Incorporating Computation Time Measures during Heterogeneous Features Selection in a Boosted Cascade People Detector

Alhayat Ali Mekonnen, Frédéric Lerasle, Ariane Herbulot, Cyril Briand

Incorporating Computation Time Measures during Heterogeneous Features Selection in a Boosted Cascade People Detector

A. A. Mekonnen[a], F. Lerasle[a,b], A. Herbulot[a,b], C. Briand[a,b]

[a] *CNRS, LAAS, 7, Avenue du Colonel Roche, F-31400 Toulouse, France*
[b] *Univ de Toulouse, UPS, LAAS, F-31400 Toulouse, France*

Abstract

In this paper, we investigate the notion of incorporating feature computation time measures during feature selection in a boosted cascade people detector utilizing heterogeneous pool of features. We present various approaches based on pareto-front analysis, computation time weighted adaboost, and Binary Integer Programming (BIP) with comparative evaluations. The novel feature selection method proposed based on BIP – the main contribution – mines heterogeneous features taking both detection performance and computation time explicitly into consideration. The results demonstrate that the detector using this feature selection scheme exhibits low miss rates with significant boost in frame rate. For example, it achieves a 2.6% less miss rate at 10^{-4} FPPW compared to Dalal and Triggs HOG detector with a 9.22x speed improvement. The presented extensive experimental results clearly highlight the improvements the proposed framework brings to the table.

Keywords: People Detection, Feature Selection, Binary Integer Programming, AdaBoost.

1. Introduction

In modern era computer vision plays a significant role in automated people detection. It has vast pool of applications spanning many research domains, including but not limited to: Human-Robot Interaction, Human-

Email addresses: aamekonn@laas.fr (A. A. Mekonnen), lerasle@laas.fr (F. Lerasle), aherbulo@laas.fr (A. Herbulot), briand@laas.fr (C. Briand)

Computer Interaction, Pedestrian Protection Systems (part of Advanced Driver Assistance Systems), Video Surveillance, and Automated Image Indexing and Management. Automated people detection involves perceiving the whereabouts of people in the information of a scene captured by a sensor. Depending on the mode of the sensor, this can mean localizing the accurate 3D position or rough 2D position of each person in the scene. Visible spectrum cameras (mostly called classical cameras) are the most widely used sensors as they capture very informative data covering wide spatial area, with color and texture information of the scene. This work focuses on this domain – people detection using visible spectrum cameras, commonly referred as visual people detection. Unfortunately, visual people detection is by far one of the most challenging tasks in computer vision, mainly due to *physical variation of people*, *body deformations due to articulation*, *illumination variation*, *viewpoint change*, *background clutter*, *occlusions*, *sensor limitations*, and *computational constraints*. In recent years astounding progresses have been made by the scientific community[4, 9, 12], but there is still room for improvement.

In many applications, e.g., robotics, video surveillance, a real time people detector is required. In these domains the frame rate of the detector is as important as the accuracy of the detector. For example, a mobile robot needs to be reactive during navigation/interaction in human occupied environments. Thus, its people detection module – which is one component of an entire functioning system – should be fast. The advent of powerful camera systems in the robotic community that provide high resolution omnidirectional images, *e.g.*, the Ladybug series[33] from Point Grey, stresses this point further urging the need to give extra focus on computation time during detector design. Hence, we propose a framework that tries to address this during detector design.

In this work, we investigate different approaches to incorporate computation time measures during people detector design. Balancing computation time and detection performance is challenging; best detection results are obtained using complex features and descriptors which are computationally expensive. As an example, Histogram of Oriented Gradients (HOG)[5] is the most discriminant single feature thus far, but it is also computationally expensive compared to simple features like Haar variants[43]. Furthermore, most detectors that improve over HOG either use complex human models, *e.g.*, parts based models[10], or consider various heterogeneous pool of features,[4, 7, 44, 47], both of which contribute to added computation time

unless explicit computation considerations are made. In line with this, we present investigations on a people detector that uses heterogeneous pool of features and makes explicit computation time versus detection trade-off optimization to build a performant detector that leads to a significant gain in computation time while maintaining acceptable detection performance.

This paper is structured as follows: Section 2 presents an overview and related works on visual people detection. Section 3 presents the proposed people detection and detector learning framework. Subsequent sections present the different components of the proposed framework in detail, namely: features and weak classifiers, in section 4; and feature selection and classifier learning, in section 5. All experiments carried out and obtained results are detailed in section 6. Finally, the paper finishes with discussions in section 7 and concluding remarks in section 8.

2. Overview and Related Work

Undoubtedly, automated people detection is a very important research area with prominent applications. All methods in the literature more or less adhere to the generic scheme depicted in figure 1. For a given input image, possible candidate windows are hypothesized. Using the person model adopted, the original raw image input is transformed into a convenient format by extracting certain types of features that capture specific cues. Finally, each hypothesis is labeled as either a person or not using a learned classification rule. Even though not shown in the figure, there is usually a last post-processing step in the form of *Non-Maximal Suppression* (nms) which merges multiple detections that may arise from the same person into one. The scheme shown in figure 1 shows the flow used during detection. The types of features, descriptors, classifiers, and exact person model employed is a detector design choice. But, the actual subset of features/descriptors to use and the exact classifier parameters are determined via a training, also called learning, phase using a training dataset that contains positive and negative instances in a supervised learning approach. Recently, unspervised learning approach based on convolutional neural networks have also been successful applied to person detection, notably Sermant et al. *ConvNets*[38] detector.

The entire literature in visual people detection is overwhelming and a discussion on the different techniques is beyond the scope of this paper (please refer to[4, 7, 9, 12] for extensive surveys). The presentation here is limited to the approaches that are relevant for this work. We will primarily focus on

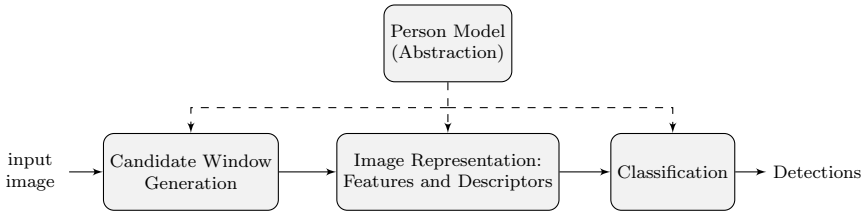


Figure 1: Important components of a visual people detector.

approaches that rely on sliding window candidate window generation mode which generates candidate windows with a fixed aspect ratio by sampling at all positions and scales of the input image.

In visual people detection, a person can be modeled either as a holistic indivisible object or as a parts-based entity. The holistic representation[43, 5, 8], considers a person as a whole indivisible object. On the other hand, parts-based approaches[24, 49, 10], rely on detecting different parts of a body – either explicitly looking for a head, torso, arms, and legs or looking for implicit dividends – to detect a person. Holistic approaches are simple, have straight forward model training, and have reduced computation time, relatively speaking, during detection. But, they perform poorly with non-standard poses (articulations) and partial visibility. On the other hand, parts based approaches are better suited for person detection thanks to their ability to better deal with partial occlusions, view point changes, and pose variations. However, they are difficult to train and computationally demanding during detection. They also perform perform poorly with lower resolution images as the parts require ample spatial support for robustness

The classification stage is responsible for labeling each candidate window generated and described in accordance with the abstraction adopted as either a person or not. This block can either output a binary label (person or non-person) or a continuous valued score that reflects its confidence, and can further be thresholded to provide a binary label. These classifiers are mostly trained with a discriminative learning algorithm given positive and negative example instances. Discriminative learning algorithms in addition to robust image representation are the key reasons to recent advances in people detection. The most frequently used discriminative classifiers for people detection are variants of Support Vector Machines (SVM) and Boosted classifiers. On few occasions Fisher's Linear Discriminant Analysis (LDA)[30], and Artificial Neural Networks[39, 54] have also been used; recently, Random Forest

classifiers are also gaining a significant momentum[40].

Features enable us to capture the essence of the underlying scene by extracting meaningful information from a group of data points (pixels). Different features capture differing facets of the underlying scene and careful feature choice plays an important role on the detection performance. Early success in people detection was achieved using rudimentary Haar like features inspired by Haar Wavelets[32, 43]. These features capture region intensity differences which has limited descriptive power. Especially considering the distinctive boundary of peoples' figures, which can better be captured using edges. This intuition led to the adoption of gradient based features: Edge Orientation Histogram (EOH)[17, 13] and Histogram of Oriented Gradients (HOG)[5]. Recently, variants of Local Binary Pattern (LBP)[28] features have also been burgeoning. Color features are rarely used in person detection because of the variability induced by clothing. But, color shows local similarity even over clothing. This notion was exploited by Color Self Similarity (CSS) features, proposed by Walk *et al.*[44], which proved successful by encoding similarities in different sub-regions. Looking at the trend in the literature, the gist in features used for people detection can be captured with two important terms: gradient and histogram. The most successful features consider image gradients with local pooling in the form of histograms. This is evident considering peoples' global silhouettes, illumination and contrast variations in imaging, and deformation in physical structure. In general, these considerations tend to lead to complex features that require increased computation time entailing more focus on computation time related optimizations. This being said, the next natural question would be, how about combination of features? Indeed, using a combination of features have shown to improve detection further, for example, the top 4 current best detectors (in terms of detection performance) in the state-of-the-art use a mixture of heterogeneous features[7].

Heterogeneous features help capture complementary information useful to handle various detection challenges – the more complementary the features, the better. Many works in the literature have attested this complementary nature. Geronimo *et al.*[13] showed this with Haar like features and EOH; Wang *et al.*[45] with HOG and LBP; Wojek *et al.*[47] with Haar like features, HOG, and shape context features; Walk *et al.*[44] with a concatenation of HOG, Histogram of Flow (HOF[6]), and CSS. Similar conclusions were also made by Schwartz *et al.*[37] and Hussain and Triggs[15] using – HOG, color frequency, and co-occurrence features – and – HOG and LBP variant fea-

tures – respectively. This is also true with the recently burgeoning integral channel features derivatives[4].

Given heterogeneous pool of features, different ways can be used to build the final detector. Four main trends can be observed in the literature: (1) Direct concatenation[44, 47] in which the different features are concatenated to make one high dimensional feature vector and an SVM used afterwards for classification. This is computationally costly owing to the complex feature and SVM weights applied in sliding window detection. Dimensionality reduction techniques after concatenation do improve detection performance but not detection speed[15, 37]. (2) Direct boosting where an ensemble classifier is learned using the entire heterogeneous pool of features[8, 47, 13]. The problem here is in boosting, on each iteration, the feature with the least weighted classification error is added to the ensemble irrespective of its computation time. This favors complex features resulting in computationally costly detector. (3) Coarse-to-fine hierarchical arrangement where a cascade is constructed using cheap features at the initial stages and using complex features at later stages[25, 31]. This approach is quite advantageous and tries to find a balance between detection performance and speed. The concern is, how to decide which features to use at the different stages systematically? Both[25, 31] adopt a heuristic based rule and use homogeneous family of features they deemed cheap at the initial stages, and homogeneous complex features at the latter. Finally, (4) computation time versus detection trade-off. This notion has been considered in the works of Wu and Nevatia[50] and Jourdheuil *et al.*[16]. In all cases, they defined a criterion composed of feature detection performance and computation time in a multiplicative manner. But, considering a multiplicative factor masks the contributions from the corresponding objectives and is not guaranteed to be optimal.

In this paper, which is an extended version of our previous work Mekonnen *et al.*[23], we investigate different schemes to incorporate feature computation time measure during feature selection and primarily focus on Binary Integer Programming (BIP) based feature selection. The proposed BIP framework falls in the 4^{th} category; but, it can also be considered as a variant of coarse-to-fine hierarchy in which the exact features to use at each cascade node are selected automatically via an optimization step. We use five pervasively used heterogeneous features that exhibit significant discriminative power and computation time differences, namely: Haar-like features[43], Edge Orientation Histogram (EOH)[13], CSS[44], Center Surround Local Binary Patterns (CS-LBP)[14], and HOG[5] in a classical cascaded boosting configuration[43]

with an added explicit optimization step based on BIP to select a subset of features that have the least combined computation time and achieve a stipulated detection performance.

