Fast PRISM

Branch and Bound Hough Transform for Object Class Detection

Originally published in:
International Journal of Computer Vision 94(2), https://doi.org/10.1007/s11263-010-0342-x

Int J Comput Vis (2011) 94:175–197
DOI 10.1007/s11263-010-0342-x

Fast PRISM: Branch and Bound Hough Transform for Object Class Detection

Alain Lehmann · Bastian Leibe · Luc Van Gool

Received: 21 September 2009 / Accepted: 9 April 2010 / Published online: 28 April 2010

Abstract This paper addresses the task of efficient object class detection by means of the Hough transform. This approach has been made popular by the Implicit Shape Model (ISM) and has been adopted many times. Although ISM exhibits robust detection performance, its probabilistic formulation is unsatisfactory. The PRincipled Implicit Shape Model (PRISM) overcomes these problems by interpreting Hough voting as a dual implementation of linear sliding-window detection. It thereby gives a sound justification to the voting procedure and imposes minimal constraints. We demonstrate PRISM's flexibility by two complementary implementations: a generatively trained Gaussian Mixture Model as well as a discriminatively trained histogram approach. Both systems achieve state-of-the-art performance. Detections are found by gradient-based or branch and bound search, respectively. The latter greatly benefits from PRISM's feature-centric view. It thereby avoids the unfavourable memory trade-off and any on-line pre-processing of the original Efficient Subwindow Search (ESS). Moreover, our approach takes account of the features' scale value while ESS does not. Finally, we show how to avoid soft-matching and spatial pyramid descriptors during detection without losing their positive effect. This makes algorithms simpler and faster. Both are possible if the object model is properly regularised and we discuss a modification of SVMs which allows for doing so.

Keywords Object detection · Hough transform · Sliding-window · Branch and bound · Soft-matching · Spatial pyramid histograms

A. Lehmann (✉) · L. Van Gool
Computer Vision Laboratory, ETH Zurich, Zurich, Switzerland
e-mail: lehmann@vision.ee.ethz.ch

L. Van Gool
e-mail: vangool@vision.ee.ethz.ch

B. Leibe
UMIC Research Centre, RWTH Aachen, Aachen, Germany
e-mail: leibe@umic.rwth-aachen.de

L. Van Gool
ESAT-PSI/IBBT, KU Leuven, Leuven, Belgium

1 Introduction

Object detection is the problem of joint localisation and categorisation of objects in images. It involves two tasks: *learning* an accurate object model and the actual *search* for objects, *i.e.*, applying the model to new images. While learning accurate models is not time critical (as it is done off-line), speed is of prime importance during detection. As a matter of fact, the structure of the model has a direct impact on the efficiency of the detection. Therefore, object detection involves an additional, third task: *modelling the problem* such that it allows for *efficient search.* This task, which precedes both others, and the subsequent acceleration of object detection is the focus of this paper. A central aspect towards this goal is to move as much computation as possible to the off-line training stage where runtime is not critical.

Most state-of-the-art object detectors are based on either the sliding-window paradigm (Viola and Jones 2004; Schneiderman and Kanade 2004; Dalal and Triggs 2005; Ferrari et al. 2007; Felzenszwalb et al. 2008; Maji et al. 2008) or the Hough transform (Ballard 1981; Opelt et al. 2006; Leibe et al. 2008; Liebelt et al. 2008; Maji and Malik 2009). Sliding-window considers all possible sub-images of an image and a classifier decides whether they contain an object of interest or not. For reasons of efficiency, mostly linear classifiers are used, although fast non-linear approaches

have been proposed recently (Maji et al. 2008). Moreover, advanced search schemes based on branch and bound have been designed to overcome the computationally expensive exhaustive search (Lampert et al. 2009). In this work we focus on the second aforementioned paradigm, *i.e.* the Hough-transform, and show that it can also—and even more—benefit from branch and bound search.

The Hough transform was originally introduced for line detection, while the Generalised Hough Transform (Ballard 1981) presented modifications for finding predefined shapes other than lines. More recently, the Implicit Shape Model (ISM) (Leibe et al. 2008) has shown how the underlying idea can be extended to object category detection from local features. It is this extended form which we refer to in this paper. In ISM, processing starts with local feature extraction and each feature subsequently casts probabilistic votes for possible object positions. The final hypothesis score is obtained by marginalising over all these votes. In our opinion, such bottom-up voting schemes seem to be more natural than the exhaustive sliding-window search paradigm.

Indeed, object class detectors based on ISM's idea have become increasingly popular (Liebelt et al. 2008; Opelt et al. 2006; Maji and Malik 2009; Gall and Lempitsky 2009; Chum and Zisserman 2007). However, we argue that the commonly used ISM formalism is unsatisfactory from a probabilistic point of view. In particular, the summation over feature likelihoods is explained by marginalisation. However, the extracted features co-exist and are *not* mutually exclusive. Thus, marginalising over them is meaningless which will be detailed in Sect. 2.3. Nevertheless, ISM empirically demonstrates robust detection performance. Furthermore, the summation is crucial for the voting paradigm. Hence, the question is: "How else can this summation be justified?". We set out to give a sound answer to this question by formulating Hough-based object detection as a dual implementation of linear sliding-window detection (Lehmann et al. 2009b). As a result, PRISM brings together the advantages of both paradigms. On the one hand, the sliding-window reasoning resolves ISM's deficiencies, *i.e.*, PRISM gives sound justification for the voting procedure and also allows for discriminative (*i.e.*, negative) voting weights (which was not possible before). This contribution plays out on the level of object *modelling* and motivates the name PRISM: *PRincipled* Implicit Shape *Model*. On the other hand, we will show that the feature-centric view of the Hough-transform leads to computational advantages at detection time.

The central aspect of our PRISM framework (Lehmann et al. 2009b) is to consider the sliding-window and the Hough paradigms as two sides of the same coin. This duality is exploited to define the object score from a sliding-window point of view, while the actual evaluation follows the Hough transform. The core concept which allows the fusion of the two paradigms is a (visual) *object footprint*. It represents all features in a canonical reference frame which is defined through invariants. This compensates for (geometric) transformations of the object. Object hypotheses are then scored by comparing their footprint to a linear object model. The latter is compulsory for the Hough transform, but most sliding-window detectors do likewise for reasons of efficiency. Contrary to spatial histogram-based approaches (Dalal and Triggs 2005; Lampert et al. 2009), we keep track of each individual feature. This leads to a feature-centric score which is crucial for a Hough transform like algorithm. Moreover, it is this feature-centric score which can be exploited to improve the runtime properties of Efficient Subwindow Search (Lampert et al. 2009). In particular, we show significant memory savings and avoid any on-line pre-processing.

Efficient Subwindow Search (ESS) (Lampert et al. 2008, 2009) is an elegant technique to overcome exhaustive search of sliding-window systems. It thereby allows for sub-linear search time. Central to this approach is the use of bounds on sets of hypotheses, embedded in a branch and bound search scheme. Such branch and bound algorithm have already proven their effectiveness in earlier work on geometric matching (Keysers et al. 2007; Breuel, 1992, 2002). However, ESS also has its downsides, which are mainly due to the integral image representation used during detection. More precisely, ESS uses two integral images per spatial bin of its (histogram) object model. These are very memory demanding as they scale with the input image size. Hence, a single-class detector with 10×10 histogram bins (as presented in Lampert et al. 2009, Fig. 8), but without spatial pyramid) already consumes on the order of 235MB memory for moderately sized 640×480 images.[1] Due to this memory issue, ESS uses only 2D integral images although features actually reside in a 3D scale space. Hence, ESS chooses to ignore the feature scale. This limits the modelling capabilities, *i.e.*, small/large-scale feature cannot be distinguished. Furthermore, this may cause a bias towards larger-scale detections as we will discuss later. Depending on the application, the problem may be less the memory usage, but more the fact that memory has to be filled with data immediately prior to the actual search. This on-line pre-processing step may cancel the sub-linear runtime. In any case, as the number of bins scales with the number of classes, ESS will not scale well to large images and many classes. Adapting the ESS idea to the Hough-inspired PRISM framework avoids all of these problems. In particular, we present a system which assigns different weights to small/large-scale features. Therefore, it has no bias towards larger detections.

[1] 2 integral images $\cdot$ 640×480 pixel $\cdot$ 100 bins $\cdot$ 4 bytes $\approx$ 235 MB. Using the 10 level pyramid increases memory usage to about 900 MB. As we argue later, the pyramid proposed in Lampert et al. (2009) is not needed at detection time as it causes model regularisation which can be integrated into the training procedure.

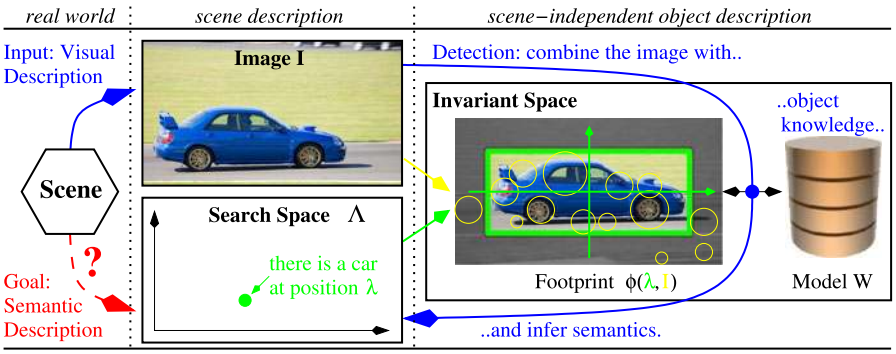

Fig. 1 (Color online) High-level illustration of PRISM's main concepts. The goal of object detection is to recover a semantic description of a scene (*e.g.* "a car at position λ") while we only have access to a visual description, *i.e.*, the image pixels. Both visual and semantic description are scene-dependent and cannot be accessed during training (as they do not exist then). Our object footprint couples them and allows for computing a scene-*in*dependent object description for each object hypothesis. This footprint compensates for (geometric) object transformations by means of invariants. This generalises the usual "windowing" of sliding-window detectors. The explicitly defined invariants induce an invariant space which is scene-independent. This space thus emerges during training which makes learning possible. The footprint's coupling is exploited during detection to combine the input (*i.e.*, the visual description) with knowledge (*i.e.*, the object model) to, eventually, infer the semantics of the scene

Furthermore, no on-line pre-computation is needed. Hence, detection directly starts with the adaptive branch and bound search. Both memory usage and runtime are sub-linear in the size of the search space and linear in the number of features. In our opinion, both dependencies are very natural, which we will discuss.

A rather complementary contribution of our work tackles the common practice of soft-matching and spatial pyramid descriptors. These techniques improve detection quality, but lead to additional cost at detection time. The former has been acknowledged by many authors (Grauman and Darrell 2005; Lazebnik et al. 2006; Ferrari et al. 2007; Ommer and Buhmann 2007; Leibe et al. 2008; Philbin et al. 2008), while this paper shows how to avoid the latter. We argue that both techniques cause model regularisation. This is a concept of learning and does not belong to the detection stage. We demonstrate that soft-matching is indeed not needed during detection if the model is properly regularised. Moreover, we integrate spatial pyramids directly into Support Vector Machines (SVMs) by modifying their usual L_2-norm regularisation. This makes the connection to regularisation explicit. Interestingly, the modified problem can be solved efficiently in the primal form (Chapelle 2007; Sandler et al. 2008). As a result, fast nearest-neighbour matching and flat histograms are sufficient during detection. This makes detection algorithms simpler (without sacrificing quality) and faster, which is the focus of this paper.

In summary, this paper tackles the task of *object modelling* and *efficient search*. It combines elements of what is currently the state-of-the-art in both sliding-window and Hough-based object detection, as it is indebted to both ESS (Lampert et al. 2009) and ISM (Leibe et al. 2008). It puts ISM on a well-grounded mathematical footing and shows that the ESS principle can also be applied in a different, feature-centric manner. Depending on the parameters of the problem at hand (such as the number of classes, number of extracted features, image size, etc.), one *may* be served better by taking this alternative view. Moreover, we will actually show that the traditional ESS bound is encompassed by our framework. This said, and as a disclaimer, the paper is not about learning better model parameters and, hence, not about improving object detection rates.

The structure of the paper is as follows: The PRISM framework (Lehmann et al. 2009b) is introduced and discussed in Sect. 2. Two implementations of this framework are presented in Sects. 3 and 4, respectively. Both algorithms achieve state-of-the-art performance and are complementary to each other, *i.e.*, the two sections can be read independently. The former builds on Gaussian Mixture Models, which allow for efficient gradient based search. The combination of PRISM with branch and bound (Lehmann et al. 2009a) is demonstrated in Sect. 4 along with a detailed discussion and a comparison to ESS. Section 5 describes how soft-matching and spatial pyramids can be moved to the training stage, thereby allowing for fast NN-matching and flat histograms during recognition. Section 6 gives concluding remarks.

2 The PRincipled Implicit Shape Model (PRISM)

Object detection can be formulated as a search problem: Given a newly observed image I and a trained object

Stage	Description	ESS	Ours
Training (off-line)	learning	SVM	RSVM
	off-line pre-processing	–	II
observing a new image			
Detection (time critical)	on-line pre-processing	FE, PYR, II	FE
	search	BNB	BNB

Fig. 2 Computations in the object detection pipeline. Clearly, feature extraction (FE) and branch and bound (BNB) search depend on the newly observed image. Hence, they cannot be avoided in the detection stage. However, PRISM's feature-centric view allows for pre-computing integral images (II) off-line during training, whereas ESS needs to compute them during detection. This is a clear advantage as speed is of prime importance during detection, while the off-line training stage is not time critical. Furthermore, spatial pyramid descriptors (PYR) can be avoided during detection by using SVMs with a modified regularisation term (RSVM)

model W, the goal is to find the best hypothesis

$$\lambda^* = \overbrace{\arg\max_{\lambda\in\Lambda}}^{\textit{searching}} \overbrace{S\Big(\lambda|I,\ \underbrace{W}_{\textit{learning}}\Big)}^{\textit{modelling}}, \tag{1}$$

where S is a score function and Λ is the search space of all potential object hypotheses. Furthermore, this formulation reveals the *three tasks* involved, *i.e.*, *modelling, learning, and searching*. In this section, we are concerned with the modelling problem, *i.e.*, the design of the score function S.

