera, a single moving arm or leg can often result in a number of false hits. Figure 7a shows a typical example.

False positives between humans. When multiple humans walk in a group in close proximity to each other, additional false hits are often generated in the spaces that form in-between the humans. An example of this phenomenon is shown in Figure 7b.

Other false positives. Occasionally, false positive classifications are generated unrelated to any occurrences of humans. These false hits occur around specific areas of moving objects such as the rear of a car, or at human-like shapes as the windows in Figure 7c. This type of misclassification is relatively rare.

False negatives. Although less common than false positives, there are of course also incorrect false classifications; that is, humans that are not detected. An image where this has occurred is shown in Figure 7d.

6.6. A complete cascade

In this section, the performance and behavior of a complete cascade is studied. The classification was carried out on an Intel Pentium 4 processor with a clock frequency of 2 GHz. The operating system used was Gentoo Linux with a version 2.4 vanilla kernel. The classifier was implemented in C++ using the LibSVM library, version 2.8 [2].

We can see from Table 1 that the vast majority of the possible positions are discarded at the first cascade stage. At the same time, the majority of the execution time was spent in the last stage. This corresponds to the intuition the the majority of the positions are easy to classify, while a small minority require more advanced classification.

7. Conclusions

A method for detection of humans in video was presented. To achieve maximal computational efficiency, fast spatio-temporal difference filters were used together with SVM classifiers. Furthermore, to make use of the fact that most classification cases in the detection procedure are easy, while a few are hard, a cascade architecture with increasingly complex SVM classifiers was utilized. The classifiers were trained with video from different street scenes, and experiments on detection accuracy were performed with the same type of data.

In the current implementation, the method does not meet the real-time demands (Section 6.6). A typical cascade takes in total 3.3 seconds, of which 2.2 seconds are spent on image preprocessing. However, several things could be done to speed up the detection:

Preprocessing. Effort has not been put down on optimizing the code for computing the filter responses. The preprocessing step could essentially be implemented in hardware, thus taking very little computational effort. As a comparison, another implementation of the same preprocessing step by Viola et al. [16] requires 0.15 seconds.

Using parts of the integral difference. The results in Section 6.3 indicate that while the extra information stored in the integral difference image leads to a much lower false positive ratio, it does not also lead to a lower false negative ratio, compared to the difference image. The reason could be that the 5 times higher dimensionality of the integral difference images makes the SVM classifier deteriorate, since many of the dimensions do not contain valuable information.[2] Non-informative dimensions could for example be the $F_t^{\uparrow}$ and $F_t^{\downarrow}$ images, which then contributes negatively to the performance even though the extra information from the $F_t^{\leftarrow}$ and $F_t^{\rightarrow}$ images contributes positively. Therefore, the effects of using only parts of the integral difference image should be investigated.

Smaller images. Another way of lowering the dimensionality of feature space is to lower the resolution of the filter response. The downside of this might be that the error rate increases since the information content in the filter responses decreases (see Section 6.4).

Cascade. Thorough experimentation on the design of the cascade must be carried out to determine the optimal combination of cascade stages.

[2] Another effect is of course higher computational cost. Remember from Section 4.2 that the computational complexity of the SVM is linearly dependent on the dimensionality.

(a) Example of a false positive around a human.

(b) Example of a false positive between two humans.

(c) A pair of windows in the facade in the upper left portion of the image is incorrectly classified as a human.

(d) The person walking away from the camera is not detected in this frame.

Figure 7: Examples of types of errors.

Table 1: Statistics of a cascade executed on a typical snapshot from one of the test sequences. A total of 14876 possible positions (x, y, s) are examined in the first cascade stage.

Stage	# Detections Passed	% Detections Passed	Time [ms] (Preprocessing/SVM)	% Time
Diff, $\dim(I_t) = 4 \times 8$, Linear	1287	8.7	708/1.46	21
Diff, $\dim(I_t) = 12 \times 24$, Linear	404	2.7	598/1.06	18
Int diff, $\dim(I_t) = 12 \times 24$, RBF	13	0.087	938/1102	61
Total	14876		2244/1105	

Each of these options are worth investigating. Together, they will most probably lead to a fast and robust real-time method for human detection in video.

References

[1] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[2] C-C. Chang and C-J. Lin. LibSVM: a library for support vector machines, 2005, *http://www.csie.ntu.edu.tw/∼cjlin/papers/libsvm.pdf*.

[3] D. Comaniciu and V. Ramesh. Robust detection and tracking of human faces with an active camera. In *IEEE International Workshop on Visual Surveillance*, 2000.

[4] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, UK, 2000.

[5] T. Darrell, G. Gordon, M. Harwille, and J. Woodfill. Integrated person tracking using stereo, color, and pattern recognition. In *CVPR*, pages 601–609, 1998.

[6] R. Fablet and M. J. Black. Automatic detection and tracking of human motion with a view-based representation. In *ECCV*, volume 1, pages 476–491, 2002.

[7] D. M. Gavrila. Pedestrian detection from a moving vehicle. In *ECCV*, volume 2, pages 37–49, 2000.

[8] I. Haritaoglu, D. Harwood, and L. Davis. W4: Real-time surveillance of people and their activities. *PAMI*, 22(8):809–830, 2000.

[9] S. Kang, H. Byun, and S-W. Lee. Real-time pedestrian detection using support vector machines. *International Journal of Pattern Recognition and Artificial Intelligence*, 17(3):405–416, 2003.

[10] B. Moghaddam and M-H. Yang. Sex with support vector machines. In *Advances in Neural Information Processing Systems 13*, pages 960–966, 2001.

[11] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, and T. Poggio. Pedestrian detection using wavelet templates. In *CVPR*, pages 193–199, 1997.

[12] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: an application to face detection. In *CVPR*, pages 130–136, 1997.

[13] H. Sidenbladh. Detecting human motion with support vector machines. In *ICPR*, volume 2, pages 188–191, 2004.

[14] Y. Song, X. Feng, and P. Perona. Towards detection of human motion. In *CVPR*, volume 1, pages 810–817, 2000.

[15] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, volume 1, pages 511–518, 2001.

[16] P. Viola, M. J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. In *ICCV*, pages 734–741, 2003.