Recently the trend in the community has shifted towards channel features based approaches and their derivatives thereof[52, 7, 27, 51, 3]. The basic principle is to transform the image into a set of feature channels and then to construct a feature vector by pooling over with a set of varying rectangular regions. These channel features are mostly used with boosting classifiers to learn a detector[4]. Their conception stems back to integral channel features (*ChnFtrs*)[8]. The idea is quite similar to Haar like features, but in stead of sum-pooling on intensity images, it is done on heterogeneous features, for example, 6 orientation quantized HOG feature and LUV color feature channels. The majority of approaches utilizing channel features can be generalized using the concept of filtered channel features[52] which basically considers the variants as applying a specific type of filter. Since these features are computed with the help of integral images over each channel, ignoring pre-processing steps, each feature will basically have the same computation time. These features are designed for fast computation. As a result, even though our BIP framework is quite generic (and works best with heterogeneous features in terms of both detection performance and speed), channel features and their derivatives are not considered. The majority of approaches utilizing channel features can be generalized using the concept of filtered channel features [52] which basically considers the variants as applying a specific type of filter. For example, *ChnFtrs* [8] a filter composed of random rectangular shapes, *ACF* [7] a filter with a 4×4 pixels pooling region, *SquaresChnFtrs* [3] a square pooling regions based filters of various sizes, *LDCF* [27] (short for locally decorrelated channel features) PCA bases as filter banks, and *Checkerboards* [52] filters with checkerboard patterns. Even the *InformedHaar* [51] detector that uses hand crafted (from a training sample) binary and ternary Haar like feature templates that conform to the shape averaged pedestrian gradient image obeys this generalization. Other detectors that are currently within the state-of-the-art but that do not fit in the filtered channel representation include: *SketchTokens* [19], *Regionlets* [46], and *Spatialpooling* [29]. Evidently, the literature also encompasses several other detectors. To name a few, using their commonly used acronyms: *PoseInv* [20], *HikSvm* [21], *Pls* [37], *MultiFtr* [47], and *MultiFtr+CSS* [47].

Furthermore, paradigms that stray from the usually manually crafted feature set selection/design with the use of deep learning frameworks, also

known as deep networks, have emerged. Deep networks are gaining a strong momentum due the improved classification performance, and their ability to operate on raw input pixels [2, 1, 41, 42]. These approaches rely on convolutional neural networks that automatically learn pertinent features, in a deep cascade, that are relevant to discriminate people from background. Even though, some authors claim better features for pedestrian detection have not yet been obtained [4], the recent work of Angelova *et al.* indicate better performance along with improved speed [2, 1] and seem to indicate promising future of deep networks in people detection.

Contributions: The work presented in this paper makes two core contributions. First, it present different detector learning paradigms that incorporate computation time measures of features during feature selection – amongst which is a novel BIP formulation to mine heterogeneous features taking both detection performance and computation time into consideration. This optimization applied to heterogeneous features marks a key contribution. Second, the paper presents a thorough evaluation of the proposed person detector – using both proprietary and public datasets – with detailed analysis of its performance compared to alternative approaches. The proprietary dataset, called Ladybug dataset (section 6.1.1), is composed of images acquired with the Ladybug2 omni-directional camera system mounted on a mobile robot. The high resolution and versatile images from the Ladybug camera series further underline the need for the proposed detector.

3. Proposed People Detector Framework

Evidently, the framework adopted to address the aforementioned objectives needs to be concerned by detector detection performance and its associated computation time. The most famous detector configuration suitable for these requirements is the attentional cascade detector configuration pioneered by Viola and Jones[43]. This configuration builds a cascade made of nodes resembling a degenerate tree. Each node rejects negative candidate windows and passes along potential positive windows onto the next stage for more scrutinized verification. Figure 2 illustrates this configuration made up of K nodes. Given a candidate window, it is passed along the cascade with a label T (for true) if it fulfills the test encountered at each node, otherwise it is rejected (labeled as F for false). A window is considered to be positive only if it makes it to the end of the cascade. This leads to an efficient structure that

uses simple classifiers at the beginning of the cascade, which reject a majority of the negative samples, and complex classifiers as one progresses along the cascade speeding up detection drastically. This structure has gained wide acceptance and has even been applied in recent part-based approaches[10].

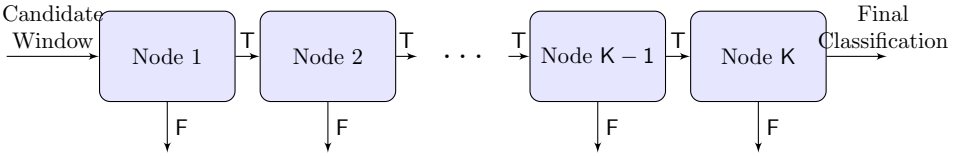


Figure 2: An attentional detector cascade configuration.

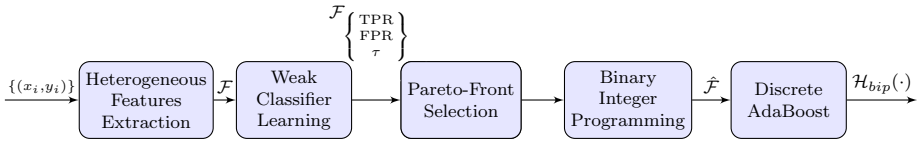


Figure 3: Proposed cascade node training scheme using heterogeneous pool of features and binary integer programming.

Figure 3 shows a block diagram representation of the the proposed classifier learning paradigm. For a cascade node, given labeled positive and negative training sets, the heterogeneous features, $\mathcal{F}$, described in section 4 are extracted and associated weak classifiers trained. Then a pareto-front extraction technique is employed to reduce the feature sets to a manageable size, $\tilde{\mathcal{F}}$. This is followed by a multi-criteria discrete optimization based feature selection technique that identifies a subset of the features that fulfill the stipulated detection performance while exhibiting the smallest combined computation time. This is realized using Binary Integer Programming (BIP) – presented in section 5.1.2 – which furnishes a few set of selected features, $\hat{\mathcal{F}}$, taking the optimization objective and constraints into account. Finally, discrete AdaBoost is used to learn the nodal strong classifier, labeled as $\mathcal{H}_{bip}(\cdot)$. Each of these constituent components are presented in detail in subsequent sections.

Additionally, to highlight the pros and cons of the proposed framework, it is compared with several learning strategies, namely: a pareto-front and AdaBoost, a computation time weighted AdaBoost, and a random sampling and

feature rearranging based learning strategies. The details of these techniques are provided in section 5.2.2.

4. Features and Weak Classifiers

This section presents the heterogeneous features along with corresponding weak classifiers used for each family of features and associated computation time analysis.

4.1. Heterogeneous Feature Set

In this work, we have chosen to use the following five family of heterogeneous features: Haar like features, Edge Orientation Histograms (EOH), Center-Symmetric Local Binary Patterns (CSLB), Color Self Similarity (CSS) features, and Histogram of Oriented Gradients (HOG). These choices are motivated mainly by two factors: (1) their frequent use in the literature for person detection, and (2) their complementary nature (in terms of both discrimination and computation requirements). EOH and HOG capture edge distributions, CSS focuses on color symmetry, and Haar-like and CS-LBP on intensity and texture variations. Each feature family is extracted within a given image window of 128×64 pixels denoted as R, a standard template size used prominently in people detection[9]. To generate the over-complete set of features, the position, width, and height of the region the features are computed is varied within the candidate window. In all references, (x, y) position refers to the top left corner of the region R relative to the top left corner of the candidate window, and (w, h) refers to the width and height of the region spanned for extraction.

4.1.1. Haar Like Features

Haar like features represent a fast and simple way to compute region differences. These features have been extensively used for face, person, and various object detections[32, 43, 18, 13]. For a given feature, the response is obtained by subtracting the sum of pixels spanned by the black region from the sum of pixels spanned by the white region, see figure 4. To incorporate various measures, we have used the extended Haar like features from Viola and Jones[43] and Lienhart and Maydt [18], which contain upright and tilted filters of various configurations as shown in figure 4.

If the operator $\Omega_{haar}(R, x, y, w, h, \varphi)$ denotes the feature extracted (scalar value) in the overlaid region (x, y, w, h) within the candidate window R using

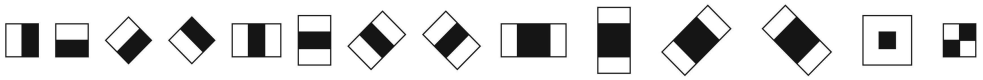


Figure 4: Set of extended Haar like feature types (configurations) used.

Haar filter type φ, the over-complete set of Haar like pool of features, denoted as $\mathcal{F}_{haar}$, is obtained by extracting features for all possible combinations of x, y, w, h, φ in R.

*4.1.2. Edge Orientation Histogram (*EOH*)*

EOH is another popular feature set that has been used for people detection[13]. These features represent ratios of gradients computed from edge orientation histograms. Within a given overlaid region, first gradients are computed. Then a gradient histogram is built by quantizing the gradient orientations. Finally, the ratios of each histogram bin with one another make up individual features. Similarly, let the operator $\Omega_{EOH}(R, x, y, w, h, k_{b_1}, k_{b_2})$ denote the feature extracted in the overlaid region (x, y, w, h) within R by first building an edge orientation histogram and then taking the smoothed ratio of the histogram counts in any two bins k_{b_1} and k_{b_2}. Consequently, the over-complete EOH feature pool set, denoted $\mathcal{F}_{EOH}$, is constructed by extracting feature values for all possible combinations of x, y, w, h and any two bins k_{b_1} and k_{b_2} within R.

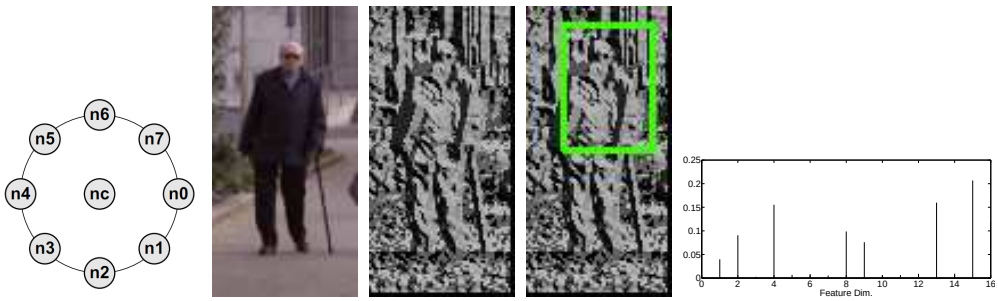
*4.1.3. Local Binary Pattern (*LBP*)*

LBP was initially proposed as a texture characterization features[28]. Since then, it has been used in many applications – primarily facial analysis[53], and person detection[26]. To date, many variants of LBPs have been proposed. In this work, we adhere to the Center-Symmetric Local Binary Pattern (CS-LBP) variant due to its small dimensional histogram and demonstrated good results on person datasets[14].

$$\text{CS-LBP} = s(n0 - n4)2^0 + s(n1 - n5)2^1 + s(n2 - n6)2^2 + s(n3 - n7)2^3 \qquad (1)$$

where, $s(x) = \begin{cases} 1 & x \geq 0 \\ 0 & otherwise \end{cases}$ and n0,...,n7 are gray scale pixel values (figure 5a).

In our implementation, CS-LBP is computed over a 3×3 pixel region (best results reported in[14]) by comparing the opposite pixels and adding a



(a) (b) (c) (d) (e)

Figure 5: CS-LBP feature extraction steps. (a) Pixel neighborhood for use with equation 1 (8-connectivity), (b) original candidate image, (c) dense CS-LBP per pixel computed values, (d) one specific feature specified by a bounding box, and (e) actual feature vector extracted from (d).

modulated term according to equation 1 with respect to figure 5a. This gives a scalar value less than 16 which is assigned to the center pixel. This is done for all the pixels in the window. A sample raw feature image is shown in figure 5c (the values are scaled to aid visibility). Finally, the actual feature vector is extracted by constructing a CS-LBP histogram (figure 5e) over a given overlaid region, figure 5d. Let $\Omega_{CLBP}(R, x, y, w, h)$ denote the feature vector constructed by making a histogram of all CS-LBP features within the region (x, y, w, h). Extracting feature vectors for all possible combinations of x, y, w, h within R with strides of 4 pixels in both direction gives the LBP feature pool, denoted $\mathcal{F}_{CLBP}$. The histograms have 16 bins corresponding to CS-LBP quantization levels.