In general, there are no restrictions on the score function. However, the structure of S plays an important role when it comes to defining efficient search algorithms. As the search space is large, quickly finding the best-scoring hypothesis is of great importance. We believe that *feature-centric* scores (as introduced in ISM (Leibe et al. 2008)) offer a powerful approach. In particular, it allows us to perform certain pre-computations off-line during training (c.f. Fig. 2). In such a framework, local features are matched to a visual vocabulary and cast votes for possible object positions (the score being the sum of votes). However, ISM's probabilistic model has various shortcomings and also does not allow for *discriminative voting weights*, *i.e.*, votes which can be negative and, thus, penalise certain hypotheses.

In the sequel, we will present the PRISM framework which resolves these problems. We pick up the reasoning of the sliding-window paradigm, but derive a feature-centric, Hough-like score in Sect. 2.1. A high-level illustration of PRISM's main concepts is shown in Fig. 1. The duality of the sliding-window and the Hough paradigms are highlighted in Sect. 2.2. The shortcomings of ISM and related work are discussed in Sects. 2.3 and 2.4, respectively. Comments on multiple object detection follow in Sect. 2.5.

2.1 Feature-Centric Score Function

The central element of PRISM (Lehmann et al. 2009b) is a footprint $\phi(\lambda, I)$ for a given object hypothesis λ extracted from the image I. This footprint maps all detected features into a canonical reference frame. In our implementation, it describes an object independently of its location and scale in the image. Intuitively, it crops out a sub-image, which is the key idea of the sliding-window paradigm (*c.f.* Fig. 1). Unlike general (non-linear) sliding-window scores, PRISM uses a linear score function. Such a linear model is compulsory for the Hough transform. Hence, the hypothesis score is computed by the inner product

$$S(\lambda|I, W) = \langle \phi(\lambda, I), W \rangle \tag{2}$$

of the footprint ϕ and a weight function W, *i.e.*, the object model. Classical sliding-window detectors (Dalal and Triggs 2005; Felzenszwalb et al. 2008; Lampert et al. 2009) represent all features in an object-centric coordinate frame. Moreover, the relative feature position is often discretised which leads to a histogram representation (Dalal and Triggs 2005) for ϕ and W. Contrary to previous work, we focus on the definition of the mapping function ϕ and avoid the discretisation. This allows us to switch from the sliding-window to a feature-driven, Hough-transform algorithm. This change of views is central to our framework and is the main reason for the favourable properties of our branch and bound algorithm (in Sect. 4).

Image-Object Invariants. An important aspect of the footprint is that it relates objects and features in an invariant way. In order to define invariant expressions, we first have to specify the image representation and hypothesis parametrisation. For the image representation we consider a set of features $\mathcal{F}$. Each feature is characterised by a descriptor, position, and scale, *i.e.*, $f = (f_c, f_x, f_y, f_s)$. In this work, we use local features (Mikolajczyk and Schmid 2004) and f_c is an index to the best-matching visual word in a learned vocabulary. For the sake of completeness, a feature may be modulated with a factor $f_m > 0$ which accounts for *e.g.* the quality of the match to the vocabulary. In the sequel, we will refer to this factor as the feature mass. As object hypothesis parametrisation we use $\lambda = (\lambda_x, \lambda_y, \lambda_s)$, *i.e.*, the object's position and size, respectively. This is equivalent to a bounding box with fixed aspect ratio. Finally, possible mappings of these scene-dependent variables (λ, f) into a translation- and scale-invariant space are *e.g.*

$$\mathbb{I}_f = \left[\frac{\lambda_x - f_x}{f_s}, \log\frac{\lambda_s}{f_s}\right] \quad \text{or} \quad \mathbb{I}_\lambda = \left[\frac{f_x - \lambda_x}{\lambda_s}, \log\frac{f_s}{\lambda_s}\right], \tag{3}$$

where the y-coordinate is analogous to x and is dropped for brevity's sake. Using the logarithm accounts for the multiplicative nature of the scale ratio and will be helpful later in Sect. 3.1. I_f considers a feature-centric coordinate frame,

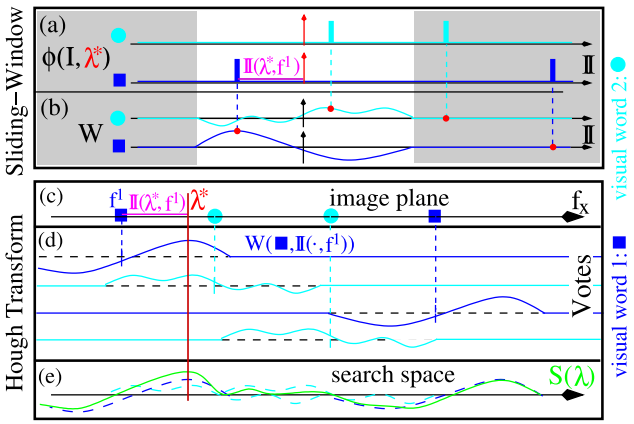

Fig. 3 (Color online) From Sliding-Window (*top*) to the Hough Transform (*bottom*). Shift-invariant 1D-example with four extracted features (**c**) quantised into two visual words (*blue rectangle* and *cyan circle*). Sliding-window reasoning (*top*): A *single* hypothesis λ^* is fixed (*red*). Its footprint (**a**) is a sum of Dirac pulses which are positioned according to the invariant $\mathbb{I}(\lambda^*, f) = f - \lambda^*$ (depicted in (**a**, **c**) for f^1). This aligns the features to the object model. The inner product with the weight function (**b**) results in a sum of point evaluations (*red dots*). Features falling into the range where the weight function is zero (*shaded*) do *not* affect the score. Hough-Transform view (*bottom*): all features are considered *simultaneously*. Extracted features (**c**) cast voting patterns (**d**) which, summed up, result in the final hypothesis score (**e**). The voting patterns are transformed (*e.g.*, mirrored and shifted) versions of the weights W. Their intersections with the *red line* corresponds to the *red dots* in the sliding-window setting (**b**). The *solid green* curve (**e**) shows the overall hypothesis score and one can identify λ^* as a local maximum. The alternation of greedy search and feature suppression (*c.f.* Sect. 2.5) implements a matching pursuit like algorithm to recover the sparse semantic description (*i.e.*, "a car at position λ^*") from the blurry semantic reconstruction (**e**)

while I_λ opts for a classical object-centric one, *i.e.*, registering feature position relative to the object center (*c.f.* Fig. 5 (left)). The implications of using one or the other will be discussed later in Sect. 3.2.

Object Footprint. Given an invariant $\mathbb{I}$, we define the joint mapping of an object-feature pair (λ, f) as a weighted Dirac delta function $f_m \delta_{f_c, \mathbb{I}(\lambda, f)}$. It is zero everywhere but at the 4D-point $[f_c, \mathbb{I}(\lambda, f)]$ and integrates to f_m. The entire object footprint is defined as the superposition of all features, *i.e.*

$$\phi(\lambda, I) = \sum_{f \in \mathcal{F}} f_m \delta_{f_c, \mathbb{I}(\lambda, f)}. \tag{4}$$

Figure 3 illustrates the footprint as well as the change to the Hough transform, which is discussed in the sequel. In classical sliding-window systems, this 4D footprint (function) is discretised and represented by a histogram. We avoid the discretisation and plug it directly into the score (2). Thus, the inner product of the two functions yields a score S for hypothesis λ as

$$S(\lambda) = \langle \phi(\lambda, I), W \rangle = \sum_{f \in \mathcal{F}} f_m W(f_c, \mathbb{I}(\lambda, f)), \tag{5}$$

i.e., as point evaluations of the weight function (*c.f.* Fig. 3(b)). This form makes the connection to the feature-driven Hough-transform explicit. The summand $f_m W(f_c, \mathbb{I}(\cdot, f))$ represents the "vote" cast by each feature which is illustrated in Fig. 3(d). Although no longer really visible, this formulation is *equivalent* to a standard sliding-window classifier (*e.g.*, Dalal and Triggs 2005) when considering the object-centric invariant $\mathbb{I}_\lambda$ and a histogram for W. We are however not restricted to this choice. Any representation for W is possible and also more general invariants than (3) can be used. They may account for *e.g.* variable aspect ratio, viewpoint changes, etc. A nice property of our framework is that the modelling of such geometric transformations is made *explicit* (*i.e.*, through the definition of $\mathbb{I}$) and is *decoupled* from the rest of the system.

Non-Contributing Features. Objects are compact and have a finite extent. Hence, unless contextual information is modelled, the weight function W has compact support and is zero outside that range (Fig. 3(b), shaded region). As a consequence, many (far away) features do not contribute to the score, *i.e.*, $W(f_c, \mathbb{I}(\lambda, f)) = 0$. Thus, cropping out a sub-image in sliding-window approaches is a consequence of the learned model, *i.e.*, the weight function W. In practice, identifying and excluding such features reduces runtime significantly. Doing that properly is *not* a detail, but an important aspect of an algorithm. As it strongly depends on the search algorithm, we postpone further discussions to Sects. 3.3 and 4.5, respectively.

2.2 Bottom-up? Sliding-Window vs. Hough-Transform

Sliding-window detectors are usually considered fundamentally different from Hough-transform like approaches. The former are said to reason in a top-down fashion, while the latter are considered to work bottom-up. Actually, previous work has shown that both paradigms can co-exist in a single detection system (Schneiderman 2004; Chum and Zisserman 2007). Schneiderman (2004) detects objects by cascades of classifiers and uses the feature-centric Hough approach in the first cascade levels. This allows for sharing feature among different hypotheses and is reported to be much faster. Later stages follow the sliding-window approach and new features are extracted for each tested hypothesis. Chum and Zisserman (2007) also start with the Hough-transform. They use it to quickly find good starting points for a sliding-window based, gradient refinement algorithm. However, both works treat the two concepts as complementary. An attempt to bring them into a closed relationship has been investigated by Williams and Allan (2006),

Sliding Window	*Hough Transform*
for $\lambda \in \Lambda$	**for** $f \in \mathcal{F}$
for $f \in \{\bar{W}(\lambda, f) \neq 0\}$	**for** $\lambda \in \{\bar{W}(\lambda, f) \neq 0\}$
$S(\lambda)$ += $\bar{W}(\lambda, f)$	$S(\lambda)$ += $\bar{W}(\lambda, f)$
end	**end**
end	**end**

Fig. 4 Pseudo-code for sliding-window and Hough transform. We use the shortcut $\bar{W}(\lambda, f) := f_m W(f_c, \mathbb{I}(\lambda, f))$. Sliding-window detectors process all hypotheses sequentially. For each hypothesis, the entire score function is evaluated *at once* which involves finding contributing features. Contrarily, the Hough transform is feature driven and processes all objects *in parallel*: each feature adds its non-zero contributions to the corresponding objects, *i.e.*, an accumulation of voting patterns in the search space

whose approach relies on a Taylor approximation. In our framework, the duality of the two approaches emerges very naturally, as can be seen from (5). The left-hand side represents the object-centric view taken by sliding-window detectors. There, the complete footprint ϕ (usually a histogram) is extracted and compared to the model. The right-hand side expresses the feature-centric formulation of Hough-transform based systems. Note that both views are *equivalent*. The actual difference is more of an *algorithmic* nature, *i.e.*, how the score is evaluated for all possible object hypotheses. This difference can be explained by an interchange of loops as illustrated in Fig. 4. Thus, in our opinion, both approaches follow the same top-down reasoning and only feature extraction (as pre-processing) should be seen as a bottom-up process. In summary, Hough transform based object detectors are a sub-class of sliding-window detectors, *i.e.*, the sub-class which pre-computes all features and which scores hypotheses with a linear detection model.

2.3 The Implicit Shape Model (ISM)

As mentioned in the introduction, ISM (Leibe et al. 2008) bears various problems which we will discuss now. The score function of ISM[2] is defined as a probability density over the whole search space, *i.e.*

$$p(\lambda|I) = \sum_{f \in \mathcal{F}} p(\lambda|f)p(f|I). \tag{6}$$

We argue that marginalisation (*i.e.*, the sum over features) is not justified. It implies that all features f are possible realisations of a *single* random variable. In other words, they would be mutually exclusive and only *one* feature could be observed. However, we do observe/extract *all* these different features concurrently, *i.e.*, they co-exist. Hence, they are all facts[3] and marginalisation over (uncertainty-free) facts is not meaningful. Thus, each feature should be modelled with a separate random variable, *i.e.*, $p(\lambda, f_1, f_2, \ldots, f_n)$, as done correctly in other probabilistic setups (Sudderth et al. 2005). This allows for computing the likelihood of the observed evidence. For tractability, features are often assumed to be independent which leads to the factorisation $p(\lambda) \prod_i p(f_i|\lambda)$, *i.e.*, a multiplicative combination instead of the additive one in ISM. The summation is however crucial for the voting concept. Moreover, as pointed out by Kittler et al. (1998), additive combination of evidence is more robust than a multiplicative one, which is in line with the empirically observed robust detection performance of ISM. Thus, the question is how to still justify the summation.

A second problem is the normalisation constraint of the density $p(\lambda|I)$. This (global) normalisation implies that every feature affects the score of every hypothesis. Unless contextual information is explicitly modelled (which is not the case), such long-range interactions are not justified. Furthermore, the ISM formulation involves only scene-dependent variables (*c.f.* Fig. 1) which causes another problem. Namely, the occurrence distribution $p(\lambda|f)$ is a function (defined on the scene-dependent search space Λ) which makes statements about the *absolute position* of objects in an image. Thus, this function is not defined if the image of interest is not yet observed, which is the case during training. Hence, strictly speaking, the function $p(\lambda|f)$ cannot be learned off-line during training. In other words, the ISM formalism implicitly assume the use of invariants which encode the relative offset of feature and object locations. ISM's formulation thus leaves open some ambiguity as there is not only one single possible invariant (*c.f.* (3)). Moreover, we will show later (Sect. 3.2 and Fig. 5) that the choice of the invariant can have significant implications on the final algorithm.