*4.1.4. Color Self Similarity (*CSS*)*

Color features are rarely used in person detection because of the variability induced by clothing. Color, actually, shows local similarity even over clothing. CSS features, proposed by Walk *et al.*[44], encode similarities in different sub-regions. To compute the features, first the image window is subdivided into non-overlapping blocks of 8×8 pixels and within each block a $3 \times 3 \times 3$ color histogram in HSV space is built with interpolation. Then similarities are computed by intersecting individual histograms. In[44], all histogram intersection values are concatenated to define one feature vector. But here, we define the intersection of one histogram block with the rest of

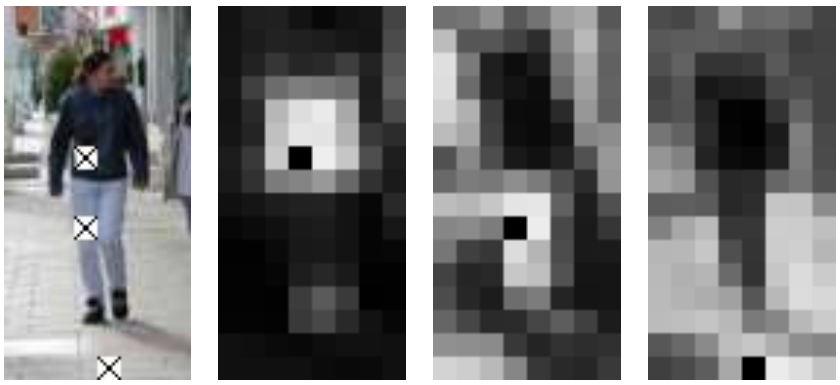
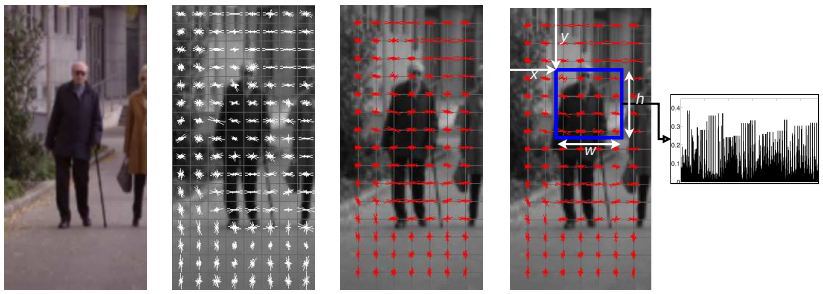


Figure 6: Illustration of sample CSS features.

the blocks as a single feature. With an 8×8 block size and 128×64 window size, there are 128 different blocks. The intersection of one block with the rest gives 127 scalar values (excluding intersection with itself). These scalar values all together make-up the feature vector computed for that specific block location. This is repeated for each block resulting in 128 different features – the CSS feature pool ($\mathcal{F}_{CSS}$). Figure 6 shows three sample features computed at the crossed block positions; observe how neighboring blocks show similarity.

*4.1.5. Histogram of Oriented Gradient (*HOG*)*

As it has been mentioned, no other single feature has been able to supersede HOG features[9]. Hence, naturally, we have resorted to use them.



(a) (b) (c) (d)

Figure 7: Illustration of the HOG feature pool set generation.

In this proposed approach, we use the original HOG features proposed by Dalal and Triggs[5] along with their widely preferred/used computation,

i.e., a cell size of 8×8 pixels, a feature block size of 2×2 cells and an 8 pixel horizontal and vertical stride. But, instead of using the entire descriptor as a single feature, we generate pool of features by concatenating only a subset of the block histograms. The main steps are illustrated in figure 7. Given a candidate window (figure 7a), cell histograms are computed (figure 7b), and neighboring cells are combined and normalized to compute block histograms (figure 7c). Then differing from the original proposition, a single feature is described by concatenating all block histograms inside a region parameterized by a starting location (x,y), width (w), and height (h) as shown in figure 7d. Finally, the entire feature pool $\mathcal{F}_{HOG}$ is generated by varying x, y, w, and h for all possible positive values in the given candidate window. This leads to a total of 3360 features with dimensions ranging from 36 (smallest) to $7 \times 15 \times 36 = 3780$ (highest and equivalent to the feature vector obtained when concatenating all block histograms).

Finally, the complete heterogeneous feature pool is determined by merging all heterogeneous feature pool sets, *i.e.*, $\mathcal{F} = \{\mathcal{F}_{Haar}, \mathcal{F}_{EOH}, \mathcal{F}_{CLBP}, \mathcal{F}_{CSS}, \mathcal{F}_{HOG}\}$. In consecutive sections, each individual feature is indexed by j, where $j \in \{1, 2, ..., |\mathcal{F}|\}$.

4.2. Weak Classifiers

The complete heterogeneous pool of features comprises of scalar and multi-dimensional features. For all scalar features, *i.e.*, Haar-like and EOH features, we have chosen to use decision trees as a weak classifier. A decision tree over a real valued scalar feature is equivalent to having multiple threshold values assigning different bands of the range for positive and negative samples. In light of the detection performances exhibited, linear SVM is used as weak classifier for HOG and CSS feature vectors. Unlike HOG and CSS, the total number of CS-LBP features is quite high (see table 1) and using SVM leads to an overwhelming training period. Consequently, for CS-LBP features, we employ Fisher's Linear Discriminant Analysis (LDA)[11] with decision trees. Since LDA seeks to find a projection direction which better discriminates the two classes, we train an LDA projection vector for all the CS-LBP features once using the positive and negative samples provided at the first node of the cascade. Afterwards, the same projection vectors are used and only a new decision tree is trained on the projected data. Experimentally, this has lead to a good balance between weak classifier performance and training duration. Each weak classifier, associated with a unique feature,

is denoted as $\mathfrak{h}_j$ and maps each instance of the training set X to a discrete label, $\mathfrak{h}_j : X \rightarrow \{-1, +1\}$.

4.3. Computation Time

The computation time of each feature is determined irrespective of any implementation optimization that can be done during detection, *e.g.*, use of caches to buffer some features. This helps establish an upper bound. For each feature considered, the associated computation time is made up of two components. A part associated with image pre-processing (including rudimentary feature preparation, integral image computation, etc) and a second part pertaining to the feature extraction and necessary computation during detection (*e.g.*, multi-dimensional feature projection). For a feature indexed by j, these are represented as $\tau_{p,j}$ and $\tau_{e,j}$ consecutively; the combined computation time of that feature becomes $\tau_j = \tau_{p,j} + \tau_{e,j}$. These values are determined by averaging over $1,000$ repeated iterations. The computationally cheapest feature, a two boxed horizontal Haar filter which takes a total time (τ_j) of 0.0535 μs to compute on a core $i7$ machine, is used as a reference to report computation time for other features. The range of computation time for each feature family is reported in table 1. The table summarizes the characteristics of the heterogeneous pool of features considered: the total number of features in each family, the minimum and maximum feature computation time (both pre-processing, τ_p, and extraction, τ_e) along with the weak classifier used are listed.

Table 1: Feature pool summary with minimum and maximum feature computation time in each feature family. Time is reported as a multiple of $\mathfrak{u} = 0.0535\ \mu s$ computed on a core i7 machine running at 2.4 Ghz.

Feature Type	No of features	τ_{min}		τ_{max}		Weak Classifier
		$(\tau_p)_{min}$	$(\tau_e)_{min}$	$(\tau_p)_{max}$	$(\tau_e)_{max}$	
Haar like	672,406	0.6$\mathfrak{u}$	0.40$\mathfrak{u}$	1.88$\mathfrak{u}$	1.60$\mathfrak{u}$	Decision Tree
EOH	712,960	2.72$\mathfrak{u}$	2.11$\mathfrak{u}$	315.65$\mathfrak{u}$	2.10$\mathfrak{u}$	Decision Tree
CS-LBP	59,520	1.24$\mathfrak{u}$	14.26$\mathfrak{u}$	111.60$\mathfrak{u}$	282.04$\mathfrak{u}$	LDA + Decision Tree
CSS	128	560.75$\mathfrak{u}$	457.19$\mathfrak{u}$	560.75$\mathfrak{u}$	457.19$\mathfrak{u}$	SVM
HOG	3,360	10.59$\mathfrak{u}$	479.12$\mathfrak{u}$	315.75$\mathfrak{u}$	51103.80$\mathfrak{u}$	SVM

Haar like, and EOH features are computed with the help of integral images, which is accounted for in the pre-processing time component. In fact, each $\tau_{p,j}$ of these feature families shares a fraction of the total time taken to construct the integral image proportional to the area it spans. Additionally,

for Haar like features, a horizontal and vertical stride of 2 pixels are used to generate the over-complete set and for EOH gradient orientation quantization levels of 4 (give best results[13]) and horizontal and vertical strides of 4 pixels are used.

5. Feature Selection and Nodal Strong Classifier Learning

As presented in section 3 and depicted in figure 3, the main focus of this work is a boosted cascade learning framework primarily utilizing binary integer programming for feature selection. The novelty lies on the BIP optimization formulation to select a subset of heterogeneous features with the least combined computation time that satisfy the required detection performance. Hence, in this section, the sequence of steps involved in the BIP based framework is initially presented under subsection 5.1. Consequently, to put the BIP based approach into perspective, the section continues with a presentation of other investigated boosted cascade learning alternatives in subsection 5.2.

5.1. BIP based Classifier Learning

The BIP based nodal strong classifier learning block diagram is shown in figure 3. The heterogeneous pool of features and their associated weak classifiers have been presented in section 4. The objective of this learning scheme is to build a nodal strong classifier $\mathcal{H}_{bip}(\cdot)$ from the extracted features $\mathcal{F}$. This is essentially achieved with three constituent steps: Pareto-Front extraction, binary integer programming, and discrete AdaBoost training. Each of these components are presented as follows. The final cascade detector trained using this scheme is referred with the label *BIP+AdaBoost*.

5.1.1. Pareto-Front Extraction

Pareto-Front analysis deals with selecting the optimal solutions when faced with competing multi-objective optimization criteria – like in equation 2 where the objective is to minimize miss rate (MR), false positive rate (FPR), and computation time (CT). It is termed competing because one has to worsen the other objectives to improve itself. The optimal solutions for these kind of optimization are termed as the Pareto optimal solutions. Pareto-Front analysis is used to find these optimal solutions that make up the Pareto optimal set. By applying this, a subset of features that are Pareto optimal with respect to MR, FPR, and CT, are retained for further use. A

feature $\mathbf{x}$ is said to *dominate* another feature $\mathbf{x}'$, only if $g_l(\mathbf{x}) \leq g_l(\mathbf{x}')$ for all $l \in \{MR, FPR, CT\}$, and $g_l(\mathbf{x}) < g_l(\mathbf{x}')$ for at least one l. Here, the $g_l(\cdot)$ denotes the MR, FPR, or CT, determined using the weak learner trained with the corresponding feature $\mathbf{x}$ and evaluated on a validation set. On the other hand, a feature is said to be *non-dominated* if no other feature *dominates* it. The Pareto optimal features set, $\tilde{\mathcal{F}}$, is actually the set that fulfills equation 3 which can easily be determined using algorithm 1.

$$\begin{aligned} minimize \quad & g(\mathbf{x}) = [g_{MR}(\mathbf{x}), g_{FPR}(\mathbf{x}), g_{CT}(\mathbf{x})] \\ s.t. \quad & \mathbf{x} \in \mathcal{F} \end{aligned} \tag{2}$$

$$\tilde{\mathcal{F}} = \{\mathbf{x} \in \mathcal{F} \mid \forall_{\mathbf{x}' \in \mathcal{F}} \;\; g_{MR}(\mathbf{x}') \geq g_{MR}(\mathbf{x}) \vee g_{FPR}(\mathbf{x}') \geq g_{FPR}(\mathbf{x}) \vee g_{CT}(\mathbf{x}') \geq g_{CT}(\mathbf{x})\} \tag{3}$$

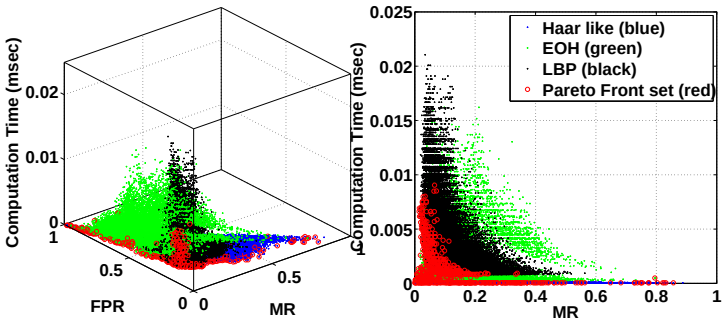


Figure 8: Sample Pareto-Front extraction. (Best viewed in color.)

Figure 8 illustrates a computed Pareto-Front, in 3D space and projected (MR vs computation time) 2D plot. To ease visibility only sub-sampled Haar like (shown in blue), EOH (in green), CS-LBP (in black) features are considered[1]. The determined Pareto optimal set is shown with red circles.

[1]This is done for demonstration purposes but in the actual cascade construction all five feature pools without sub-sampling are considered.

Algorithm 1 Pareto-Front Computation

```
1: procedure PARETO_FRONT(F)
2:     F̃ ← F_1
3:     for each x ∈ F do
4:         add ← true
5:         for each x' ∈ F̃ do
6:             if x dominates x' then
7:                 F̃ ← F̃\{x'}; continue;
8:             else if x' dominates x then
9:                 add ← false; break;
10:            end if
11:        end for
12:        if add then F̃ ← F̃ ⋃ {x}
13:    end for
14:    return F̃
15: end procedure
```

5.1.2. Binary Integer Programming (BIP)

Given the substantially decreased feature set $\tilde{\mathcal{F}}$ using Pareto-Front analysis, a further subset of features $\hat{\mathcal{F}} \subseteq \tilde{\mathcal{F}}$ are selected using the BIP formulation presented here. BIP is a special case of integer programming where decision variables are required to be 0 or 1 (rather than arbitrary integers). It aims at minimizing a given linear objective function $f = c^{\intercal}\mathbf{v}$ subject to the constraints that $A.\mathbf{v} \geq b$, where $\mathbf{v}$ represents the vector of 0-1 variables (to be determined), c and b are known coefficient vectors, A is a matrix of coefficients (called constraint matrix). It is well-known that BIP is $\mathcal{NP}$-hard in the strong sense, but in practice, branch-and-cut techniques can solve huge binary integer program very fast [34, 48]. Here, BIP is used to select a subset of features that fulfill the detection performance stipulated (in terms of TPR and FPR) with the minimum combined computation time. With respect to the reduced feature set, $\hat{\mathcal{F}}$, the BIP solutions are going to be the optimal choices. The BIP selected feature set $\hat{\mathcal{F}}$ is further used to build a strong nodal classifier with discrete AdaBoost (see subsection 5.1.3), $\mathcal{H}_{bip}(\cdot)$. The BIP formulation for feature selection is presented as follows.

Definition of parameters:. The following are list of parameters used in the optimization specification along with their definitions. A binary set is denoted as $\mathbb{B} = \{0, 1\}$.

- $N = \{1, ..., n\}$: set of training sample indexes with $n \in \mathbb{Z}$; a total of n training samples indexed by i;

- $M = \{1, ..., m\}$: set of weak learners indexes with $m \in \mathbb{Z}$; a total of m weak learners indexed by j;
- $\mathbf{y}^+ \in \mathbb{B}^n$, $\mathbf{y}^+ = \left\{y_i^+\right\}_{i \in N}$; $\mathbf{y}^- \in \mathbb{B}^n$, $\mathbf{y}^- = \left\{y_i^-\right\}_{i \in N}$; notice $y_i^- + y_i^+ = 1 \ \ \forall i \in N$

$$y_i^+ = \begin{cases} 1 & \text{if } i \text{ is positive} \\ 0 & \text{else} \end{cases} \qquad y_i^- = \begin{cases} 1 & \text{if } i \text{ is negative} \\ 0 & \text{else} \end{cases}$$

- $\mathbf{H} \in \mathbb{B}^{nxm}$ where $\mathbf{H} = \{\mathfrak{h}_{i,j}\}_{\substack{i \in N \\ j \in M}}$ with $\mathfrak{h}_{i,j} \in \{0, 1\}$

$$\mathfrak{h}_{i,j} = \begin{cases} 1 & \text{if weak learner } \mathfrak{h}_j \text{ detects sample } i \text{ as positive} \\ 0 & \text{else} \end{cases}$$

- TPR $\in [0, 1]$: minimum true positive rate set at the considered node of the cascade;
- FPR $\in [0, 1]$: maximum false positive rate at the node;
- $\tau \in \mathbb{R}^m$: with $\tau = \{\tau_j\}_{j \in M}$ computation time of weak learner j.

Decision Variables:. In BIP, the decision variables are restricted to binary values, values from the set $\mathbb{B} = \{0, 1\}$. The BIP decision variables are the following.

- $\mathbf{v} \in \mathbb{B}^m$, $\mathbf{v} = \{v_j\}_{j \in M}$ $v_j \in \{0, 1\}$: $v_j = 1$ if weak learner $\mathfrak{h}_j$ is selected, else $v_j = 0$;
- $\mathbf{t} \in \mathbb{B}^n$, $t_i \in \{0, 1\}$: $t_i = 1$ if a positive sample i has been detected as positive (true positive) by at least one selected weak learner, else $t_i = 0$;
- $\mathbf{f} \in \mathbb{B}^n$, $f_i \in \{0, 1\}$: $f_i = 1$ if a negative sample i has been detected as positive (false positive) by at least one selected classifier, else $f_i = 0$.

Let vector $\mathbf{p} = \{p_i\}_{i \in N} = \mathbf{Hv}$, which denotes the total number of weak learners that have labeled each training sample i as positive.

Objective Function and Constraints:.

$$
\begin{aligned}
&\min \ \tau^{\intercal}\mathbf{v} && && (1)\\
&\text{s.t} \quad t_i \leq y_i^{+} \cdot p_i && \forall i && (2)\\
&\qquad f_i \geq y_i^{-} \cdot \mathfrak{h}_{i,j} \cdot v_j && \forall (i,j) && (3)\\
&\qquad \|\mathbf{t}\|_1 \geq \|\mathbf{y}^{+}\|_1 \cdot \mathrm{TPR} && && (4)\\
&\qquad \|\mathbf{f}\|_1 \leq \|\mathbf{y}^{-}\|_1 \cdot \mathrm{FPR} && && (5)\\
&\qquad \mathbf{v} \in \mathbb{B}^n; \mathbf{t} = \{t_i\}_{i \in N}, \mathbf{f} = \{f_i\}_{i \in N}; \mathbf{t}, \mathbf{f} \in \mathbb{B}^n && && (6)\\
&\qquad \|\cdot\|_1 \text{ is the } l_1 \text{ norm.}
\end{aligned}
$$

The objective function (1) aims at minimizing the computation time. Constraints (2)-(5) express that a given rate of detection quality has to be reached (depending on the number of true and false positives). Constraints (2) link v_j and t_i variables (via p_i) so that $t_i = 0$ if image i has not been well-recognized by at least one selected classifier. Since it is at least, it is not expressed with respect to each selected weak classifier j – but to the sum of all selected classifiers (sum over j, the entries of p_i as it is). Constraints (3) link v_j and f_i variables so that $f_i = 1$ if a negative image i has been recognized as positive by at least one selected classifier. A negative sample should be correctly labeled (as 0) by all selected weak classifiers. It is considered as a false positive ($f_i = 1$) if at least one selected classifier labels it as positive. So in this case, there will be a distinct constraint for each sample and feature combination, hence the need for the $\forall(i, j)$. Constraint (4) expresses that the rate TPR of true positives, obtained with the selected classifiers, has to be reached. Similarly, constraint (5) expresses that the rate FPR of false positives, obtained with the selected classifiers, must not be exceeded. In this formulation, there are a total of $(n \cdot (m + 1) + 2)$ number of constraints, which could be huge for large n and m values. The final subset of features $\hat{\mathcal{F}}$ corresponds to only the selected features, *i.e.*, non zero $\mathbf{v}$ entry; since each feature indexed by j is associated with a unique weak learner $\mathfrak{h}_j$, $\hat{\mathcal{F}}$ also represents the subset of weak learners retained. At this point, a classification rule can be materialized by looking at p_i for a given sample i. If p_i is greater than zero, it means one weak classifier out of the selected ones has labeled the sample as positive. But, in order to break this discrete nature, we instead use discrete AdaBoost to further train a strong classifier. This provides a means to: (1) improve the classifier's generalization to unseen samples with the help of weights assigned to the weak classifiers based on discriminative ability; and (2) provide additional flexibility to adjust the TPR and FPR

margin, by varying a threshold value, with the same computation time – this could, for example, be useful to fine tune an already trained detector for a new dataset.

5.1.3. Discrete AdaBoost

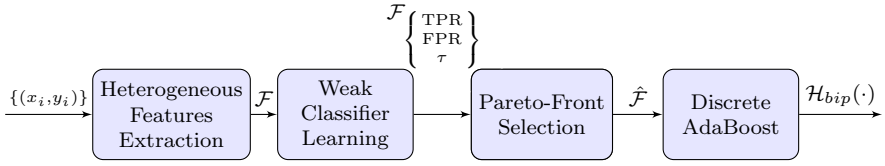
Discrete AdaBoost is one instance of the Boosting classifier variants which build a strong classifier as linear combination (weighted voting) of a set of weak classifiers. Suppose, we have a labeled training set $\{(x_i, y_i)\}\{i = 1, ..., (n_+ + n_-)\}$ where $x_i \in X$, $y_i \in Y = \{-1, +1\}$, where n_+ and n_- denote the number of positive and negative training samples respectively. Given a set of weak learners (features) $\hat{\mathcal{F}} = \{\mathfrak{h}_j\}_{j \in M}$, with $M = \{1, 2, ..., m\}$ the total number of weak learners, that can assign a given example a corresponding label, *i.e.*, $\mathfrak{h} : x \rightarrow y$, Discrete Adaboost constructs a strong classifier of the form $\mathcal{H}(x) = \sum_{t=1}^{T} \alpha_t \mathfrak{h}_t(x)$ with $sign(\mathcal{H}(x))$ determining the class label. The t indexes connote the sequence of the weak learners and this specific classifier has a total of T weak learners. The specific weak learner to use at each iteration of this boosting algorithm and the associated weighting coefficients, α_t, are derived minimizing the exponential loss, which provides an upper bound on the actual 1/0 loss [35]. The complete algorithm for training a per node classifier with Discrete AdaBoost is can be found in [35].

5.2. Other Approaches

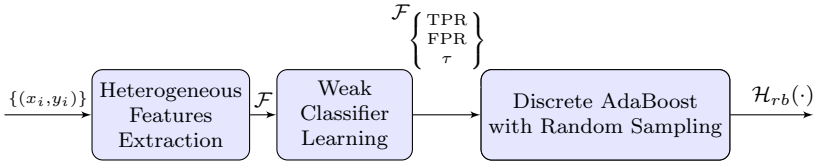
To investigate the advantages and disadvantages of the above presented BIP based framework, we also consider other simpler variants that are essentially derived by altering some of the constituent components in the BIP based framework. In doing so, we present four different classifier learning approaches, namely: a Pareto-Front with AdaBoost, AdaBoost with random feature sampling, computation time weighted AdaBoost, and hierarchically arranged cascade. Their block diagrams are depicted in figures 9a, 9b, 9c, and 10 consecutively.

5.2.1. Pareto-Front with AdaBoost

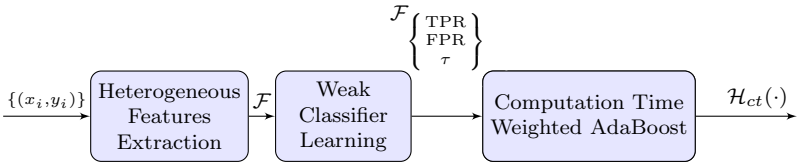
In this mode, a cascade node is trained via a combination of Pareto-Front analysis and discrete AdaBoost. First, the presented Pareto-Front analysis is used to extract a subset of non-dominated features $\tilde{\mathcal{F}}$ from $\mathcal{F}$, which are in turn used by discrete AdaBoost to build a strong nodal classifier $\mathcal{H}_{pb}(\cdot)$ as shown in figure 9a. This is simply jumping the BIP step in the framework presented in section 5.1 and directly using AdaBoost with the Pareto optimal



(a) Pareto-Front with AdaBoost.



(b) Random feature sampling with discrete AdaBoost.



(c) Computation time weighted AdaBoost.

Figure 9: Block diagrams of three other nodal strong classifier learning approaches.

features. The final cascade detector trained using this scheme is referred with the label *Pareto+AdaBoost*.

5.2.2. AdaBoost with Random Feature Sampling

Using AdaBoost directly on the extracted feature set, without any prior feature reduction, is the most classical detector learning scheme. It has been applied by various researchers following the pioneering work of Viola and Jones [43]. For a given cascade node, the node trains a nodal strong classifier using AdaBoost iteratively. On each iteration, AdaBoost selects a single feature that minimizes the weighted error over the training samples, assigns a proper weight to the associated weak classifier, and adds it to the ensemble. This entails evaluating the error term, ϵ_j, for all the weak classifiers in the pool. Given that we have a total of 1,448,374 weak classifiers in our pool, exhaustive search is not feasible. Therefore, we randomly sample a total number of $\mathcal{R}_s$ weak classifiers (in accordance with the relative proportion of features from each family) on each AdaBoost iteration and select the one

that minimizes the classification error. The nodal strong classifier trained with this scheme is denoted as $\mathcal{H}_{rb}(\cdot)$ – figure 9b – and the corresponding cascade labeled as *Random+AdaBoost*.