PRISM (Lehmann et al. 2009b) overcomes these issues by a sliding-window based reasoning. The scene-independent invariant space adds an intermediate representation which makes clean learning possible. Moreover, each hypothesis is scored independently, *i.e.*, there is no global normalisation, and the compact support of W ensures that the features have only local influence. The summation of feature contributions results from the definition of the footprint and the linear model. Both are valid modelling choices and are compulsory to obtain a Hough transform algorithm. PRISM imposes minimal constraints as there are no further restrictions on the function W. Therefore, we are

[2] The Minimum Description Length (MDL) post-processing is not considered here as it is not part of the Hough voting procedure.

[3] Feature matching may be formulated probabilistically. Then, each feature's visual word id f_c is a random variable which expresses the probabilistic matching (to the vocabulary). Marginalising over these random variables is perfectly valid as visual words are mutually exclusive and only one visual word should be activated. This marginalisation would lead to an additional summation in (6).

free to choose any suitable representation and learning algorithm. In particular, discriminative voting weights (*i.e.*, giving positive *and/or* negative contributions to the score) can be learned which was not possible in ISM. We demonstrate this potential in Sect. 4.5 by means of SVM learning (Shawe-Taylor and Cristianini 2004).

2.4 Related Work

To the best of our knowledge, there has not been previous work which addressed the problem of giving a sound justification to ISM's voting procedure. Yet, there are extensions to ISM which aim at discriminative learning. They are however less general than PRISM and in fact only reinforce the need for such a justification. Fritz et al. (2005) present a discriminative validation procedure which reuses information from the voting stage. However, it is only a *post-processing* and is not integrated into the voting procedure itself. Such validation can be applied to *any* detection system. More recently, a maximum-margin framework (Maji and Malik 2009) was introduced which learns a per-visual-word re-weighting. This is less general than PRISM as each spatial position encounters the same re-weighting and the final weights are all still restricted to positive values. Moreover, they also adopt the questionable marginalisation argument. Gall and Lempitsky (2009) present a probabilistic model based on randomised forests. They introduce the "event" of an object being at a certain position. This actually implements the sliding-window paradigm. Unfortunately, their probabilistic formulation restricts the voting weights to be positive and they admit the lack of a sound probabilistic interpretation of the summation over features. However, a very interesting contribution of their work is to learn spatial position and visual appearance jointly. In other words, they avoid the common two-step procedure where one first clusters a visual vocabulary, followed by estimating the voting weights. PRISM does not prevent such joint-learning. This would be achieved by learning a weight function $W : f \times \lambda \mapsto W(f, \mathbb{I}(\lambda, f))$ which directly takes the continuous feature descriptor (or the observed image patch) as first argument instead of a discrete visual word identifier. Nonetheless, we stick to the classical two-step learning procedure here.

In summary, PRISM imposes minimal restrictions on the model. This offers great flexibility. The only restriction is that the features contribute linearly and independently of each other. Both are necessary for the Hough transform. However, this is not a huge burden since most (fast) detectors rely on such linear models (Lampert et al. 2009; Felzenszwalb et al. 2008; Dalal and Triggs 2005; Leibe et al. 2008).

2.5 Multiple Objects

Detection of multiple objects is another point where, in our opinion, the sliding-window reasoning (of our framework) offers a cleaner solution. ISM's probabilistic model (without MDL, *i.e.* (6)) answers the question "*Where* is the object?" and completely ignores the important questions "*Is* there an object?"[4] or "How many objects are there?". Sliding-window reasoning, on the other hand, answers for each position independently if it contains an object or not. It postpones model selection (*i.e.* "How many objects are there?") to later components in the object detection pipeline. Such model selection may be implemented via *e.g.* the Minimum Description Length (MDL) principle as nicely demonstrated in ISM (Leibe et al. 2008).

Although sliding-window allows for multiple detections, it does not account for already explained evidence. Hence, it tends to re-detect the same object. A possible way to resolve this problem is to limit each feature to support only one single object. In this work, we consider a greedy strategy: once the best hypothesis is found, we eliminate all features which explain this object and restart the search for more objects. This procedure automatically suppresses the score of nearby hypotheses and relates to the usual non-maximum suppression post-processing. Clearly, the MDL based optimisation strategy implemented in ISM (Leibe et al. 2008) is more powerful as all hypotheses compete for evidence (in a pair-wise fashion). This is an advantage in situations with strongly overlapping objects. However, MDL is applied as post-processing and only considers hypotheses found by greedy detection.

At first sight, our alternation of greedy search and local suppression seems to be a heuristic. However, it is actually closely related to the well-established and thoroughly analysed matching pursuit (MP) algorithm (Mallat and Zhang 1993), which gives a sound justification for our multi-object detection approach. MP decomposes a signal into a linear expansion of functions which are selected from a redundant, over-complete dictionary. The signal is assumed to be sparse (w.r.t. this dictionary) and MP aims at recovering the non-zero elements. Unfortunately, finding an optimal decomposition is NP-complete which makes approximate algorithms like MP inevitable. In our context, each hypothesis yields one dictionary item which is a translated and scaled version of the model. The signal corresponds to the extracted low-level features and the hypothesis score measures the match of each dictionary item. Sparsity follows from the semantic nature of the dictionary, *i.e.*, each real object causes one non-zero dictionary item. The main difference to a proper MP algorithm is that our dictionary is

[4] Assuming that an object is present may be reasonable in a tracking context, but not for object detection.

not complete. This is because we only model objects, *i.e.*, things and not stuff (Heitz and Koller 2008) and we are not interested in reconstructing the image.

3 Gaussian Mixture Model & Gradient Search

So far, we have presented an abstract framework. To obtain a concrete object detection system, we need to specify the geometric invariant, the representation of the weight function W, and the search algorithm. In this section, we closely follow the ideas of ISM (Leibe et al. 2008) in order to show how pure ISM can be modelled within our framework. Moreover, this makes a direct comparison possible. As model representation, we use Gaussian Mixture Models (GMMs) which are presented in Sect. 3.1. The choice of the geometric invariant and proper scale-adaptation is discussed in Sect. 3.2, followed by algorithmic details in Sect. 3.3. The system is evaluated in Sect. 3.4 and is shown to compare well with the state-of-the-art. We further demonstrate that the GMM makes it possible to speed up the search with no, or only little loss of accuracy.

3.1 Gaussian Mixture Model

The Implicit Shape Model follows a generative object modelling approach, where the weight function is given by a density estimator

$$W(c, \mathbb{I}(\lambda, f)) = p_c(\mathbb{I}(\lambda, f)) \quad \forall c \tag{7}$$

for each visual word c, *i.e.*, the occurrence distributions in ISM terminology. We do likewise in this section, but consider Gaussian Mixture Models (GMMs) instead of non-parametric kernel density estimators as used in ISM. The disadvantage of kernel density estimators is their strong dependence on the training data (in terms of storage and computation time) which is unfavourable for large training sets. Our GMM approach avoids this downside. Modelling the ratio of scales in a logarithmic space, *i.e.*, $\log \lambda_s / f_s$, is important in such a GMM setup. Otherwise, the positivity constraint $\lambda_s / f_s > 0$ could not be satisfied with infinitely supported Gaussians. Learning is performed by mapping all features of all training objects into the invariant space, where the distributions are estimated. We use an Expectation-Maximisation (EM) implementation which automatically adapts the number of mixture components using a split/merge criterion (Baggenstoos 2002). The analytic structure of GMMs allows for computing the derivatives of the score function, *i.e.*, $\frac{\partial S(\lambda)}{\partial \lambda}$. Thus object localisation can be done by means of efficient gradient-based search (Carreira Perpiñán 2000), which relates to the mean shift algorithm in ISM. Such a local refinement strategy may be of particular interest if one already has prior knowledge about objects (*e.g.* from object tracking). The initialisation of the search (*i.e.* finding starting points) will be discussed in Sect. 3.3.

Modified GMM. Contrary to the ISM (Leibe et al. 2008), we are not forced to use densities $p_c(\mathbb{I})$. We exploit this freedom and consider a simple re-scaling of the weight function, *i.e.* $W(c, \mathbb{I}) = \eta_c p_c(\mathbb{I})$ with $\eta_c = 1/(\max_{\mathbb{I}} p_c(\mathbb{I}))$, which shows significant performance improvements. The goal of this step is that the maximal weight (of every visual word) is one.[5] An intuitive explanation is as follows. The maximal weight of multi-modal occurrence distributions tends to be smaller than the one of rather peaked distributions. Thus, depending on which visual words are activated, the object score may be lower, which is undesirable. A more sophisticated re-scaling strategy is explored by Maji and Malik (2009). They present a maximum margin framework which allows for learning good re-weighting factors η_c in a discriminative fashion. The fact that both their and our re-weighting yields performance improvements suggests that pure density estimation is not an optimal choice.

3.2 Invariants and Voting

We introduced two possible invariants for translation- and scale-invariant object recognition, *i.e.*, $\mathbb{I}_\lambda$ and $\mathbb{I}_f$ (3). Although they achieve invariance under the same transformation group, there are differences which impact on the algorithm and thus require some analysis. The two invariants are related by the difference of object and feature location, *i.e.*,

$$\underbrace{\begin{pmatrix} f_s \mathbb{I}_{f,x} \\ \mathbb{I}_{f,s} \end{pmatrix}}_{\stackrel{def}{=} \mathbb{I}_f(f)} = \underbrace{\begin{pmatrix} \lambda_x - f_x \\ \log \lambda_s - \log f_s \end{pmatrix}}_{\lambda - f} = \underbrace{-\begin{pmatrix} \lambda_s \mathbb{I}_{\lambda,x} \\ \mathbb{I}_{\lambda,s} \end{pmatrix}}_{\stackrel{def}{=} \mathbb{I}_\lambda(\lambda)}, \tag{8}$$

where the subscripts x and s refer to the first and second coordinate of the invariants, respectively. Despite the close relation, there is a subtle difference due to the dependency on f_s or λ_s, which suggests the preferable usage: predicting objects $\lambda \sim \mathbb{I}_f(f) + f$ given a feature f or sampling features $f \sim \mathbb{I}_\lambda(\lambda) + \lambda$ to verify an object hypothesis λ.[6] The former approach is taken by the Generalised Hough Transform (*e.g.* Ballard 1981; Leibe et al. 2008), while the latter corresponds to a sliding window classifier scenario (*e.g.*, Dalal and Triggs 2005; Lampert et al. 2009). In both cases, it is a simple scaling and translation of the model. The converse use, on the other hand, is unfavourable, due to complicated couplings (*e.g.*, $\lambda_x = f_x - f_s \exp(-\mathbb{I}_{\lambda,s})\mathbb{I}_{\lambda,x}$), but not impossible. The feature-centric invariant $\mathbb{I}_f$ has clear advantages for voting, as illustrated in Fig. 5. On the one

[5] We actually approximate the rescaling factor by the mixing factor of the dominant mixture component. Thus, the maximal weight will only be approximately one.

[6] Informally, we interpret $\mathbb{I}$ as a random variable where the probability of a particular value is proportional to the corresponding weight $W(f_c, \mathbb{I}) = p_c(\mathbb{I}(\lambda, f))$.

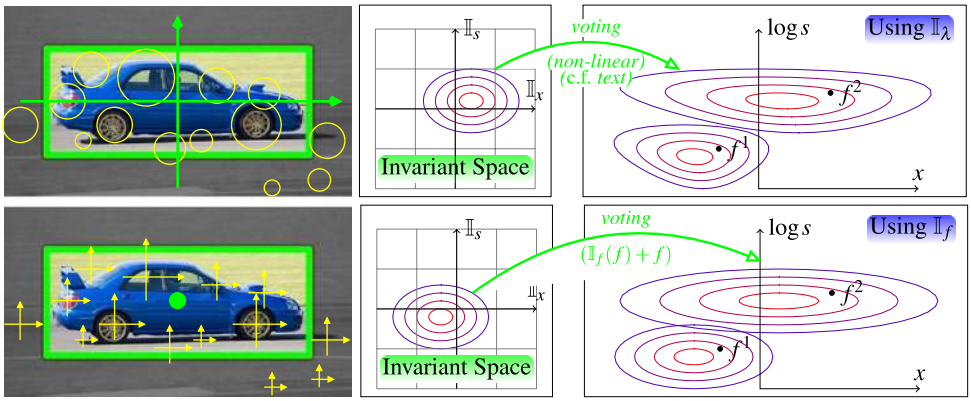

Fig. 5 (Color online) Illustration of the two invariants (3). (*Left*): Visualisation of the established coordinate frame(s). Features are shown in *yellow*, while object-related elements are in *green*. The *object-centric* invariant $\mathbb{I}_\lambda$ (*top*) attaches a single coordinate frame (*arrows*) to the object and registers the relative position of every feature. The *feature-centric* invariant $\mathbb{I}_f$ (*bottom*) attaches a coordinate system to every feature, each of which records the objects reference point (*green dot*). (The *arrow* length accounts for the feature/object scale.) (*Middle*): Contour lines of a single Gaussian (of a learned mixture model) in the induced invariant space. Due to the non-linear relation of the two spaces, it is not possible to define Gaussians which represent the same information. (*Right*): Result of the voting process given two features f^1 and f^2 (*black dots*). The mapping into the hypothesis space strongly depends on the chosen invariant. In case of $\mathbb{I}_f$ (*bottom*), the resulting contours are still elliptic (*i.e.*, again a Gaussian), while $\mathbb{I}_\lambda$ (*top*) leads to complicated, non-linear distortions which make *explcit* parametric voting impossible

hand, each mixture component in the invariant space can be mapped into the hypothesis space by simply adapting mean and covariance of each Gaussian. This is the actual *voting*. Such (explicit) voting is not possible with $\mathbb{I}_\lambda$, because its non-linear distortion breaks the algebraic structure of a Gaussian (*i.e.*, the contour lines are no longer elliptic, see Fig. 5). On the other hand, also the computation of the (score-)gradient (with respect to λ) is simpler than with $\mathbb{I}_\lambda$.

The above analysis actually nicely connects to the early development of ISM (Leibe and Schiele 2004). In order to cope with scale-invariant object detection, a modified, scale-adaptive version (Comaniciu et al. 2001) of mean-shift had to be used. This scale-adaptation implements the above-mentioned scaling, which results naturally in our framework. In other words, ISM implicitly operates in the invariant space (which is not defined in ISM (Leibe and Schiele 2004)) instead of the hypothesis space. Further note that the scaling only affects the covariance matrix of a Gaussian, not its mixture weight. Thus, while the per-feature GMM is a density in the invariant space, it no longer is one in the hypothesis space, *i.e.*, it does not integrate to one.

3.3 Implementation

In essence, this object detector works by performing gradient ascent on a sum of weighted Gaussians. This has been well studied by Carreira Perpiñán (2000), but certain algorithmic details should be considered in practice. On the one hand, we exploit the spatial locality of features by *lists of contributing features*. On the other hand, we have to start the search from multiple starting points in order to escape local maxima and to find multiple objects. These points are managed by a *list of candidates* which leads to some kind of pseudo-parallel version of multiple starts.

Contributing Features. As discussed earlier, not every feature contributes to every hypothesis. Identifying such contributing features (and ignoring others) positively affects the algorithm's runtime. In this GMM setup, the voting pattern of each feature is a linear combination of Gaussians. Thus, the idea of contributing features can be refined to contributing Gaussians.[7] To accelerate evaluation, we keep a *coarse* $3D$-(location, scale) grid over the hypothesis space. Each grid cell keeps a list of contributing mixture components. Thus, to compute the hypothesis score we only have to consider the Gaussians which are listed in the same grid bin the hypothesis falls into.

Candidate Generation. The grid structure from the last paragraph can also be used to initialise the local gradient based search. Note that each cell of this grid corresponds to a set of object hypotheses. We adopt the idea of computing score bounds on such sets (Lampert et al. 2009) to initialise the local search. We consider a crude estimate by summing the peak contribution (*i.e.*, the value at the mean) of each Gaussian in a given bin. Local maxima of this bound serve as starting points for the gradient-based, local refinement (Carreira Perpiñán 2000).

Algorithm. Our algorithm keeps a list of candidates that is populated as described above. After gradient-based refine-

[7] We assume that a Gaussian contributes significantly within two times the standard deviation.

ment, the best candidate is reported as detection, and candidates with a score below a threshold are removed from the list. As detailed in Sect. 2.5, features with a significant contribution to the detected object are eliminated from the system (*i.e.*, the corresponding Gaussians are removed from the cells and the bound is updated). Then, the algorithm restarts to find more objects. Keeping a list of candidates helps saving computation in subsequent iterations. The reason is that feature removal only changes the score function *locally*. Hence, the refinement of most candidates is likely to converge quickly.

3.4 Experiments

Datasets. We evaluate our system on two benchmark datasets consisting of motorbikes (Fritz et al. 2005) and side views of pedestrians (Leibe et al. 2008). The former includes 153 training and 115 test images, whereas the latter consists of 426 training and 205 test images (Fig. 6 (top)). In order to compare our results to the published baseline, we use the same evaluation criterion as in Fritz et al. (2005), Leibe et al. (2008). A detection is accepted as correct if its bounding box overlap with the ground-truth annotation is above 50%; multiple detections on the same object are counted as false positives. Performance is measured using precision-recall curves which is the common measure for these datasets. As image representation, we consider Hessian-Laplace (Mikolajczyk and Schmid 2004) interest points and compute Shape-Context descriptors (Mikolajczyk and Schmid 2005) for each of these. We choose f_m inversely proportional to the distance of the descriptor to the best-matching visual word.

GMM vs. modified GMM. The probabilistic argument given in Leibe et al. (2005) defines the sum of densities as the objective function to hypothesise object positions. This roughly corresponds to the GMM without modification. Figure 6 (center) evaluates the effect of using the modified GMM. The results are reported along with the baseline performance from (Fritz et al. 2005; Leibe et al. 2005, 2008). The difference between the two objective functions is actually fairly small, *i.e.* only a rescaling of each density function by a factor. Looking at the effect of this little change on the performance, an astonishing impact can be observed. Namely, the modified GMM based objective function clearly outperforms the density based one and yields comparable performance to (motorbikes) or even improves (pedestrians) on the baseline. It has to be stressed at this point that, compared to the baseline systems, our approach does *not* perform a final MDL or SVM verification of the hypotheses.

Runtime. The last experiment measures the runtime of our system (measured on a 3 GHz Intel Core 2). For this experiment we use an external Hessian-Laplace feature extractor from (Mikolajczyk and Schmid 2004), which takes about 0.6 s on a typical motorbike image (340×255) and 1.2 s on a pedestrian image (480×360). This can be reduced dramatically by changing to similar SURF features (Bay et al. 2008) which allows GPU-implementations (Cornelis and Van Gool 2008) which run at about 100 Hz. Thus, feature extraction is negligible compared to the rest of our algorithm. Moreover, the structure of our parametric model allows for a simple heuristic to improve detection time. Let us denote the peak contribution of mixture component i by γ_i. Then, we drop all mixture components whose peaks are smaller than β percent of the maximal peak, *i.e.*, $\gamma_i < \beta \max_k \gamma_k$. The average detection time, as well as the achieved performance at equal error rate are reported in Fig. 6 (bottom) for both datasets. We see that for values up to $\beta = 0.1$ and 0.15, respectively, the computation time (ignoring feature extraction) can be reduced by about a third without decrease in accuracy. This yields a detection time of about 0.35 s for a motorbike image and 0.75 s for pedestrians. If runtime is of prime importance, we can also sacrifice some accuracy in favour of speed. On the pedestrians, for example, we can increase the threshold to $\beta = 0.35$, which yields the same equal error rate as the baseline. Doing so, the detection time decreases from initially 1.25 s to only 0.25 s. This simple, but very effective heuristic exploits the structure of our parametric model and would not be easily realisable in a non-parametric framework (Leibe et al. 2008).

4 Feature-Centric Efficient Subwindow Search

In this section, we develop an alternative detection system (Lehmann et al. 2009a) which is complementary to the previous GMM approach. The focus is on advanced search strategies that have been developed to speedup sliding-window detectors. The duality elaborated in Sect. 2 allows us to transfuse such techniques to our Hough-based PRISM framework. The feature-centric view of the latter is shown to yield improved runtime properties. Moreover, we switch to an SVM-based training procedure to show that discriminative voting weights are possible. We thus change model representation, search strategy, and learning procedure, thereby emphasising the versatility of the PRISM framework.

4.1 Related Work

Ignoring large parts of the search space with only little effort is a crucial factor for efficient search strategies. Cascades of (increasingly complex) classifiers (Viola and Jones 2004) have proven their effectiveness. The idea is to reject most hypotheses with a cheap criterion. Subsequently, more costly but also more accurate classifiers are evaluated on hypotheses which have not yet been rejected. Hypotheses that pass

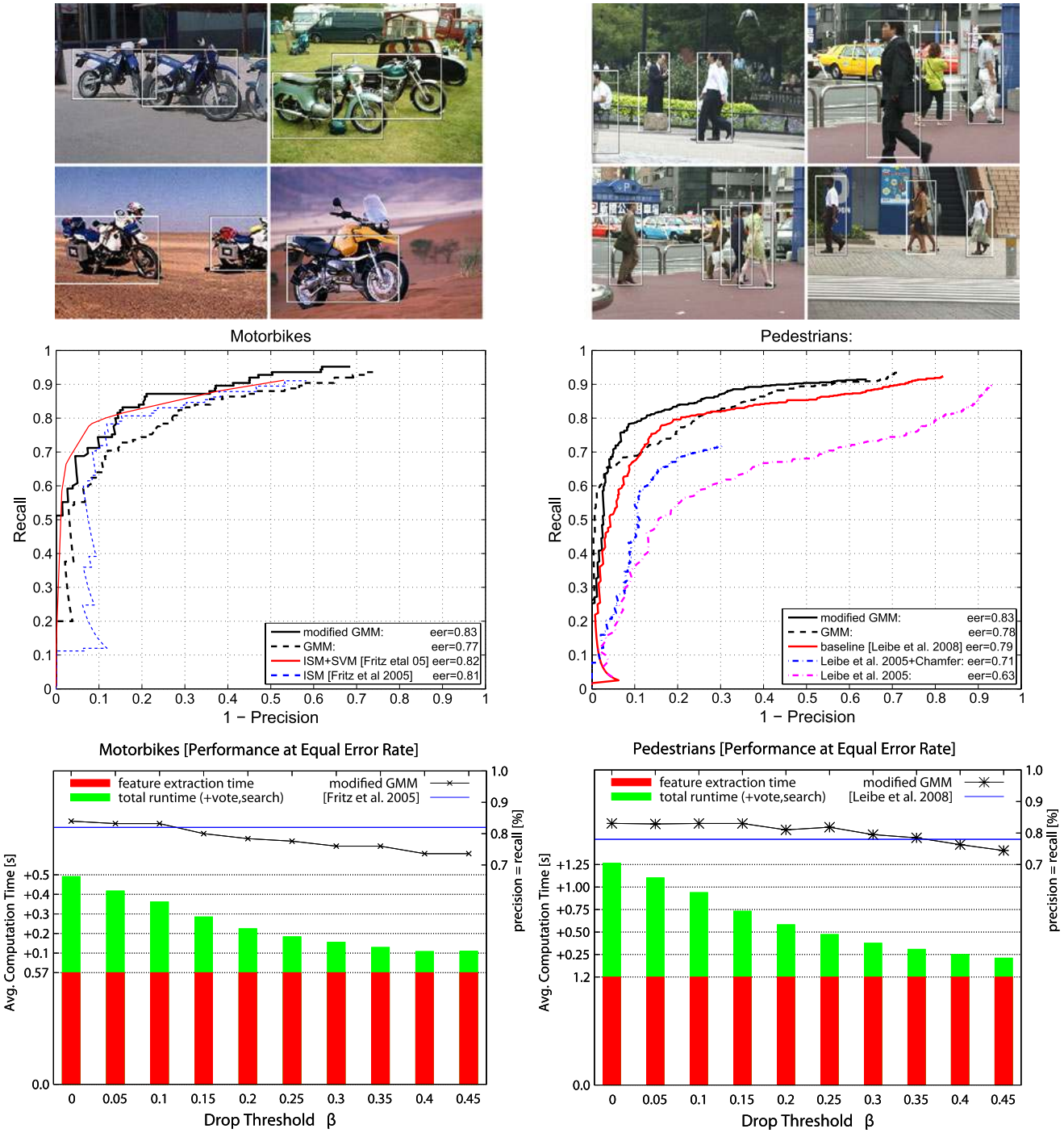

Fig. 6 (Color online) Quantitative evaluation on two benchmark datasets of motorbikes (*left*) and pedestrians (*right*). (*Top*): Sample images of the datasets along with detections. (*Middle*): Performance compared to published baseline; (*bottom*): Detection time and recognition performance with our reduced model. The detection time can be reduced significantly at the cost of only a small decrease in accuracy

all cascade levels are reported as detections. Such cascades drastically reduce computation, but *still* process the search space exhaustively (at least with the first cascade level).

Branch and bound provides a nice alternative. It avoids processing every hypothesis individually, and thus skirts exhaustive search without missing the global optimum. The key idea is to examine entire *sets of hypotheses* by means of an *upper bound* on their score. This allows the algorithm to focus on promising sets and to ignore low-scoring hypotheses early on (without touching each of them individually). Breuel (1992) demonstrated the potential of this technique in the context of geometric matching of point sets.

Keysers et al. (2007) apply that scheme to image matching using patch-based image representations. Only recently, Lampert et al. (2008) extended this scheme from image-to-image to image-to-class matching using a learned object-class model. However, as outlined in the introduction, ESS makes an unfavourable memory trade-off and needs an online pre-processing step. Furthermore, ESS ignores the features' scale value which may cause problems as we will discuss later. These issues can be avoided when applying branch and bound for image-to-class matching in a feature-centric context, which is a key contribution of this work.

Breuel (2002) distinguished two different branch and bound strategies in his comparison paper. One is based on matchlists while the other makes use of point location data structures to efficiently compute bounds. Our approach implements the matchlist strategy whereas ESS is closer related to the point location data structure variant. In that sense, our approach completes the picture of branch and bound for image-to-class matching by the matchlist strategy. Clearly, we are not the first who adopt the ESS principle and other extensions have already been published in the literature.

ESS with a bag-of-words model actually tackles the *maximum sub-matrix* problem, *i.e.*, finding the sub-matrix whose sum of elements is maximal. It's 1D counterpart, the *maximum sub-array* problem, can be solved in linear time using Kadane's algorithm (Bentley 1984). An et al. (2009) combine this dynamic programming technique with ESS's branch and bound. They present two algorithms which have better worst-case guarantees and empirically converge much faster than ESS. However, their bag-of-words model is restrictive and an extension to spatial histograms seems not straightforward. Moreover, some downsides of the original ESS are inherited. Firstly, the sub-matrix formulation ignores the feature scale. Secondly, their algorithms strongly relies on prefix-sums which (similar to integral images) need to be computed at detection time and are expensive in terms of memory.