5.2.3. Computation Time Weighted AdaBoost

This scheme is quite similar to the above discussed variant based on random sampling and discrete AdaBoost (subsection 5.2.2) and has appeared in our previous work [22]. We use a modified discrete AdaBoost that not only takes detection performance, *i.e.*, minimize ϵ_j for selecting the best feature, but rather selects the one that minimizes a multiplicative term composed of feature computation time and detection error:

$$\mathfrak{h}_t = \arg\min_{\mathfrak{h}_j \in \mathcal{F}} \tilde{\tau}_j * \epsilon_j \tag{4}$$

$$\tilde{\tau}_j = \frac{\tau_j^{\beta}}{\sum_l \tau_{max,l}^{\beta}} \tag{5}$$

where $\tilde{\tau}_j$ denotes a smoothed normalized feature computation time. $\beta \in [0, 1]$ is an exponential smoothing coefficient. $\tau_{max,l}$ denotes the maximum computation time registered within each distinct feature pool family, *i.e.*, $l \in \{Haar, EOH, CSLBP, CSS, HOG\}$.

The computation time associated with each feature, $\tau_j = \tau_{p,j} + \tau_{e,j}$, is not constant (consequentially $\tilde{\tau}_j$ changes too). The exact value evolves during the classifier learning stage. It changes in two cases. The first is when a feature that has already been selected is considered in future cascade nodes, and the second is when a feature from the same family gets selected. In the prior case, the computation time of the selected feature is replaced by a constant time, τ_0, in future references which accounts for only memory access. In the latter case, the computation time for all of the features in the same family gets affected, specifically, the time associated with the pre-processing stage, $\tau_{p,j}$, is set to zero for all the features in that family. This is logical and is done to favor features of the same family. For example, if a Haar feature is selected, it will be better to consider another Haar feature so the integral image computation can be done once for the area spanned by the two features, rather than considering another feature from a different family which will require a different pre-processing step. This way the computation time of the features within the same family will be levied contributing to speed

up. Accordingly, the normalized computation time of all affected features is updated.

Algorithm 2 Computation Time Weighted AdaBoost

procedure TRAIN_ADABOOST($\mathcal{F}$, $\{(x_i, y_i)\}_{i \in N}$)
  Initialize: $D_1(i) = \dfrac{1}{(n_+ + n_-)}$
  for t = 1,2,...,T **do**
    · Find the best weak learner $\mathfrak{h}_t$:
      → $\mathcal{F}_r \leftarrow$ randomly sample $\mathcal{R}_s$ features from $\mathcal{F}$
      → **Compute** $\tilde{\boldsymbol{\tau}}_{\boldsymbol{j}} = \dfrac{\boldsymbol{\tau}_{\boldsymbol{j}}^{\boldsymbol{\beta}}}{\sum\limits_{\boldsymbol{i}} \boldsymbol{\tau}_{\boldsymbol{max,i}}^{\boldsymbol{\beta}}}$
      → $\boldsymbol{\mathfrak{h}_t = \underset{\mathfrak{h}_j \in \mathcal{F}_r}{\arg\min}\, \tilde{\tau}_j * \epsilon_j}$ where $\epsilon_j = \sum\limits_{i=1}^{n_+ + n_-} D_t(i)[y_i \neq \mathfrak{h}_j(x_i)]$
    · Compute weak learner weight: $\alpha_t = \dfrac{1}{2} \ln \dfrac{1 - \epsilon_t}{\epsilon_t}$
    · Update data weight distribution: $D_{t+1}(i) = \dfrac{D_t(i) \exp(-\alpha_t y_i \mathfrak{h}_t(x_i))}{Z_t}$
  end for
end procedure

This modification enables AdaBoost to select the feature (weak learner) that offers a compromise between computation time and detection error. This is detailed in algorithm 2 (main modifications on the classical one are shown in bold typeface). Again, due to the huge number of features, an exhausting search is not feasible and hence $\mathcal{R}_s$ randomly sampled features are used, as in section 5.2.2. The nodal strong classifier trained with this scheme is referred as $\mathcal{H}_{ct}(\cdot)$, see figure 9c. The cascade trained with this learning algorithm is referred as *CTWeightedAdaBoost*.

5.2.4. Coarse-to-Fine Hierarchical Arrangement

In this approach, we want to investigate the performance of a cascade detector when it is manually tailored to use specific features in specific order in coarse-to-fine hierarchy. Taking detection and computation time into consideration, the initial cascade nodes will use computationally cheap features, while trailing nodes would take on incrementally complex features (similar to [25, 31]). With the help of the feature computation time characteristics highlighted in table 1, a simple cascade is envisaged with the first G nodes using Haar like features, the next EOH features, then CS-LBP, followed by

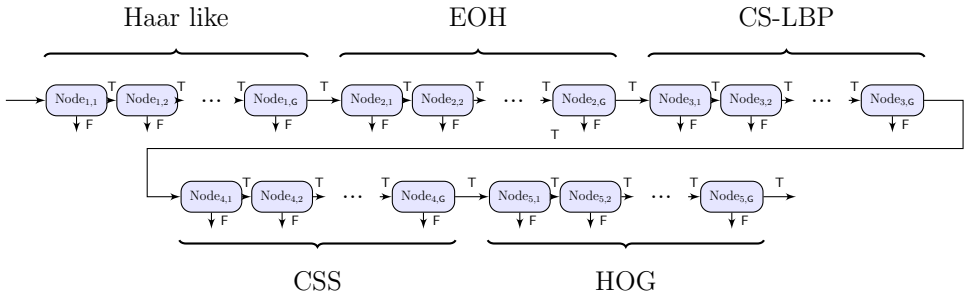


Figure 10: Coarse-to-fine hierarchically arranged sequence of cascade nodes.

CSS, and finally HOG features as depicted in figure 10. On each node, discrete AdaBoost with random node specific feature sampling is utilized to learn the nodal strong classifier, denoted $\mathcal{H}_{mb}(\cdot)$. Hence, at the node level heterogeneous feature pool is no more considered, it is rather a single feature family. But, globally, the entire cascade indeed considers heterogeneous features. The cascade node trained in this manner is labeled as *Hierarchical+AdaBoost*.

In this section, the BIP based and several other alternative nodal cascade classifier learning paradigms have been presented. They are the *BIP+AdaBoost*, *Pareto+AdaBoost*, *Random+AdaBoost*, *CTWeightedAdaBoost*, and *Hierarchical+AdaBoost*. Each node k of the cascade is trained to fulfill a stipulated nodal TPR_k and FPR_k detection performance constraints. In all cases, the cascade construction starts with all positive training samples and a subset of the negative training samples (equivalent to the positive ones) to learn the set of relevant features and classifiers for the initial cascade node, following the learning approach specified for each method. Once this is done, all negative training samples in the dataset are tested with it. All those that get classified correctly are rejected while all those labeled as positive samples (false positives) are retained along with the positive samples for training the following nodes. This step is repeated until all negative training sample are exhausted. This data mining technique makes it possible to use vast number of negative training samples.

6. Experiments and Results

In this section the different experiments carried out to investigate the performance of the investigated approaches and obtained results are presented. Principally, five different cascade detector training approaches – based on five differing nodal classifier learning schemes – using heterogeneous pool of features are presented. The evaluation aims to analyze the pros and cons the presented four different classifier learning schemes referred as: *BIP+AdaBoost*, *Pareto+AdaBoost*, *Random+AdaBoost*, *CTWeightedAdaBoost*, and *Hierarchy+AdaBoost*.

6.1. Visual Datasets

All experiments are carried out using two datasets: A proprietary dataset refereed as the Ladybug dataset, and the public INRIA person dataset.

6.1.1. Ladybug Dataset

The Ladybug dataset is a custom compiled dataset using images acquired with the *Ladybug2* camera in our robotic laboratory. This dataset features images of people acquired in a very cluttered indoor environment. The *Ladybug2* (figure 11a), manufactured by Point Grey Inc[33], is a polydioptric camera that provides real omni-directional view without pronounced geometric, resolution, and/or illumination artifacts. It is a spherical omni-directional camera system that has six cameras mounted in such a way to view more than 75% of a full sphere. Each camera has a maximum resolution of 1024×768 pixels resulting in a 3500×1750 pixels stitched high resolution panoramic image (figure 11b).

(a) (b)

Figure 11: *Ladybug2* camera and a corresponding stitched image.

This dataset consists of two distinct sets. The first one, referred as the training set, consists of 1990 positive samples (original and mirrored version) annotated by hand and scaled to a 128×64 pixels window. It also

contains 58 person free full resolution images acquired from our robotic and other rooms in the laboratory. A total of 488,992 negative windows of 128×64 pixels are uniformly sampled from these person free images. The second set used for testing purposes – hence, called the test set – contains 1,000 original and mirrored manually cropped positive samples of 128×64 pixels and 41 person free images, out of which 319,653 negative windows are uniformly sampled. Illustrative positive samples are shown in figure 12a. This dataset is made publicly available and can be downloaded from `http://homepages.laas.fr/aamekonn/dataset/ladybug/`.

(a) Ladybug dataset

(b) INRIA public dataset

Figure 12: Sample positive images from the Ladybug and INRIA datasets.

6.1.2. INRIA Person Dataset

The INRIA person dataset, introduced by Dalal and Triggs[5], is the most important and widely used public dataset for benchmarking purposes in people/pedestrian detection works. The dataset is divided in two formats, a format which contains original images (full images with people in natural scenes) with corresponding annotations and a second format which contains cropped positive images and people free negative images.

This format consists of both training and testing sets. The training set features 2416 cropped positive instances (originals and mirrored versions) and 1218 images free of persons. The cropped instances have a resolution of 160×96. But, the actual size of pedestrians bounding is 128×64. The extra

padding is provided to minimize the border effect. In this work, a total of 2.55×10^6 negative cropped instances are generated from these person free images. The test set contains 1132 positive instances and 453 person free images. Similarly, 2×10^6 cropped negative windows are uniformly sampled from the person free images during testing. Sample cropped positive images from this dataset are shown in figure 12b.

6.2. Evaluation Metrics

To evaluate our trained detectors, we use the Per Window (PW) evaluation scheme. In addition, we also define a metric which we call Average Speed Up (ASU) to characterize the speed gain with respect to a known benchmark.

6.2.1. PW Evaluation: Detector Error Trade-off (DET)

As the name clearly suggests, the PM method determines performance based on cropped positive and negative image windows. This approach isolates classifier performance from overall detection system, thus, making it ideal for characterizing classifier performance[9]. A widely used metric in this category is Detection Error Trade-off (DET). This metric depicts DET curve with miss rate (MR) versus False Positives Per Window (FPPW) on a log-log scale[5], where $MR = 1 - TPR = 1 - \frac{TP}{TP+FN}$ and $FPPW = \frac{FP}{FP+TN}$. This metric is principally used to report comparative results of the various investigated detectors.

6.2.2. Average Speed Up (ASU)

We define the ASU criterion to compare the performance of a trained detector in terms of computation time. Recall that for a cascade detector the average computation time for a given candidate window is affected by the FPR of each node. Let K be the total number of nodes in the cascade, FPR_k be the false positive rate and ζ_k be the total computation time of the k^{th} node during detection. Assuming the nodal FPR characteristics hold on a generic input image, the average time spent on a test candidate window, ζ_{av}, can be estimated as $\zeta_{av} = \sum_{k=1}^{K} (\prod_{z=0}^{k-1} \mathrm{FPR}_z) \zeta_k$.

$$\mathrm{ASU} = \frac{\zeta_{HOG}}{\zeta_{av}} \tag{6}$$

Using Dalal and Triggs[5] detector – which takes ζ_{HOG} per candidate window – as a reference, the ASU over it is determined using equation 6.

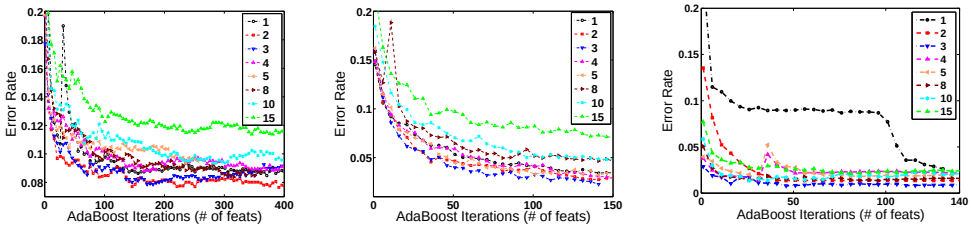
Consequently, the ASU values reported henceforth are with respect to Dalal and Triggs detector.