Yeh et al. (2009) focus on multi-class (specific instance) detection and more general bounding regions, *i.e.*, composite bounding boxes and bounding polygons. The aim of the latter is to have better bounding regions (than just rectangles). Thus, non-object regions are ignored and do not impair the object score. Unfortunately, their flexible bounding regions are restricted to the bag-of-words model. Again, generalisations to account for spatial information (within these flexible bounding regions) seem difficult. Conversely, a spatial histogram model easily allows one to ignore features which do not lie on the object (by setting bin weights to zero), but those clutter regions must be known in advance. Hence, such flexible bounding regions are of particular interest for structureless, texture-defined objects. Yeh et al. (2009) also take a feature-centric view like we do in this work. However, their bound construction is closer related to ESS than ours. Moreover, they do not account for the feature scale. Although possible, the overhead in such a flexible bounding region setup seems to be high. They further present a technique called *data dependent sampling* which yields a significant speedup in their system. It exploits the fact that in a bag-of-words model the optimal bounding box corners coincides with feature point locations. This no longer holds in a spatial histogram model, however. Furthermore, the positive effect gets smaller when increasing the sampling density, which is a common tendency for robust object detection.

We now show how to fuse the ESS principle and our PRISM framework originally presented in Lehmann et al. (2009a). This allows for spatial histogram models which properly account for the features' scale value. Furthermore, it avoids any pre-computation at detection time. Section 4.2 presents a rather generic class of bounds which makes no assumption about the weight function and encompasses the traditional ESS bound (Lampert et al. 2009). From that, we derive a concrete bound in Sect. 4.3 which exploits the histogram structure. This feature-centric bound is complementary to ESS's bound as it makes different trade-offs. A detailed comparison of the two bounds is given in Sect. 4.4. The branch and bound algorithm is revisited in Sect. 4.5, while experiments are shown in Sect. 4.6.

4.2 Score Function on Hypothesis Sets

We start by extending the definition of the score (5) to entire sets of hypotheses $\Omega \subset \Lambda$. Jensen's inequality yields the generic upper bound

$$S(\Omega) \overset{def}{:=} \max_{\lambda \in \Omega} S(\lambda) \leq \sum_{f \in \mathcal{F}} f_m \max_{\lambda \in \Omega} W(f_c, \mathbb{I}(\lambda, f)), \tag{9}$$

where equality holds if all features attain the maximum for the same hypothesis. This is possible, although unlikely. The max term has the following geometric interpretation (*c.f.* Fig. 7). Given a feature f, a hypothesis set Ω (in the hypothesis space) maps to a region $\bar{\Omega}_f = \{\mathbb{I}(\lambda, f) | \lambda \in \Omega\}$ (in the invariant space). For brevity's sake, we drop the feature index, *i.e.* $\bar{\Omega}$. Then, we are interested in the maximum of the function $W|_{\bar{\Omega}}$, *i.e.*, W restricted to the domain $\bar{\Omega}$. This can be interpreted as a *maximum query*: in such a context the weight function W can be considered to play the role of a "database" and $\bar{\Omega}$ that of a query. The goal is to design a method which answers such queries efficiently. It is worth noting that there is *no* restriction on the representation of W. Moreover, facilitating helper structures (*e.g.* integral images) to answer such queries efficiently can be computed *off-line* during learning because the weight function W does not dependent on the test data. The next subsection shows a concrete implementation of such maximum queries.

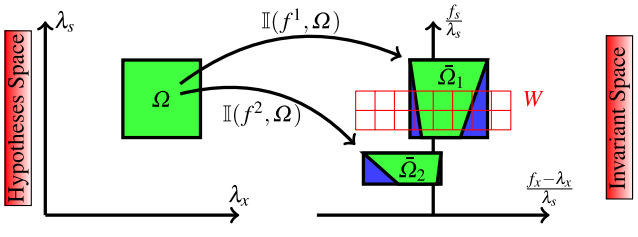

Fig. 7 (Color online) Given (two example) features f^i, a hypothesis *set* Ω is mapped to the invariant space by means of the invariant $\mathbb{I}$. Both the original set Ω and the mapped one $\bar{\Omega}$ are coloured in green. The weight histogram W (only one slice corresponding to the matching visual word) is shown in *red*. Feature f^1 is *contributing*, as the mapped region overlaps with W, while f^2 is not. The actual contribution of f^1 is the maximal value within W's bins falling into the green region. The axis-aligned extension (*blue rectangle*) allows for fast evaluation of a bound on this maximum using integral images

4.3 Maximum Query Using Integral Images

A possible method to efficiently process the maximum queries of (9) is by means of integral images. We discretise the weight function W and represent it by a spatial histogram. This is a common setup (Dalal and Triggs 2005; Felzenszwalb et al. 2008; Lampert et al. 2009) where learning can be accomplished using discriminative methods (*e.g.* linear SVMs). The histogram assumption lets us rephrase the geometric interpretation: The hypothesis set Ω maps to a region $\bar{\Omega}$ which intersects with some bins of the histogram (*c.f.* Fig. 7). Thus, we can think of $W|_{\bar{\Omega}}$ as a set of values, *i.e.*, the weights of the intersecting bins. The task is then to efficiently find the maximum of these values or an upper bound, respectively. In this work, we construct an upper bound for the maximum using mean μ and variance σ^2, *i.e.*,

$$\begin{aligned}\max W|_{\bar{\Omega}} &= \mu(W|_{\bar{\Omega}}) + \big(\max(W|_{\bar{\Omega}}) - \mu(W|_{\bar{\Omega}})\big) \\ &\leq \mu(W|_{\bar{\Omega}}) + g(|\bar{\Omega}|)\sigma(W|_{\bar{\Omega}}) \qquad (10)\end{aligned}$$

with a fixed correction factor $g(n)$ and $n = |\bar{\Omega}|$ the number of intersecting bins. The worst-case is that all values but the maximal one are equal which gives rise to the correction $g(n) = \sqrt{n-1}$.[8] In general, this is overly pessimistic and leads to slow convergence. Due to smoothness in W, this worst-case is very unlikely and approximations are discussed in the experiments (Sect. 4.6). As the region $\bar{\Omega}$ has a complicated boundary (green polygon in Fig. 7), we slightly loosen the bound by expanding the region to the smallest enclosing rectangle. Hence, keeping integral images for $F = \int W$ and $F2 = \int W^2$ allows for computing $\mu = F(\bar{\Omega})/|\bar{\Omega}|$ and $\sigma^2 = F2(\bar{\Omega})/|\bar{\Omega}| - \mu^2$ efficiently. These integral images are defined on the scene-independent invariant space (*c.f.* Fig. 1) and depend only on the weight function (and not on the observed image). Thus, they can be pre-computed *during learning* (*c.f.* Fig. 2) and are only as large as the weight function itself. This is an important difference to the original ESS approach and is a consequence of the feature-centric view. Hence, although we also rely on integral images, our approach differs from ESS in various ways. We will briefly revisit ESS's bound in order to discuss these differences thereafter. For a detailed description of ESS, we refer to the original authors' papers (Lampert et al. 2008, 2009).

[8] Without loss of generality, assume the maximum is M and all other values are $-M/(n-1)$ which yields $\mu=0$ and $\sigma^2=(1\cdot M^2+(n-1)\cdot M^2/(n-1)^2)/n = M^2/(n-1)$. Thus, $g=(\max-\mu)/\sigma=\sqrt{n-1}$.

4.4 Efficient Subwindow Search (ESS)

A central aspect of ESS's bound is to split the weights into positive and negative ones, *i.e.*, $W = W^+ - W^-$ with $W^\pm > 0$. Moreover, the weights are represented by histograms. The following description is accompanied by the visualisation in Fig. 8. Let us denote a (spatial) bin of a histogram by b and, its back-projection onto the image grid (given a hypothesis λ) by $b(\lambda) = \{f \mid \mathbb{I}(\lambda, f) \in b\}$, *i.e.*, all features that fall into bin b. ESS rewrites the score (5) as

$$\begin{aligned}S(\lambda) &= \sum_{f\in\mathcal{F}} f_m W(f_c, \mathbb{I}(\lambda, f)) \\ &= \sum_b \sum_{f\in\mathcal{F}} f_m [W^+(f_c,b) - W^-(f_c,b)] \mathbb{1}_{\mathbb{I}(\lambda,f)\in b} \\ &= \sum_b Q_b^+(b(\lambda)) - \sum_b Q_b^-(b(\lambda)), \qquad (11)\end{aligned}$$

where $Q_b^\pm(q) = \sum_f [f_m W^\pm(f_c,b)] \mathbb{1}_{f\in q}$ and $\mathbb{1}$ is an indicator which is 1 if $f \in q$ and 0 otherwise. The change of the indicator's index follows from the equivalence $f \in b(\lambda) \Leftrightarrow \mathbb{I}(\lambda, f) \in b$. The expressions $Q_b^\pm$ are (weighted) range sum queries. They can be evaluated in constant time using integral images if the query region $q := b(\lambda)$ is rectangular. However, this efficient evaluation procedure makes an unfavourable trade-off, since the integral images are defined on the scene-dependent feature space (*i.e.* the image plane, *c.f.* Fig. 1). Firstly, they are as large as the input image. Secondly, there are two of them per spatial histogram bin, *i.e.*, one for each $Q_b^\pm$. Thus, memory requirements scale with the product of image size and spatial histogram resolution. Last but not least, the integral images must be computed during detection (*c.f.* Fig. 2) as they depend on the observed image.

ESS's Upper Bound. The upper bound of ESS is obtained by considering upper bounds on Q_b^+ and lower bounds on Q_b^-. The upper bound on Q_b^+ is achieved by counting all features that fall into bin $b(\lambda)$ for *at least one* hypothesis $\lambda \in \Omega$. Hence, Q_b^+ is queried with the union $q_b^+ :=$

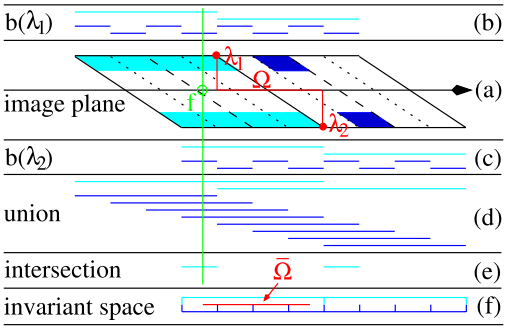

Fig. 8 (Color online) 1D illustration of ESS's bound computation. The setup shows a shift-invariant detector with a fine (*dark blue*) and a coarse histogram (*light blue*). For visibility, we show just one single feature f (*green*) and consider the set of hypotheses $\Omega = [\lambda_1 \dots \lambda_2]$ (*red*). (**a**): Alignment of the spatial histogram with the image plane (*i.e.*, classical sliding window thinking). (**b**, **c**): Projection of the bins b onto the image plane for hypothesis λ_1 and λ_2, resp. (**d**, **e**): Bin union q_b^+ and intersection q_b^- of all tested hypotheses, *i.e.* the query regions for the positive (negative) weights. Note that the intersections are empty for the fine histogram. (**f**): The invariant space along with the mapped hypothesis set $\bar{\Omega}$ (*red*). As $\bar{\Omega}$ is entirely within bin one of the coarse histogram, the feature falls into the corresponding intersection query region (*i.e.*, q_1^-)

$\bigcup_{\lambda\in\Omega} b(\lambda)$.[9] In the case of Q_b^-, the feature has to fall into $b(\lambda)$ for *all* $\lambda \in \Omega$. Thus, the query for Q_b^- is just the intersection $q_b^- := \bigcap_{\lambda\in\Omega} b(\lambda)$, which may be empty. Therefore, the bound computed by ESS is

$$S^{ESS}(\Omega) = \sum_b Q_b^+ \left(\bigcup_{\lambda\in\Omega} b(\lambda) \right) - \sum_b Q_b^- \left(\bigcap_{\lambda\in\Omega} b(\lambda) \right). \tag{12}$$

Comparison. We now compare our bound (*i.e.*, (9, 10)) to the one of ESS (12). Contrary to ESS, our integral images do *not* depend on the size of the test image, but only on the discretisation of the weight function W. Thus, they use less memory and can be *pre-computed offline*, while ESS has to compute them on-line prior to the actual search (*c.f.* Fig. 2). A second difference concerns the evaluation cost of the bounds: while ESS needs two integral image evaluations per (spatial) *bin*, our bound requires two evaluations per (contributing) *feature*. As a consequence, we refer to ESS as the *bin-centric* bound, and to ours as the *feature-centric* one. Depending on the choice of features and the resolution of the spatial histogram, one or the other may be advantageous. We want to emphasise however that one should not neglect the cost for computing the integral images in the case of ESS. Comparison of memory usage is postponed to Sect. 4.6, where we show our feature-centric system can get along with 25 × less memory than traditional ESS. Last but not least, there is the question about the quality of these bounds. This is a crucial aspect as it strongly affects the convergence speed of branch and bound. Hence, its analysis is important. A theoretical comparison is given in the next paragraph, while a quantitative evaluation is shown in the experiments.

Feature-Centric View of ESS. Interestingly, ESS's bound can be expressed within our feature-centric framework. This makes a direct comparison possible. Recall the geometric interpretation: $\bar{\Omega}$ denotes the mapping of a hypothesis set Ω into the invariant space given a feature f. Then, this feature falls into the following query regions (*c.f.* Fig. 8): $f \in q_b^+$ if $b \cap \bar{\Omega} \neq \emptyset$ and $f \in q_b^-$ if $\bar{\Omega} \subset b$. The latter implies that $\bar{\Omega}$ does not intersect with any other bins, as all bins are non-overlapping. Hence, the contribution of such a feature is simply the weight of that single bin, no matter if it is positive or negative. If $\bar{\Omega}$ covers $n > 1$ bins, the contribution to the bound is the sum of all positive weights, *i.e.*, $\sum_{b\cap\bar{\Omega}\neq\emptyset} W^+(f_c, b) = n\mu(W^+|_{\bar{\Omega}})$. Thus, ESS's bound is equivalent to using

$$\max_{\lambda\in\Omega} W = \begin{cases} n \cdot \mu(\max(W|_{\bar{\Omega}}, 0)), & n > 1, \\ W(f_c, B), & n = 1 \end{cases} \tag{13}$$

(instead of (10)), where B denotes the single bin if $n = 1$. There are various points worth noting. First of all, if $n > 1$, the negative weights (*i.e.*, penalties) are completely ignored. They affect the bound only close to convergence. Thus, too fine a discretisation of the histogram has negative effects in ESS: it makes the evaluation more expensive and negative contributions set in later. Note that the discretisation has no effect on our bound. Moreover, ESS only uses the mean (plus point evaluations if $n = 1$), while our bound incorporates mean *and* variance. Thus, we expect our bound to be more powerful. A quantitative comparison will be given in Sect. 4.6.

4.5 Implementation

We now give implementation details of our algorithm, which is to be seen mainly as an illustration of the feature-centric ESS idea, a key contribution of this work. As mentioned earlier, our system is similar to ESS, as we use a histogram representation as well as the object-centric invariant $\mathbb{I}_\lambda$.[10] Furthermore, we also employ branch and bound to search for objects, but use the feature-centric bound (9, 10) instead of ESS's bin-centric one (12). Consequently, our approach avoids any online pre-processing, which is a clear advantage. In the sequel, we discuss the following three aspects.

[9] Actually, the smallest enclosing rectangle is used to make fast integral image evaluation possible.

[10] The drawbacks of $\mathbb{I}_\lambda$ indicated in Sect. 3.2 do not apply here as we neither perform explicit voting nor do we compute score-gradients.

Fig. 9 (Color online) Adaptive subdivision of the search space. The object center serves as hypothesis parametrisation. Each *yellow box* corresponds to a set of hypotheses (which comprises objects whose center is within the box). The discretisation around object centers is much finer than on background regions

Firstly, we briefly revisit the branch and bound algorithm and explain the subdivision strategy. Secondly, the handling of contributing features is explained, which is an important step to avoid unnecessary computations. Finally, we discuss how our system is extended to properly detect multiple objects in an image.

Branch and Bound. Branch and bound adaptively subdivides the search space. It keeps a priority queue, each element of which represents a set of hypotheses. Initially, the whole search space is entered into the queue as a single element. A bound is used as ordering criterion to gradually split the most promising hypothesis set into two halves. We split along the dimension with the largest extent. This leads to a kD-tree like partitioning of the search space which enables sub-linear search time. Each node in the queue corresponds to a leaf node of the kD-tree (yellow boxes in Fig. 9). Eventually, the size of a hypothesis set becomes very small, *i.e.*, we converge to a single object which is reported as detection. Note that the dimensions of the mapped hypothesis set $\bar{\Omega}$ (using $\mathbb{I}_\lambda$) can be interpreted as a measure of localisation precision (relative to the object size). Thus, a natural scale-invariant convergence criterion is to stop whenever the extent of $\bar{\Omega}$ in each dimension is less then a threshold (*e.g.* 0.01).

Contributing Features. An important aspect for efficiency is to ignore non-contributing features, *i.e.*, those where $\bar{\Omega}$ falls completely outside the discretisation range of W (see Fig. 7). The sub-divisive nature of the algorithm enables us to efficiently determine such features: if a feature is not contributing to a set Ω, it will not contribute to any subset $\Omega' \subset \Omega$. Thus, we keep a list of active features for every leaf node in the kD-tree. On each split, we have to determine the features which no longer contribute and remove them from the list (of the new leaves). Such lists were called *match-lists* in previous work on geometric matching (Breuel 1992, 2002; Keysers et al. 2007).

Multiple Objects. As detailed in Sect. 2.5, we stick to a rather simple approach to detect multiple objects. We limit each feature to explain only one single object. This leads to a greedy matching pursuit like algorithm (Mallat and Zhang 1993): We seek for the best object hypothesis and eliminate all features which contribute positively to it. Then, the search for more object is restarted. Thus, upon each restart, the list of active features and the bound of every node needs to be updated. This can be done efficiently by recording not only the active features, but also their actual contribution (*i.e.*, 1 int + 1 float = 8 Bytes per feature). Hence, updating involves no new integral image computations. Such feature removal is not easily possible in ESS (Lampert et al. 2009), as all integral images would have to be recomputed from scratch.

4.6 Experiments & Discussion

We evaluated our algorithm on two benchmark datasets. As in Lampert et al. (2009), we use the UIUC cars database (Agarwal et al. 2004) and focus on the multi-scale subset. It consists of car side-views with 1050 training images and 107 test images (containing 139 cars). Moreover, we consider the TUD motorbikes dataset (Fritz et al. 2005) which includes 153 training images and 115 test images. In order to compare our results to the published baseline performance, we use the same evaluation criterion as in Agarwal et al. (2004), Fritz et al. (2005). Both datasets provide ground truth annotation and evaluation software. Detections are accepted as correct if the overlap of their bounding box and the ground-truth annotation is above 50%; multiple detections on the same object are counted as false positives. Here, we use SURF (Bay et al. 2008) features and set $f_m = 1$. The model weights W are learned using Support Vector Machines (SVMs), which we will discuss in more detail later in Sect. 5.

Recognition Performance. For UIUC cars, the equal error rate of our systems is 2.2%, which compares well with the results in the literature (ESS 1.4% (Lampert et al. 2009) and ISM 5% (Leibe et al. 2008)). On TUD motorbikes we achieve an equal error rate of 19%, *i.e.* close to the 18% reported by the original authors (Fritz et al. 2005). Although we demonstrate state-of-the-art performance on both datasets, we would like to emphasise that the contribution of this section is about efficient search, and *not* about learning robust models, *e.g.* (Blaschko and Lampert 2008). In the sequel, we focus on the analysis of runtime properties and on the comparison to ESS (Lampert et al. 2009).

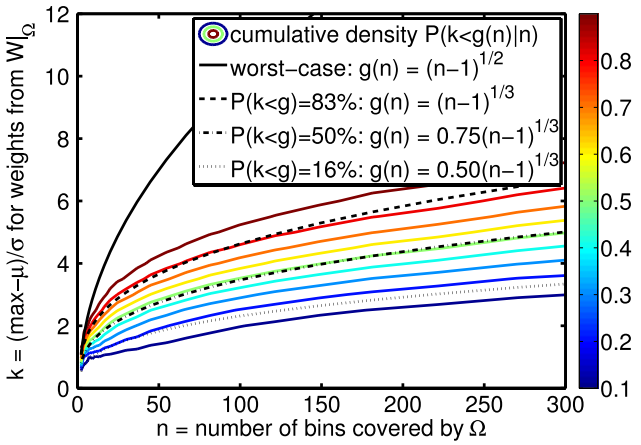

Fig. 10 (Color online) Confidence $P(\frac{\max-\mu}{\sigma} < g(n)|n)$ in the bound (10) given $g(n)$. For fixed n, we can estimate this cumulative density by examining all possible rectangular sub-regions of W consisting of n bins. The iso-probability lines (as function of n) are colour coded with high (low) confidence in *red* (*blue*). (*Black*, *solid*): The rather loose worst-case upper bound $\sqrt{n-1}$. (*Black*, *dotted*): The correction term $\alpha\sqrt[3]{n-1}$ approximates the iso-probability lines rather well. Thus, its likelihood is almost independent of n. Best viewed in colour

Approximate Correction Term. As mentioned in Sect. 4.3, the worst-case correction term $g(n) = \sqrt{n-1}$ is very pessimistic. It completely ignores the dependency of spatially adjacent histogram bins. We now seek a more optimistic choice which accelerates convergence, but does not degrade performance. Given a learned model, we can estimate the probability that some $\tilde{g}(n) < g(n)$ leads to a correct upper bound. Figure 10 shows the estimation for our UIUC cars model. Clearly visible, the guaranteed bound obtained with $\sqrt{n-1}$ is loose, especially for larger n. Interestingly, the correction term $\tilde{g}(n) := \alpha\sqrt[3]{n-1}$ seems to approximate the iso-probability lines (of the UIUC cars model) rather well. Thus, the confidence in such a "probabilistic bound" is (almost) independent of n, which is why we choose $\tilde{g}(n)$ for all following experiments. Choosing an appropriate α will be discussed in the next paragraph. Plugging all elements together, the complete, feature-centric bound of our system reads as

$$S^{FC}(\Omega) = \sum_{f\in\mathcal{F}} f_m \left[\mu(W|_{\bar{\Omega}}) + \sigma(W|_{\bar{\Omega}})\alpha\sqrt[3]{|\bar{\Omega}|-1}\right]. \quad (14)$$

Runtime Comparison. We evaluate the probabilistic bound in terms of quality and speed. The quality is measured in terms of performance at equal error rate, while speed is measured in number of iterations relative to the baseline. As baseline we consider a system which has access to the true maximum of (10). The results for both datasets are reported in Fig. 12. As the behaviour is similar for both datasets, we limit our discussion to UIUC cars (*i.e.* Fig. 12(left)). For comparison, we emulate the bound of ESS in our framework. ESS performs as well as the baseline, but needs about

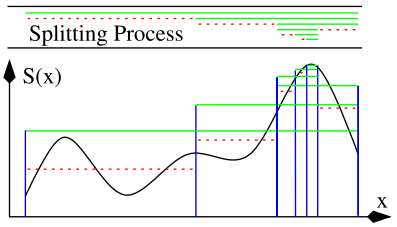

Fig. 11 (Color online) Branch and bound search using the mean (*i.e.*, $g(n) = 0$) instead of an upper bound. The splitting process is shown on *top*: *green solid lines* represent hypotheses sets which are refined, while *red dotted lines* have low priority and are not split any further. The actual priority (*i.e.*, mean value) of each set is shown in the *bottom graph*. Here, the search converges to the best maximum, but there is no guarantee for this behaviour

1.7× more iterations. On the other hand, our probabilistic bound with $\alpha \approx 0.82$ requires the same number of iterations as the baseline. This choice corresponds to a bound guarantee of about 60%. Thus, the over-/under-estimation roughly compensate, which intuitively explains why there is no loss in performance. However, the performance at equal error rate is stable for values as low as $\alpha = 0.2$. Thus, we have no loss of performance for $\alpha > 0.2$. For a safe choice of $\alpha = 0.3$, the number of iterations is only 6% compared to the baseline. At this setting, the average number of iterations is 185, and our Matlab implementation[11] detects objects in about 0.8 s. It is rather astonishing that there is no loss in performance, as such a bound holds with very low probability. We provide two arguments which may explain this phenomenon. First of all, we ignored the inequality of (9) in our reasoning. This compensates for a slight underestimation of the max term (10). Secondly, branch and bound does not break down completely if we use a *priority function* which is *not* an upper bound. Figure 11 gives a simple example to reinforce the claim that a weaker criterion may be sufficient, but further investigations are necessary.

Scaling Behaviour. To demonstrate that our algorithm scales with the amount of information rather than the image resolution, we show the impact on runtime when the images are upscaled. Figure 13 reports the increase of iterations relative to the unscaled image for scale factors of 1 to 4 (averaged over the whole dataset). As reference, we show the growth inherent in an exhaustive, sliding-window based search, which is linear in the number of pixels and thus quadratic in the scale factor. As can be seen, the impact on our adaptive search is significantly lower. In general, we believe that our algorithm exhibits very natural scaling behaviour. Both runtime *and* memory consumption scale sub-linearly with the size of the entire hypothesis space $|\Lambda|$

[11] Available at www.vision.ee.ethz.ch/lehmanal/iccv09.

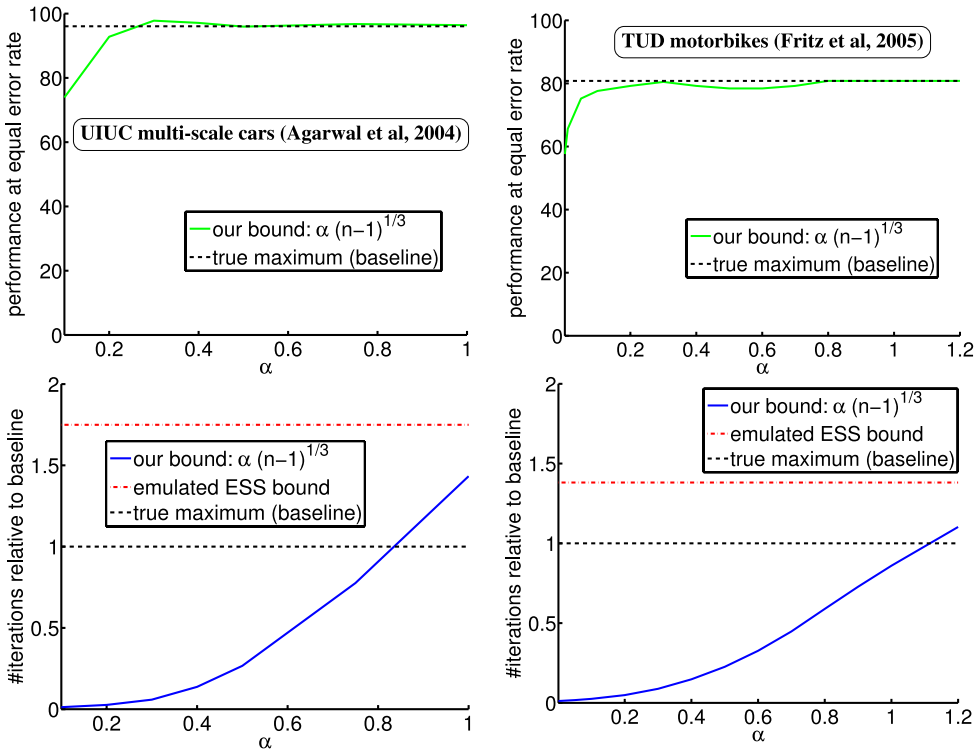

Fig. 12 (Color online) Evaluation of the probabilistic bounds in terms of performance at equal error rate (*top*), and number of iterations relative to the baseline (*bottom*). As baseline we consider a system which has access to the true maximum in (10). The results are shown for UIUC multi-scale cars (*left*) and TUD motorbikes dataset (*right*). From the top plots we see that our probabilistic bound performs well for values $\alpha > 0.2$ and breaks down for smaller values. For a safe choice of $\alpha = 0.3$, the number of iterations is just about 6% (*left*) and 8% (*right*) compared to the baseline system. In comparison, ESS's uses a true bound which converges slower than the baseline by a factor of about 1.7 and 1.5, respectively

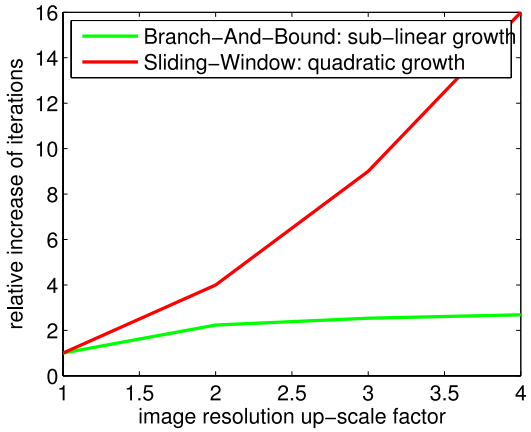

Fig. 13 (Color online) Runtime/memory dependence on image resolution: Our adaptive search (*green*) versus quadratic growth of classical sliding-window search (*red*)

and linearly with the (average) number of (contributing) features $\bar{F}$. Formally, that is $O(sl(|\Lambda|) \cdot \bar{F})$ where $sl(\cdot)$ denotes sub-linearity. This feels natural as the number of features is affected by the visual content, *i.e.*, more complicated scenes require more work.