6.3. Implementation Details and Validation

Recall that the cascade node training is governed by two parameters: the nodal FPR_k and TPR_k. In all the experiments a nodal TPR_k value of 1.0 and FPR_k of 0.5 is used unless specified otherwise (the exception is the variant discussed in section 6.4.2). During training the TPR and FPR values exhibited by the weak classifiers or cascade node is determined on a separate validation set. Actually, the training dataset is initially divided into a 60% training and a 40% validation set; the new training set is used to train the weak classifiers and nodal classifiers whereas the validation set is used to determine detection performance exhibited. Once all feature selection is done, the node is retrained using the complete training set. This is applied when considering both the Ladybug and INRIA datasets (section 6.1).

In the following subsections, the validation steps taken to determine free parameters that are crucial for the performance of the different presented nodal classifier training schemes are discussed. These parameters are: (1) the depth of the decision trees used ($\mathcal{D}$), (2) number of features randomly sampled ($\mathcal{R}_s$) with the *Random+AdaBoost* and *CTWeightedAdaBoost* strategies, and (3) the computation time smoothing coefficient (β) used in the *CTWeightedAdaBoost* variant. In all cases, a training and validation set from the Ladybug dataset is used.

6.3.1. Decision Tree Depth ($\mathcal{D}$)



(a) (b) (c)

Figure 13: Decision tree depth validation for (a) Haar like features – $\mathcal{D}_{haar}$, (b) EOH features – $\mathcal{D}_{eoh}$, and (c) CS-LBP features – $\mathcal{D}_{cslbp}$.

The depth of the decision trees for Haar like features ($\mathcal{D}_{haar}$), EOH ($\mathcal{D}_{eoh}$), and CS-LBP ($\mathcal{D}_{cslbp}$) features are validated as follows. Using only a single

cascade node, a strong classifier is trained using AdaBoost with each individual feature families, and its performance on a validation set analyzed. Multiple runs are performed using different decision tree depths. On each AdaBoost iteration, only $2,000$ randomly sampled features are used to limit the validation time to a reasonable duration. For Haar like, EOH, and CS-LBP features, the error rate on the validation set as a function of the AdaBoost iteration for different decision tree depths is shown in figures 13a, 13b, and 13c respectively. Based on this, decision tree depths of $\mathcal{D}_{haar} = 2$, $\mathcal{D}_{eoh} = 3$, and $\mathcal{D}_{cslbp} = 3$ are used. Computing Fisher LDA weights, for CS-LBP features, per each node makes the classifier over-fit on the training set with deteriorated performance on the validation set. Hence, the Fisher LDA weights computed at the first node are used throughout the cascade by learning only new decision trees.

6.3.2. Number of Randomly Sampled Features ($\mathcal{R}_s$)

At each AdaBoost iteration in *Random+AdaBoost* and *CTWeightedAdaBoost* variants, we use a randomly selected subset of features (as is commonly done, *e.g.*, in[55, 50]). According to Scholkopf and Smola[36](pp. 180), given a set of samples, it can be guaranteed to sub sample amongst the best r_s percentage of estimates with a probability p by randomly sampling a sub sample of size $\frac{log(r_s)}{log(p)}$. This reduced set will do as well as considering the entire set with a probability p. In our case, to select amongst the best 5% features with a 99% probability, we need to sample a total of $\frac{log(0.05)}{log(0.99)} \approx 299$ samples. In our implementation we use 3000 features which is way above the suggested number of samples and guarantees inclusion of relevant features with a high probability. Hence, $\mathcal{R}_s = 3000$.

Table 2: Values of different parameters obtained with the validation step.

Parameter	Description	Value
TPR_k	required True Positive Rate for node k (fixed case)	1.0
FPR_k	required False Positive Rate for node k (fixed case)	0.5
$\mathcal{D}_{haar}$	decision tree depth for Haar like features	2
$\mathcal{D}_{eoh}$	decision tree depth for EOH features	3
$\mathcal{D}_{cslbp}$	decision tree depth for CS-LBP features	3
$\mathcal{R}_s$	number of randomly sampled features	3000
β	computation time smoothing coefficient	0.2
G	number of cascade nodes per feature family (for Hierarchical + AdaBoost)	3 (ladybug training) 5 (inria training)

6.3.3. Computation Time Smoothing Coefficient (β)

The computation time smoothing exponential factor, β, used in the time weighted AdaBoost is determined empirically through a validation step. The *CTWeightedAdaBoost* is used to learn a single nodal cascade using different β values on a subset of the training set. Then the classification errors on a validation set and the conglomerated computation time of the trained node is determined to select the best value that offers a good trade-off. Figure 14 shows the validation result plots for different values of β. Clearly, higher β reduces smoothing, in effect, features with low computation time dominate improving speed but with poor detection performance. Lower values favor complex features. As a compromise, a β value of 0.2 is used to train the final cascade classifier in this scheme.

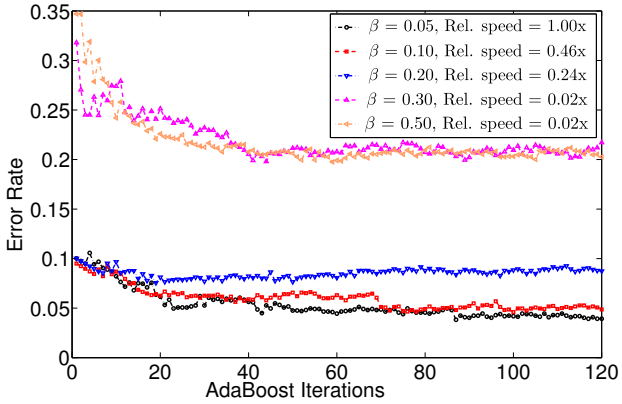


Figure 14: Error rate on a validation set using the computation time weighted AdaBoost trained with different β values.

The different parameters tuned in the validation step are summarized in table 2.

6.4. Results

The results corresponding to all experiments are reported in this section categorized under each dataset.

6.4.1. Ladybug Dataset

We train five different cascade detectors (see section 5) using its training set and test on its test set. Since this dataset is composed of cropped positive

samples, we use the PW evaluation to determine detection performance. The main obtained results are depicted in figure 15 and summarized in table 3. Clearly *Pareto+AdaBoost* and *Hierarchy+AdaBoost* result in the best detection performance, 2.9% MR, followed by Dalal and Triggs detector trained on this dataset, 3.0%, at 10^{-4} FPPW. *CTWeightedAdaBoost* shows the lowest detection performance with a 10% MR at 10^{-3} FPPW, but it manages to learn a detectors that is 1.8× faster than Dalal and Triggs HOG. The *Hierarchy+AdaBoost* detector achieves a 8.4× speed improvement. In terms of detection, *BIP+AdaBoost* trails behind *Random+AdaBoost* with marginal loss. But, the most important result to notice is that *BIP+AdaBoost* results in a drastic 42.7x speed up over Dalal and Triggs with only a 7% loss in MR at 10^{-4} FPPW. The main reason for this speed up is that *BIP+AdaBoost* systematically uses cheap features in the initial stages of the cascade and only starts using computationally expensive features at later stages. The trained classifier has 10 cascade nodes with CSS features initially appearing at the 6^{th} node and HOG at the final stage; figure 16 depicts this showing the selected features at some of the nodes.

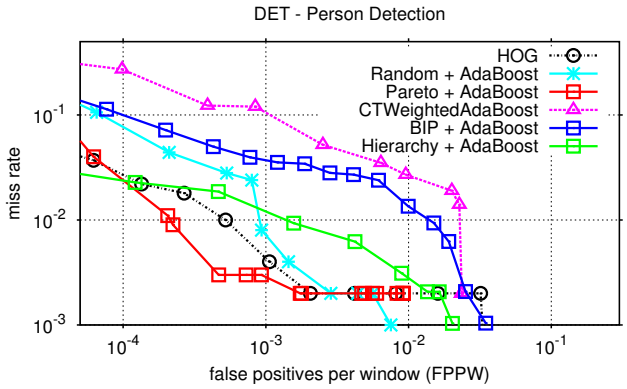
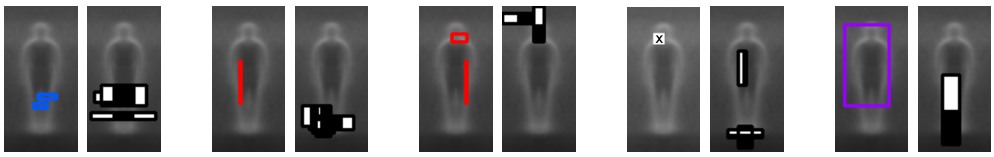


Figure 15: DET of different detectors trained and tested on the Ladybug dataset.

Apparently, *Pareto+AdaBoost* and *Random+AdaBoost* result in worsened speeds. This is because AdaBoost always privileges the most discriminant feature, irrespective of computation cost, from the pool of features passed to it, and both Pareto-Front extraction and random sampling are likely to pass such kind of complex features. Consequently, the set of features selected in the first node result in a conglomerate that is effectively computationally demanding than Dalal and Triggs detector. *CTWeightedAdaBoost*



(a) node-0 (b) node-1 (c) node-2 (d) node-5 (e) node-9

Figure 16: Illustration of the different features selected on different nodes of cascade (superimposed on an average human gradient image) of *BIP+AdaBoost* trained on Ladybug dataset. Black and white rectangular regions are Haar features, red for EOH, blue for CS-LBP, crossed white boxes for CSS, violet HOG features.

Table 3: Summary of the cascade detector trained on the Ladybug dataset. Miss rate is reported at 10^{-4} FPPW.

Detector	Feature Proportion					MR	ASU
	Haar	CS-LBP	CSS	EOH	HOG		
Dalal and Triggs[5]	–	–	–	–	100%	3.0%	1.0x
Pareto + AdaBoost	10.7%	0.0%	0.0%	0.0%	83.7%	2.9%	0.7x
CTWeightedAdaBoost	53.3%	33.3%	0.0%	10.0%	3.3%	25.0%	1.8x
Random + AdaBoost	51.6%	6.2%	1.5%	36.0%	4.7%	8.0%	0.6x
Hierarchy + AdaBoost	17.8%	67.4%	7.6%	5.1%	2.1%	2.9%	8.4x
BIP + AdaBoost	54.3%	8.6%	8.5%	25.7%	2.8%	10.0%	42.7x

improves upon this by selecting a slightly less complex feature. The intuition of using cheaper features initially followed by complex features, the *Hierarchical+AdaBoost* detector, pays off both in terms of detection performance and computation time improvement. But, the computation time improvement is much lower than the *BIP+AdaBoost* variant. Figure 17 shows the features selected in the initial cascade of the four trained detectors. The results for *Hierarchy+AdaBoost* are not shown as they convey no useful information – containing 14 Haar like features that clutter the window. *Random+AdaBoost*, *Pareto+AdaBoost*, and *CTWeightedAdaBoost*, have HOG features in this node contributing to reduced speed; on the contrary, for *BIP+AdaBoost*, only CS-LBP and Haar features are used. These result are obtained using a fixed nodal FPR_k of 0.5 for all constructed nodes and the obtained results are very precise that altering the FPR is not necessary.

The proportion of features present from each family is also consistent with

(a) Random+AdaBoost (b) Pareto+AdaBoost (c) CTWeightedAdaBoost (d) BIP+AdaBoost

Figure 17: The features selected and used in the first node of the cascade under the different learning approaches with the Ladybug dataset superimposed on an average human gradient image. Black and white rectangular regions show Haar features, blue for CS-LBP, crossed white boxes represent CSS features and their position indicates the reference block, and finally, violet shows the spatial region spanned by the concatenated HOG blocks.

the underlying training scheme adopted (see Table 4). Both *CTWeightedAdaBoost* and *BIP+AdaBoost* emphasize on computation time, accordingly, they have the highest proportion of Haar features in their trained model. They also have the least proportion of HOG features taking only 3.3% and 2.8%, respectively, of the total proportion of features. *Pareto+AdaBoost* favors complex features with superior detection performance which explains the 83.7% HOG features presence.

6.4.2. INRIA Dataset

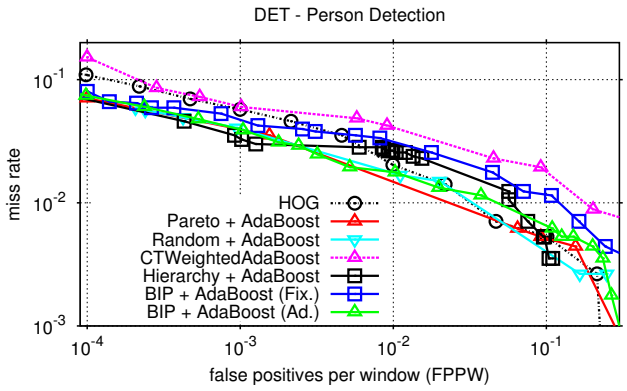


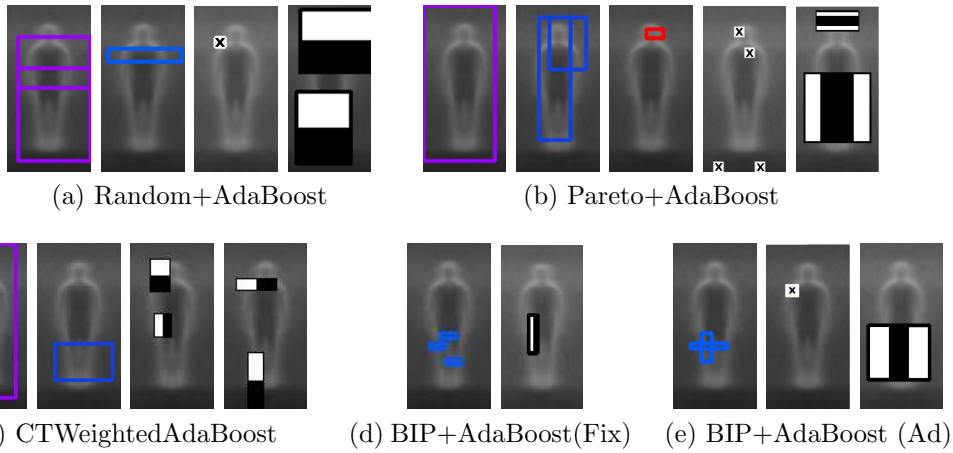
Figure 18: DET of different detectors trained and tested on the INRIA dataset.

Similar results obtained for the INRIA dataset are shown in figure 18 and summarized in table 4. As this dataset is challenging, two variants of the *BIP+AdaBoost* classifier are trained. In the first case, a fixed nodal FPR_k of 0.5 is used for all nodes, called *BIP+AdaBoost (Fix)*. In the second case, an adaptive FPR is employed which starts at 0.3 in the initial stage and continues training nodes, whenever a solution for the BIP optimization does not exist, this constraint is relaxed/incremented by 0.1 and the procedure continues from that node likewise until all negative samples are depleted. This is called *BIP+AdaBoost (Ad)*. The best detection results at 10^{-4} FPPW are obtained by the *Random+AdaBoost* and *Hierarchy+AdaBoost* variants (*Pareto+AdaBoost* is edged out marginally). But, this time both variants of *BIP+AdaBoost* beat Dalal and Triggs detector at 10^{-4} by more than 2%. On top of this, the *BIP+AdaBoost(Fix)* achieves a 15.6x speed up while that of *BIP+AdaBoost (Ad)* trails with a 9.22x speed up. As expected the *Hierarchy+AdaBoost* also results in improved speed (an ASU of 4.05×). *Random+AdaBoost*, *Pareto+AdaBoost*, and *CTWeightedAdaBoost* variants on the other hand result in increased computation time (even with respect to [5]). The reason is all these three start off with complex features in the initial nodes. Figure 19 shows the features selected in the first node of all trained detector variants. Only the ones employing BIP do not have HOG features in the initial stage. Even though the *CTWeightedAdaBoost* variant does not result in a significant boost in speed, it is twice as much faster as its random counterpart (*Random+AdaBoost*) with marginal loss in miss rate. The explicit computation time consideration does help even in this case.

Table 4: Summary of the cascade detector trained on the INRIA datasets. Miss rate is reported at 10^{-4} FPPW.

Detector	Feature Proportion					MR	ASU
	Haar	CS-LBP	CSS	EOH	HOG		
Dalal and Triggs[5]	–	–	–	–	100%	11.0%	1.0x
Pareto + AdaBoost	42.8%	14.5%	7.8%	25.6%	9.3%	7.0%	0.4x
Random + AdaBoost	26.3%	10.8%	3.7%	53.5%	5.6%	6.0%	0.4x
CTWeightedAdaBoost	86.7%	9.1%	2.4%	0.0%	3.9%	14.6%	0.8x
Hierarchy + AdaBoost	26.6%	9.7%	5.7%	24.9%	33.%	6.8%	4.05x
BIP + AdaBoost (Fix)	60.4%	10.8%	8.0%	9.7%	11.0%	8.0%	15.6x
BIP + AdaBoost (Ad)	55.0%	14.6%	8.1%	9.3%	13.0%	7.4%	9.22x

Concerning the *BIP+AdaBoost* variants, as the initial FPR constraints are stringent on the *BIP+AdaBoost (Ad)* variant, it favors relatively discriminant features with increased computation time. This contributes to its superior detection performance, over *BIP+AdaBoost (Fix)*, throughout the FPPW range shown in figure 18. Observe in table 4, there are more proportion of Haar like features (5.4% more) and less proportions of HOG features (2.0% less) in the fixed variant compared to the adaptive variant which results in the increased speed.

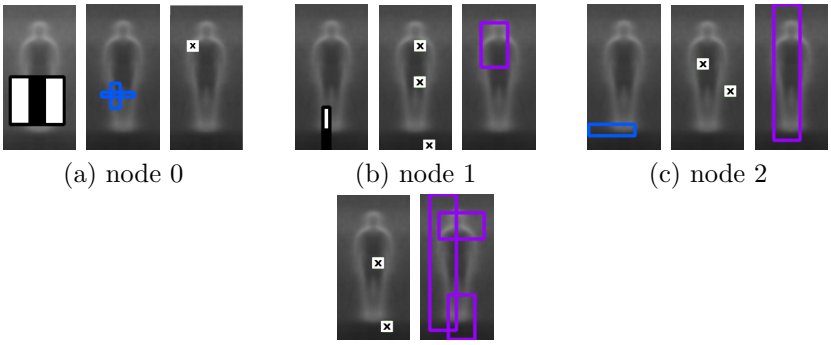


(a) Random+AdaBoost (b) Pareto+AdaBoost

(c) CTWeightedAdaBoost (d) BIP+AdaBoost(Fix) (e) BIP+AdaBoost (Ad)

Figure 19: The features selected and used in the first node of the cascade trained on the INRIA dataset superimposed on an average human gradient image. Black and white rectangular regions are Haar features, blue for CS-LBP, red for EOH, crossed white boxes for CSS, violet HOG features.

Figure 20 illustrates a few of the selected features overlaid on an average human gradient image for *BIP+AdaBoost (Ad)*. Observe that all selected features capture discriminant facets of people. Figure 21 shows histogram of the selected features, with relative proportions, for the first 9 nodes of both the fixed and adaptive variants. Clearly, the fixed variant initially uses cheaper features and increases along the cascade both in number and complexity. On the contrary, for the variable variant, complex features appear in the initial nodes and increase in number along the cascade.



(a) node 0 (b) node 1 (c) node 2

(d) node 8

Figure 20: Sample depictions of the heterogeneous features selected at different nodes of the cascade *BIP+AdaBoost (Ad)* trained on the INRIA dataset using an adaptive FPR. Black and white rectangular regions are Haar features, blue for CS-LBP, crossed white boxes for CSS, violet HOG features.

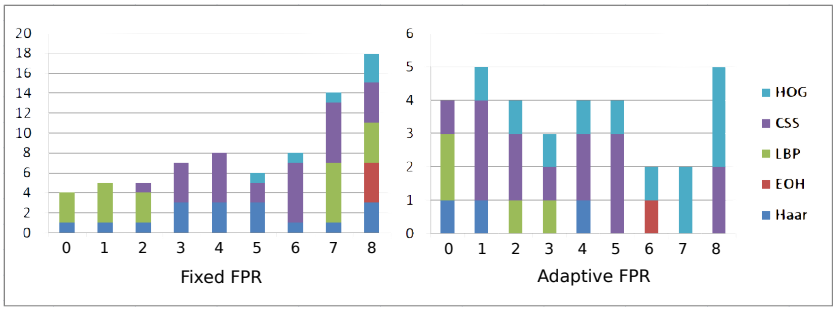


Figure 21: Histogram of selected features in the first 9 nodes of the model trained on the INRIA dataset using both fixed FPR of 0.5 and adaptive FPR.

7. Discussions

The features presented in this work capture varying cues relevant to people in an image. They are highly variable in terms of discriminative ability as well as computation time. Given this kind of feature pool with diverse characteristics, extra care should be given to the way these features are mined to build a people detector. This work favors a sparser solution by employing feature selection to reduce the set to a few performant ones.

Settling on feature selection, to mine relevant features out of the pool, we

have investigated five different approaches all based on the popular AdaBoost with cascade of nodes detector. The approaches are: *Random+AdaBoost*, which uses discrete AdaBoost to learn a strong nodal classifier by randomly sampling $\mathcal{R}_s$ features on each boosting iterations and adding the best one to the ensemble; *Pareto+AdaBoost*, which uses Pareto-Front analysis to retain only non-dominated features with respect to TPR, FPR, and computation time, and then builds a nodal strong classifier with discrete AdaBoost and the retained subset of features; *CTWeightedAdaBoost*, which makes use of a modified AdaBoost that selects the best features using the classification error weighted by a computation time measure to enable AdaBoost to give consideration for feature computing speed; *Hierarchy+AdaBoost*, which takes on a coarse-to-fine hierarchical arrangement approach to use computationally cheap features in the initial nodes of the cascade and then consider incrementally complex features in subsequent cascade nodes; and finally, *BIP+AdaBoost* which uses a discrete optimization formulation based on BIP to retain the minimum number of features that fulfill the stipulated detection performance with the minimum combined computation time. All five approaches have been evaluated on proprietary and public datasets thoroughly and compared to the state-of-the-art. In terms of detection, all five achieve between 6.0% and 14.6% miss rates at 10^{-4} FPPW on the INRIA dataset which confirms that considering heterogeneous features is relevant.

The *Random+AdaBoost* variant avoids the need to iterate through all features in the pool, at each boosting iteration, which would have otherwise made the training non-realistic. As long as the number of randomly sampled features are high, it will lead to comparable results as the exhaustive search. In terms of detection, this approach is expected to lead to the best (compared to the four variants) result as it always picks the best discriminative features iteratively. The downside of this approach is, reflecting the properties of AdaBoost, it always selects features based on classification performance blind to feature computation time. If there are two competing features with only a slight difference in classification error but big difference in computation time, the minimum error feature will be selected. This characteristics is exemplified in the evaluations as it leads to the worst detector speed, 0.4× Dalal and Triggs HOG, with amongst the best detection performance, *e.g.*, 6.0% MR at 10^{-4} FPPW on the INRIA dataset, in all cases.

To overcome the shortcomings of *Random+AdaBoost*, we investigated *CTWeightedAdaBoost* variant. In the experimental results, this approach achieves approximately 2× and 3× as much faster detector compared to the

Random+AdaBoost variant on the Ladybug and INRIA trained models respectively. But, the modification down plays its detection performance leading to reduced detection rate. For example it achieves a 5.6% MR reduction at 10^{-4} FPPW compared to *Random+AdaBoost* on the INRIA evaluation. The third investigated scheme, *Pareto+AdaBoost* variant, avoids the exhaustive search that needs to be done by AdaBoost and yet is guaranteed to pass on the most performant features. This is clearly seen by the low miss rate it exhibits on the test cases, 2.9% and 7.0% MR at 10^{-4} FPPW on the Ladybug and INRIA dataset evaluations respectively. But, again as long as computationally intensive discriminant features exist in the selection, which is actually the case as observed in the experiments, AdaBoost is bound to greedily favor the discriminant ones leading to computationally demanding detector. These remarks are seen on the evaluation results, for example on the INRIA test set, it achieves the second highest detection rate, 7.0% miss rate at 10^{-4} FPPW, with the least frame rate, more than twice slower than Dalal and Triggs HOG.

The fourth scheme termed as *Hierarchy+AdaBoost* uses a unique family of features in each node. It starts with computationally cheap features, the Haar like features, and continues until HOG, training G number of nodes with each feature family, a kind of coarse-to-fine hierarchical detector. This simple approach does indeed lead to both improved detection and speed compared to Dalal and Triggs HOG – a 4.05× speed gain and a 3.2% improvement on MR at 10^{-4} FPPW. But, as it has been demonstrated, the speed gain is inferior to the BIP based approach.

The main approach presented, *BIP+AdaBoost*, makes explicit optimization to select the features that achieve the required detection performance with the minimum possible computation time. The two modes investigated in this scheme are learned using a fixed nodal FPR and an adaptive nodal FPR. Both variants result in a detector that is most considerate as both detection and speed aspects are taken into account to come up with the best compromise. The *BIP+AdaBoost (Fix)* variant for example achieves a 10.0% and 8.0% MR at 10^{-4} on the Ladybug and INRIA datasets respectively. It contains significant proportions (more than 54% on the Ladybug model and more than 60% on the INRIA model) of Haar features and less proportions of the costly features, *e.g.*, only 2.8% and 11% HOG features in both datasets respectively. This helps it achieve a 42.7× and 15.6× speed up over Dalal and Triggs HOG using the Ladybug and INRIA trained models respectively. Its adaptive variant trained on the INRIA dataset improves the detection

further achieving a 7.4% MR at 10^{-4} FPPW with a 9.22× speed up over Dalal and Triggs HOG. The trained detector has more proportion of Haar features (55.0%) and less proportion of HOG features (13.0%). Hence, it can be safely concluded that the BIP based detector variants are the most considerate ones as they work on both detection and speed aspects to come up with the best compromise.