Interestingly, neither the model size (*i.e.*, number of bins) nor the image resolution affect runtime directly. Both are desirable properties as increasing the spatial resolution of the object model should not make the localisation task any harder. The same argument holds when we get a higher resolution image for the same image content. The additional information should not impair the algorithm's runtime. However, ESS's runtime (Lampert et al. 2009; An et al. 2009) does depend on both these parameters due to a necessary pre-processing step. To be fair, both approaches rely on feature extraction which scales with the image resolution. Parallel architectures like GPUs enable us to extract features very quickly (*e.g.* Cornelis and Van Gool 2008), though, and this initial cost is alleviated when thinking about multi-class setups where features are shared among different classes.

Memory Comparison. Both ESS and our approach use integral images. However, there is a crucial difference due to the feature-centric versus bin-centric definition of the score bound. In our system, the integral images depend only on the model, while in ESS, they depend on the model *and* the observed image jointly. Thus, our integral images can be computed off-line and detection directly starts with the adaptive search. In contrast, ESS needs an (unadaptive) online pre-processing (to build the integral images) prior to the search. Moreover, the memory consumption of these integral images is much higher. Considering the experiment from Fig. 13, the $4\times$ scaled images have a size of 860^2 on average. The memory requirement of ESS (assuming a 10×10 histogram) scales with the input image size and would be $860^2 \times 10^2 \times 2 \times 4$ B ≈ 564 MB. On the other hand, our system stores 2×4 Bytes per contributing feature and it requires about 1.8×10^6 integral image evaluations. Thus we use roughly 20 MB, *i.e.* 25 times less memory.

True Scale-Invariance. An implication of the memory bottleneck is that ESS uses only 2D (instead of 3D) integral images for its range queries. This is an issue as features live in a 3D scale-space (Lindeberg 1994) and not only in the 2D image plane. Using only 2D integral images corresponds to using $\mathbb{I}_\lambda$ and a histogram which extends infinitely in the scale-ratio dimension ($\mathbb{I}_{\lambda,s}$). That implies that the feature scale f_s is completely ignored. This is a limitation, as assigning different weights to (relatively) large-/small-scale features is desirable. In particular, setting the weight to zero is important to ignore too small/large features. Otherwise, detection at a larger scale will have more contributing features. This may result in a tendency for higher scores and thus a bias towards larger detections. Properly accounting for the feature scale would require discretising the feature scale-space. That would increase the memory consumption even further. Dealing with the feature scale properly is not a problem in our framework and there is no such bias towards larger detection windows. Thus, in contrast to ESS, our approach makes no further model restrictions than the ones of the Hough transform, *i.e.*, the linear model constraint. In the experiments we actually use a histogram with $10 \times 10 \times 3$ bins.

Multi-Class. Finally, assume a multi-class setup. That is accomplished by adding an additional *class* dimension to the weight function W. Consequently, the footprint maps each feature f to a Dirac pulse at the $(4+1)$D-point $[f_c, \mathbb{I}(\lambda, f), \lambda_c]$, where λ_c denotes the class of an object λ. Apart from minor changes, the algorithm remains the same. It processes all classes simultaneously and thus still scales sub-linearly, *i.e.*, $O(sl(|\Lambda|C)\bar{F})$, where C denotes the number of classes (and views). Hence, the number of classes affects runtime/memory only through the size of the (multi-class) search space, where we expect sub-linear scaling. In contrast, runtime of ESS scales as $O(|I|BC + sl(|\Lambda|C)B)$, where B denotes the number of spatial bins. Thus, the preprocessing introduces a linear dependency (in C) which cancels the sub-linear behaviour of the actual search. Moreover, it emphasises the memory bottleneck since memory usage scales with $O(|I|BC + sl(|\Lambda|C))$. We conclude that ESS may be faster when lots of features are extracted, but its memory trade-off gets prohibitive when it comes to multi-class/multi-view setups.

5 Model Regularisation for Fast Detection

The previous section dealt with efficient object search where we showed the advantages of PRISM's feature-centric view. However, there are also considerations during learning which allow for accelerating object detection. More precisely, we show that certain computations during detection lead to model regularisation. Such computation can thus be avoided if sufficient regularisation has already been applied to the model during training.

Two examples of such computations are spatial pyramid descriptors and soft-matching. These two techniques are commonly used and, as acknowledged by many authors (Leibe et al. 2008; Lazebnik et al. 2006; Ommer and Buhmann 2007; Philbin et al. 2008; Ferrari et al. 2007; Grauman and Darrell 2005), improve detection quality. Unfortunately, their downside is an increase of computation cost during detection where speed is of prime importance. We claim that this may well be unnecessary and show how to avoid it. As already mentioned, the effect of both can be interpreted as model regularisation, which is a concept of learning. As such, it should be applied during learning only, thereby making fast nearest neighbour matching and flat histogram representations during detection possible.

5.1 Soft-Matching

Soft-matching is a heuristic where a feature activates multiple codewords from the visual vocabulary, instead of just the best-matching one. We now show how to move soft-matching to the training stage, thus allowing for NN-matching during detection, where runtime is of prime importance.

In PRISM, a footprint which accounts for soft-matching has multiple Dirac pulses (per feature) instead of a single one (at the best-matching visual word, *c.f.* (4)). Each Dirac pulse is weighted according to the quality of the match. This weight decreases as a function of the distance of the feature descriptor from the visual word representative. That allows us to interpret soft-matching as a diffusion which blurs the Dirac pulse of the best-matching visual word. Hence, soft-matching can be implemented in PRISM by a blurred footprint $D\phi$ where D denotes the blurring operator.

However, instead of applying the blur to the footprint, we may equally well apply it to the model, *i.e.* $\langle D\phi, W\rangle = \langle \phi, D^T W\rangle$. The latter incorporates soft-matching completely into the training stage (where we can pre-compute $D^T W$) and avoids it during detection. More precisely, we apply stronger blurring during learning and NN-matching during recognition. Hence, the system benefits from soft-matching's robustness without having the computational overhead during recognition. We validate this claim by an experiment with our GMM detector on the motorbikes database (as described in Sect. 3.4). In Fig. 14, we see the result of using $\{1, 3, 5\}$-NN during detection in addition to 6-NN during learning. Additional soft-matching clearly degrades the performance. In other words, applying soft-matching twice leads to an over-smoothing which causes under-fitting.

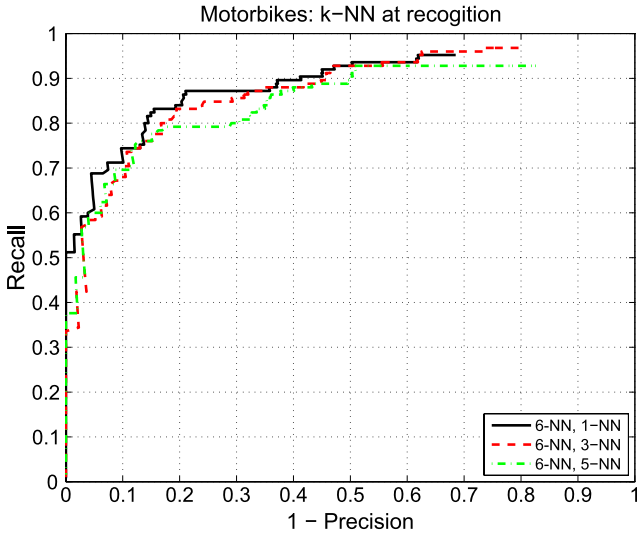

Fig. 14 (Color online) The consequence of applying additional soft-matching during recognition. During learning, the soft-matching is fixed to 6-NN matching while we vary between 1 to 5 NN during recognition. The performance clearly drops when deviating form the hard-NN matching rule

Benefits. Moving soft-matching entirely to the learning stage has clear advantages. On the one hand, NN-matching during recognition is much simpler and faster. In the above example, changing from 5-NN to 1-NN yields a speedup of approximately 4. On the other hand, stronger soft-matching during learning causes more occurrences which makes density estimation more stable. Contrary to non-parametric density estimators (Leibe et al. 2008), our GMM compresses the increased number of occurrences. Hence, the overhead of soft-matching during recognition is very low. In summary, soft-matching causes model regularisation which is controlled by the degree of soft-matching. Moving it entirely to the training stage makes fast nearest-neighbour matching during detection possible.

5.2 Spatial Pyramid Descriptors

Spatial pyramid descriptors (Grauman and Darrell 2005; Lazebnik et al. 2006) consist of multiple histogram layers, each of which has a different spatial resolution. We can avoid such pyramids during detection by a similar argument as before. Taking an approach like ESS (Lampert et al. 2009), this leads to significant memory savings (*c.f.* footnote 1), but also runtime benefits.

As a matter of fact, certain multi-layer pyramid representations $\tilde{x}$ can be computed from a *flat* histogram x by means of a mapping $P : x \mapsto \tilde{x}$.[12] For example, consider a 1D histogram with 4 bins, *i.e.*, $x \in \mathcal{R}^4$. The corresponding pyramid with 1, 2, and 4 bins per level can be computed by the transform

$$\tilde{x} = Px \in \mathcal{R}^7 \quad \text{with } P^T = \left(\begin{array}{c|cc|cccc} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right). \tag{15}$$

Such mappings exist if the bin boundaries of coarser histograms coincide with those of the highest resolution (flat) histogram x. This seems to be a restriction, but the point is that a more general pyramid (*e.g.* with 1, 2, 3, and 4 bins per layer) effectively samples at a higher resolution than just 4 bins, which is not necessarily desired. We now assume that the flat histogram x has the highest *effective* histogram resolution (*i.e.*, considering all bin boundaries arising in the pyramid), in which case the above mapping exists. Anyway, the explicit mapping is only needed for the following argumentation and our final learning algorithm (17) allows for more general regularisation strategies as we will demonstrate in Sect. 5.4.

As in the last subsection, instead of computing the pyramid representation for each hypothesis, it is possible to apply P^T once to the pyramidal model $\tilde{w}$,[13] *i.e.*, $\langle Px, \tilde{w}\rangle = \langle x, P^T\tilde{w}\rangle$. This is computationally attractive as it is done once, off-line during training. The back-transform $P^T\tilde{w}$ compresses the learned model back into a lower-dimensional, flat histogram. Thus, the pyramid is *not* needed during detection which reduces algorithmic complexity and runtime.

5.3 Explicit Regularisation

Interestingly, we can avoid the pyramid representation altogether, *i.e.* we do not have to learn a pyramid representation, followed by computing its low-dimensional counterpart (*i.e.*, $w = P^T\tilde{w}$). Instead, we integrate P into the learning algorithm which then directly returns the flat histogram w. This enables us to exploit the structure of P during training and it will make the connection to regularisation explicit.

Let us start with the linear SVM optimisation problem for the pyramid representation. The training data consists of a label $y_i \in \{-1, +1\}$ and a pyramid descriptor $\tilde{x}_i = Px_i$ for each sample i. Then, the pyramid model $\tilde{w}$ is obtained by solving

$$\max_{\tilde{w},\xi_i} \|\tilde{w}\|^2 + \sum_i \xi_i \quad \text{s.t.} \quad \forall i : y_i \tilde{w}^T \tilde{x}_i \geq 1 - \xi_i,\ \xi_i \geq 0 \tag{16}$$

where ξ_i are the usual, per-sample slack variables.

[12] To avoid confusions later on, we use the symbol x for the footprint instead of ϕ, as the latter has a different meaning in the SVM literature.

[13] We use lower case letters $w, \tilde{w}$ to denote the discretised histogram representation of the continuous weight function W defined in Sect. 2.

It is well-known that the dual of (16) can be written entirely in terms of inner products of the samples, which allows for the kernel trick. Actually, our particular case, *i.e.*, $\tilde{x}_i^T \tilde{x}_j = x_i^T P^T P x_j = \kappa(x_i, x_j)$, can be interpreted in the dual as a kernelised SVM working on the flat histograms x_i. This so-called semantic similarity kernel (Shawe-Taylor and Cristianini 2004) is defined by a similarity matrix $M = P^T P$ which is positive (semi-)definite. Thus, without loss of generality, P can be thought to be the Cholesky factor of M, *i.e.* a square matrix (instead of rectangular, *c.f.* (15)). We now assume that M (and thus P) are also invertible. Therefore, the model post-processing from the last section $w = P^T \tilde{w}$ can be inverted, *i.e.*, $\tilde{w} = (P^T)^{-1} w$. Applying this variable substitution to the regularisation term yields $\tilde{w}^T \tilde{w} = w^T P^{-1} (P^T)^{-1} w = w^T (P^T P)^{-1} w = w^T M^{-1} w$, and the data term simplifies. Hence, one obtains an equivalent optimisation problem in terms of the low-dimensional, flat histogram variables x_i, *i.e.*

$$\max_{w,\xi_i} w^T M^{-1} w + \sum_i \xi_i$$
$$\text{s.t.} \quad \forall i : y_i w^T x_i \geq 1 - \xi_i, \quad \xi_i \geq 0. \tag{17}$$

The difference to ordinary SVMs is the modified regularisation term which causes the positive effects of the pyramid representation. This thus demonstrates that pyramids (and similarly soft-matching) can be interpreted as model regularisation. Moreover, (17) can be solved efficiently in this primal form (Chapelle 2007; Sandler et al. 2008), which directly yields the desired flat histogram w (instead of $\tilde{w}$).