Another advantage of the BIP based framework is its flexibility with respect to computational resource constraints and detection requirement. On any dataset, the stipulated detection parameters used during training (nodal TPR_k and FPR_k) can either be made stringent or relaxed to learn a model that can either consume more or less computational resources respectively, giving explicit control on the detection vs speed trade off. This has been demonstrated with the adaptive and fixed detector variants trained on the INRIA dataset.

Additionally, another flexibility of the framework is its ability to adapt the complexity of the learned model to the challenge inherently present in the training dataset. The framework takes advantage of the underlying challenge manifested by the training dataset to furnish an appropriate detector model. For simpler datasets, it provides simpler model with increased frame rate, *e.g.*, the 42.7x improvement achieved with the Ladybug dataset trained model contrary to the 9.22x improvement achieved with the INRIA dataset. This quality would enable developing a detector that is suited for specific scene/domain that reflects on the detection challenge, *e.g.*, for indoor open environment (like the hall of a shopping mall) application that might feature less background clutter with upright people having less pose variability, with faster frame rates. Most of the detectors listed in the state-of-the-art do not have the ability to automatically change the complexity of the detector based on the dataset. As an example, consider Dalal and Triggs HOG [5], *HogLbp*[45], *LatSvm-V1*[10], which have a fixed size high-dimensional classifier that is always fixed irrespective of the dataset.

8. Conclusions

In this work, different strategies to train a people detector using heterogeneous pool of features have been investigated. Various experiments have been carried out to investigate the advantages and shortcomings of each strategy using proprietary and multiple public datasets. The obtained results ascertain that complementary heterogeneous features lead to improved detection

performance, and under explicit consideration of computation time, lead to improved frame rate as well. The different results also show the superiority of the proposed BIP based feature selection strategy. The proposed BIP strategy is quite capable in taking advantage of the diversity that exists in the feature pool from detection as well as speed perspectives. Further improved frame rates can also be achieved by parallelizing the trained model with the help of specialized hardwares like a Graphical Processing Unit (GPU).

In the near future, we plan to investigate ways to achieve more faster versions of the detector by focusing on implementation optimization and specialized accelerator hardwares. Additionally, we are in the process of integrating the proposed BIP based detector in a "tracking-by-detection" framework for multi-person tracking. This will be used in robotic applications to better identify the trajectory of each individual in the vicinity for acceptable human-aware robot navigation.

Acknowledgment

This work was supported by a grant from the French National Research Agency (ANR) under project RIDDLE with grant number ANR-12-CORD-0003.

References

[1] A. Angelova, A. Krizhevsky, and V. Vanhoucke. Pedestrian detection with a large-field-of-view deep network. In *IEEE International Conference on Robotics and Automation (ICRA'15)*, pages 704–711, May 2015.

[2] A. Angelova, A. Krizhevsky, V. Vanhoucke, A. Ogale, and D. Ferguson. Real-time pedestrian detection with deep network cascades. In *British Machine Vision Conference (BMVC'15)*, September 2015.

[3] R. Benenson, M. Mathias, T. Tuytelaars, and L. Van Gool. Seeking the strongest rigid detector. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'13)*, pages 3666–3673, June 2013.

[4] R. Benenson, M. Omran, J. Hosang, and B. Schiele. Ten years of pedestrian detection, what have we learned? In *Computer Vision - ECCV 2014 Workshops*, volume 8926 of *Lecture Notes in Computer Science*, pages 613–627, 2015.

[5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'05)*, San Diego, CA, USA, June 2005.

[6] N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. In *European Conference on Computer Vision (ECCV'06)*, pages 428–441, Graz, Austria, May 2006.

[7] P. Dollár, R. Appel, S. Belongie, and P. Perona. Fast feature pyramids for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1532–1545, Aug 2014.

[8] P. Dollár, Z. Tu, P. Perona, and S. Belongie. Integral channel features. In *British Machine Vision Conference (BMVC'09)*, pages 91.1–91.11, London,UK, September 2009.

[9] P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):743–761, 2012.

[10] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[11] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(7):179–188, 1936.

[12] D. Gerónimo, A. López, A. Sappa, and T. Graf. Survey of pedestrian detection for advanced driver assistance systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(7):1239–1258, 2010.

[13] D. Gerónimo, A. M. López, D. Ponsa, and A. D. Sappa. Haar wavelets and edge orientation histograms for on-board pedestrian detection. In *Iberian Conference Pattern Recognition and Image Analysis (IbPRIA'07)*, pages 418–425, Girona, Spain, June 2007.

[14] M. Heikkil, M. Pietikinen, and C. Schmid. Description of interest rregions with local binary patterns. *Pattern Recognition*, 42(3):425 – 436, 2009.

[15] S. Hussain and B. Triggs. Feature sets and dimensionality reduction for visual object detection. In *British Machine Vision Conference (BMVC'10)*, pages 1–10, Aberystwyth, UK, August 2010.

[16] L. Jourdheuil, N. Allezard, T. Chateau, and T. Chesnais. Heterogeneous adaboost with real-time constraints - application to the detection of pedestrians by stereovision. In *International Conference on Computer Vision Theory and Applications (VISAPP'12)*, pages 539–546, Rome, Italy, February 2012.

[17] K. Levi and Y. Weiss. Learning object detection from a small number of examples: The importance of good features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'04)*, pages 53–60, Washington, DC, USA, June 2004.

[18] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. In *IEEE International Conference on Image Processing (ICIP'02)*, pages 900–903, New York, USA, September 2002.

[19] J.-J. Lim, C. L. Zitnick, and P. Dollár. Sketch tokens: A learned mid-level representation for contour and object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'13)*, pages 3158–3165, Portland, OR, USA, 2013.

[20] Z. Lin and L. S. Davis. A pose-invariant descriptor for human detection and segmentation. In *European Conference on Computer Vision (ECCV'08)*, pages 423–436, Marseille, France, October 2008.

[21] S. Maji, A. Berg, and J. Malik. Classification using intersection kernel support vector machines is efficient. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08)*, pages 1–8, Anchorage, AL, USA, June 2008.

[22] A. A. Mekonnen, F. Lerasle, and A. Herbulot. Person detection with a computation time weighted adaboost. In *Advanced Concepts in Intelligent Vision Systems (ACIVS'13)*, pages 632–644, Poznan,Poland, October 2013.

[23] A. A. Mekonnen, F. Lerasle, A. Herbulot, and C. Briand. People detection with heterogeneous features and explicit optimization on computation time. In *International Conference on Pattern Recognition (ICPR'14)*, pages 4322–4327, Stockholm, Sweden, August 2014.

[24] K. Mikolajczyk, C. Schmid, and A. Zisserman. Human detection based on a probabilistic assembly of robust part detectors. In *European Conference on Computer Vision (ECCV'04)*, volume I, pages 69–81, Prague, Czech Republic, May 2004.

[25] A. Mogelmose, A. Prioletti, M. Trivedi, A. Broggi, and T. Moeslund. Two-stage part-based pedestrian detection. In *IEEE International Conference on Intelligent Transportation Systems (ITSC'12)*, pages 73–77, Anchorage, AK, USA, September 2012.

[26] Y. Mu, S. Yan, Y. Liu, T. Huang, and B. Zhou. Discriminative local binary patterns for human detection in personal album. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08)*, pages 1–8, Anchorage, AK, USA, June 2008.

[27] W. Nam, P. Dollar, and J.-H. Han. Local decorrelation for improved pedestrian detection. In *Advances in Neural Information Processing Systems (NIPS'14)*, pages 424–432, 2014.

[28] T. Ojala, M. Pietikinen, and D. Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, 29(1):51 – 59, 1996.

[29] S. Paisitkriangkrai, C. Shen, and A. van den Hengel. Strengthening the effectiveness of pedestrian detection with spatially pooled features. In *Computer Vision ECCV 2014*, pages 546–561. Springer International Publishing, 2014.

[30] S. Paisitkriangkrai, C. Shen, and J. Zhang. Fast pedestrian detection using a cascade of boosted covariance features. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(8):1140–1151, 2008.

[31] H. Pan, Y. Zhu, and L. Xia. Efficient and accurate face detection using heterogeneous feature descriptors and feature selection. *Computer Vision and Image Understanding*, 117(1):12 – 28, 2013.

[32] C. Papageorgiou and T. Poggio. A trainable system for object detection. *IJCV*, 38(1):15–33, 2000.

[33] Point Grey Inc. Ladybug2. `http://www.ptgrey.com/products/ladybug2/`, 2012. [Online; accessed 29-January-2013].

[34] T. J. V. Roy and L. A. Wolsey. Valid inequalities for mixed 0-1 programs. *Discrete Applied Mathematics*, 14(7):199–213, 1986.

[35] R. E. Schapire. The boosting approach to machine learning: An overview. *Lecture Notes in Statistics*, pages 149–172, 2003.

[36] B. Scholkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[37] W. R. Schwartz, A. Kembhavi, D. Harwood, and L. S. Davis. Human detection using partial least squares analysis. In *IEEE International Conference on Computer Vision (ICCV'09)*, pages 24–31, Kyoto, Japan, October 2009.

[38] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. Lecun. Pedestrian detection with unsupervised multi-stage feature learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'13)*, pages 3626–3633, Washington, DC, USA, 2013.

[39] M. Szarvas, A. Yoshizawa, M. Yamamoto, and J. Ogata. Pedestrian detection with convolutional neural networks. In *IEEE Intelligent Vehicles Symposium (IV'05)*, pages 224–229, Las Vegas, NV, USA, June 2005.

[40] D. Tang, Y. Liu, and T. Kim. Fast pedestrian detection by cascaded random forest with dominant orientation templates. In *British Machine Vision Conference (BMVC'09)*, pages 1–11, London, UK, September 2012.

[41] Y. Tian, P. Luo, X. Wang, and X. Tang. Pedestrian detection aided by deep learning semantic tasks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'15)*, pages 5079–5087, 2015.

[42] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler. Efficient object localization using convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'15)*, pages 648–656, 2015.

[43] P. A. Viola and M. J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[44] S. Walk, N. Majer, K. Schindler, and B. Schiele. New features and insights for pedestrian detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'10)*, pages 1030–1037, San Francisco, CA, USA, June 2010.

[45] X. Wang, T. Han, and S. Yan. An HOG-LBP human detector with partial occlusion handling. In *IEEE International Conference on Computer Vision (ICCV'09)*, Kyoto, Japan, October 2009.

[46] X. Wang, M. Yang, S. Zhu, and Y. Lin. Regionlets for generic object detection. In *IEEE International Conference on Computer Vision (ICCV'13)*, pages 17–24, Washington, DC, USA, 2013.

[47] C. Wojek and B. Schiele. A performance evaluation of single and multi-feature people detection. In *DAGM-Symposium*, pages 82–91, Munich, Germany, June 2008.

[48] L. A. Wolsey. Strong formulations for mixed integer programs: Valid inequalities and extended formulations. *Mathematical Programming*, 97(7):423–447, 2003.

[49] B. Wu and R. Nevatia. Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors. In *IEEE International Conference on Computer Vision (ICCV'05)*, pages 90–97, Beijing, China, October 2005.

[50] B. Wu and R. Nevatia. Optimizing discrimination-efficiency tradeoff in integrating heterogeneous local features for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08)*, Anchorage, AK, USA, June 2008.

[51] S. Zhang, C. Bauckhage, and A. Cremers. Informed haar-like features improve pedestrian detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'14)*, pages 947–954, June 2014.

[52] S. Zhang, R. Benenson, and B. Schiele. Filtered channel features for pedestrian detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'15)*, Boston, MA, USA, 2015.

[53] G. Zhao and M. Pietikainen. Dynamic texture recognition using local binary patterns with an application to facial expressions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):915–928, 2007.

[54] L. Zhao and C. Thorpe. Stereo-and neural network-based pedestrian detection. In *IEEE International Conference on Intelligent Transportation Systems (ITSC'99)*, pages 298–303, Tokyo, Japan, October 1999.

[55] Q. Zhu, M.-C. Yeh, K.-T. Cheng, and S. Avidan. Fast human detection using a cascade of histograms of oriented gradients. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'06)*, New York, NY, USA, June 2006.