Discussion. The optimisation problem (17) bears various advantages over mapping to the pyramid representations explicitly (*i.e.* (16)) or applying the semantic similarity kernel in the dual. On the one hand, explicit feature mapping increases the dimensionality and breaks the sparsity inherent in bag-of-visual-words representations. This increases memory usage and computation time. On the other hand, solving the kernelised dual formulation is discouraging for two reasons. Firstly, the evaluation of the semantic kernel is quadratic (instead of linear) in the number of (non-zero) elements. Secondly, optimisation of linear SVMs in the primal has been reported to be faster than solving the dual problem (Chapelle 2007). In summary, solving (17) directly avoids the pyramid representation altogether without losing its regularising effect. Moreover, it fully accounts for sparsity in the data and bypasses an increase in the representation's dimensionality.

5.4 Explicit Spatial Regularisation

In the last subsection, we showed that SVMs with the modified regularisation matrix $M^{-1} = (P^T P)^{-1}$ yield a solution equivalent to one we could have obtained by training on the pyramid descriptors. This leads to a spatially smoothed, regularised solution. In our opinion, however, the use of such pyramids is just a heuristic which showed to work well in practice. Instead (17) enables us to explicitly define the regularisation matrix $R = M^{-1}$. This not only gives better control over the regularisation, it can also lead to computational advantages in the optimisation (which we will show).

Sandler et al. (2008) also tackled this problem, but in the context of document processing with bag-of-words models. They first compute a graph over words where the edges between words account for their similarity. These similarities are estimated from a given text corpus and the regularisation matrix R is then deduced from the graph. That regularisation relates to soft-matching in our context, whereas we focus on spatial regularisation in this work. Compared to the work of Sandler et al. (2008), we are facing a much more structured problem since a 3D (x, y, scale) spatial histogram corresponds to a regular 3D grid graph. Thus, we do not have to construct a graph explicitly.

The basic idea of our spatial regularisation is simple. In addition to the ordinary SVM regularisation (*i.e.*, $\sum_i w_i^2 = w^T w$), we also penalise the difference of neighbouring histogram bin weights, *i.e.*, $\sum_{i \sim j} \|w_i - w_j\|^2$ where $\sim$ denotes neighbouring bins. The new term can be written in matrix form as $w^T L w$ where L denotes the graph Laplacian matrix. As we will see later, this matrix is highly structured and very sparse in our 3D grid case. Actually, minimising such a Laplacian term leads to a heat diffusion like process (Perona and Malik 1990) where positive (negative) training samples correspond to heat sources (sinks).

Simply taking a weighted combination of the two regularisations, *i.e.*, $R := \alpha I + \beta L$, is possible but not recommended. In that form, the mixing factors α and β would depend on the histogram resolution, *i.e.*, the number of bins. This is undesirable and can be avoided rather easily. Recall that the histogram w is a discretisation of the continuous weight function W from Sect. 2. Consequently, the regularisation matrix defined above can be derived as a finite-differences implementation of

$$\int_{\mathcal{R}^3} \alpha \|W\|^2 + \beta \|\nabla W\|^2 dV \tag{18}$$

where the integration is over the invariant space and ∇ denotes the gradient operator. Defining the mixing factors α and β in this continuous formulation leads to the desired resolution-independence. Let us assume a discretisation grid with N_x, N_y, and N_s bins, respectively. This yields a total of $N = N_x N_y N_s$ bins with volume $dV = N^{-1}$. Moreover, the gradient is approximated by *e.g.* $(w_i - w_j)/\delta_x$ where $\delta_x = N_x^{-1}$ is the inter-bin distance. Taking the square leads to $N_x^2 (w_i^2 - 2 w_i w_j + w_j^2)$. Thus, there is a correction factor N_x^2 which is probably hard to guess without considering the continuous form (18). Furthermore, to better

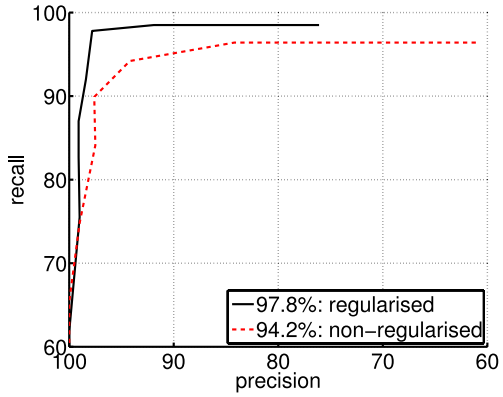

Fig. 15 (Color online) Precision-recall curve on the UIUC cars database with and without spatial regularisation

cope with the different nature of each dimension, we use a per-dimension weighting factor $\beta_{x,y,s}$. Thus, our final, resolution-independent regularisation matrix reads as

$$R_{ij} = \begin{cases} \frac{\alpha}{N} + 2\left(\frac{\beta_x N_x^2}{N} + \frac{\beta_y N_y^2}{N} + \frac{\beta_s N_s^2}{N}\right), & i = j, \\ -\frac{\beta_d N_d^2}{N}, & i \overset{d}{\sim} j, \\ 0, & \text{else}, \end{cases} \tag{19}$$

where $\overset{d}{\sim}$ denotes neighbours along dimension $d \in \{x, y, s\}$. Compared to a dense matrix (*e.g.* $(P^T P)^{-1}$) with N^2 elements, this matrix has only $7N$ non-zeros, which saves computation time.

We use the UIUC cars database (*c.f.* Sect. 4.6) to experimentally validate the effect of such spatial regularisation. Fig. 15 reports the performance of our UIUC cars detector when trained with and without spatial regularisation. In this experiment, we first fixed $\beta = 0$ (*i.e.*, no spatial regularisation) and optimised α for optimal detection performance. Then, we kept this optimal α fixed and adjusted β. As can be seen, adding spatial regularisation results in a clear improvement of the precision-recall curve. The performance at equal-error rate increases by 3.6%.

Summarising this section, we showed that soft-matching and spatial pyramids can be avoided during detection without loss of performance. This leads to simpler and faster detection algorithms. The central insight is that both techniques cause model regularisation. Thus, they can—and actually should—be implemented entirely in the training procedure since regularisation is a concept of learning. Furthermore, we presented a concrete approach how to integrate spatial regularisation directly into the SVM optimisation problem. Finally, experimental results have been shown which support our claims.

6 Conclusion

The PRincipled Implicit Shape Model (PRISM) (Lehmann et al. 2009b) is a versatile framework for object detection. It overcomes various issues of ISM's probabilistic formulation and gives a sound explanation to the voting procedure. We highlighted the duality of the Hough transform and linear sliding-window detectors, which is at the core of this unifying framework. The fusion of the two paradigms becomes possible through the concept of object footprints. This footprint nicely decouples the modelling of geometric object transformations by means of invariants. Such explicit modelling of transformations becomes of particular interest in multi-view setups. The fusion of sliding-windows and the Hough paradigm lets PRISM benefit from the advantages of both, while imposing minimal model constraints. We demonstrated this flexibility with a generatively trained semi-parametric model and a discriminatively trained histogram model.

The former approach builds on Gaussian Mixture Models which allow for efficient gradient-based search. We discussed scale-adaptation and the effects of different invariants. Moreover, a simple model post-processing was shown to yield a significant speedup. With the histogram model, we showed that discriminative voting weights (trained using SVMs) are possible. This was not the case in ISM's probabilistic formulation and is a benefit of the sliding-window reasoning. In return, the Hough-inspired, feature-centric score leads to advantages in a branch and bound setting. Compared to the original Efficient Subwindow Search (ESS), our approach (Lehmann et al. 2009a) avoids any pre-processing during detection. Such on-line computation is not adaptive and may cancel the sub-linear search time. Moreover, our system uses considerably less memory and it is truly scale-invariant, while ESS was not. It further allows for efficient feature removal (which is helpful to *e.g.* detect multiple objects). We showed theoretical and practical comparisons of the two systems, which lead to the following conclusion: While ESS may be more efficient in cases with lots of features, we expect our approach to be more scalable to multiple classes and views. In summary, our system leverages elements of what is currently the state-of-the-art in sliding-window (Lampert et al. 2009) and Hough-based (Leibe et al. 2008) object detection. This fusion leads to favourable properties during detection.

A complementary contribution of this work was the study of commonly used soft-matching and spatial pyramid descriptors. They both improve detection quality, but increase computational cost during detection, where speed is of prime importance. We showed how this can be avoided. We argued that both can be interpreted as model regularisation, which clearly is a concept of learning. Consequently, they can—and should—be applied during training *only*. We

demonstrated how soft-matching and pyramid descriptors can be integrated entirely into the training procedure. In particular, spatial pyramids can be emulated by a minor change to the usual SVM regularisation term. Interestingly, such modified SVMs can be efficiently optimised in the primal form. Moving both regularisation techniques to the training stage has clear advantages. It allows for fast nearest-neighbour matching and flat histogram representations during detection without loss of recognition performance. This claim was experimentally verified. Hence, we lowered the algorithmic and computational complexity of the detector without sacrificing quality. This in turn leads to faster detection, which was the key focus of this paper.

Future work will aim at deepening the understanding of the approximate bound and improving the splitting strategy of the branch and bound algorithm. The latter becomes important in multi-class setups where different dimensions (of the search space) are not comparable anymore. In other words, how to deal with the ordinal nature of the class "dimension"? Furthermore, combining spatial smoothing with regularisation over visual words is another topic of interest. Sandler et al. (2008) already provided a solid foundation for normal (text) words, which are discrete. However, we think that the continuous nature (of the visual descriptor) underlying *visual words* could be exploited to design more sophisticated approaches.

Acknowledgements The authors wish to thank the Swiss National Fund (SNF) for support through the CASTOR project (200021-118106).

References

Agarwal, S., Awan, A., & Roth, D. (2004). Learning to detect objects in images via a sparse, part-based representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*(11), 1475–1490.

An, S., Peursum, P., Liu, W., & Venkatesh, S. (2009). Efficient algorithms for subwindow search in object detection and localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Baggenstoos, P. M. (2002). Statistical modeling using Gaussian mixtures and hmms with Matlab. Tech. rep., Naval Undersea Warfare Center, Newport, RI, http://www.npt.nuwc.navy.mil/Csf/software.html.

Ballard, D. (1981). Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, *13*(2), 111–122.

Bay, H., Ess, A., Tuytelaars, T., & Van Gool, L. (2008). Surf: Speeded up robust features. *Computer Vision and Image Understanding*, *110*(3), 346–359.

Bentley, J. (1984). Programming pearls: algorithm design techniques. *Communications of the ACM*, *27*(9), 865–873.

Blaschko, M. B., & Lampert, C. H. (2008). Learning to localize objects with structured output regression. In *Proceedings of the European conference on computer vision*.

Breuel, T. M. (1992). Fast recognition using adaptive subdivisions of transformation space. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Breuel, T. M. (2002). A comparison of search strategies for geometric branch and bound algorithms. In *Proceedings of the European conference on computer vision*.

Carreira Perpiñán, M. Á. (2000). Mode-finding for mixtures of Gaussian distributions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(11), 1318–1323.

Chapelle, O. (2007). Training a support vector machine in the primal. *Neural Computation*, *19*(5), 1155–1178.

Chum, O., & Zisserman, A. (2007). An exemplar model for learning object classes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Comaniciu, D., Ramesh, V., & Meer, P. (2001). The variable bandwidth mean shift and data-driven scale selection. In *Proceedings of the IEEE international conference on computer vision*.

Cornelis, N., & Van Gool, L. (2008). Fast scale invariant feature detection and matching on programmable graphics hardware. In *Proceedings of the computer vision and pattern recognition (CVPR) workshop*.

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Felzenszwalb, P., McAllester, D., & Ramanan, D. (2008). A discriminatively trained, multiscale, deformable part model. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Ferrari, V., Jurie, F., & Schmid, C. (2007). Accurate object detection with deformable shape models learnt from images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Fritz, M., Leibe, B., Caputo, B., & Schiele, B. (2005). Integrating representative and discriminant models for object category detection. In *Proceedings of the IEEE international conference on computer vision*.

Gall, J., & Lempitsky, V. (2009). Class-specific hough forests for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Grauman, K., & Darrell, T. (2005). The pyramid match kernel:discriminative classification with sets of image features. In *Proceedings of the IEEE international conference on computer vision*.

Heitz, G., & Koller, D. (2008). Learning spatial context: Using stuff to find things. In *Proceedings of the European conference on computer vision*.

Keysers, D., Deselaers, T., & Breuel, T. M. (2007). Geometric matching for patch-based object detection. *Electronic Letters on Computer Vision and Image Analysis*, *6*(1), 44–54.

Kittler, J., Hatef, M., Duin, R., & Matas, J. (1998). On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*(3), 226–239.

Lampert, C. H., Blaschko, M. B., & Hofmann, T. (2008). Beyond sliding windows: Object localization by efficient subwindow search. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Lampert, C. H., Blaschko, M. B., & Hofmann, T. (2009). Efficient subwindow search: A branch and bound framework for object localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *99*(1).

Lazebnik, S., Schmid, C., & Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Vol. 2, pp. 2169–2178.

Lehmann, A., Leibe, B., & Van Gool, L. (2009a). Feature-centric efficient subwindow search. In *Proceedings of the IEEE international conference on computer vision*.

Lehmann, A., Leibe, B., & Van Gool, L. (2009b). Prism: Principled implicit shape model. In *Proceedings of the British machine vision conference*.

Leibe, B., & Schiele, B. (2004). Scale-invariant object categorization using a scale-adaptive mean-shift search. In *Proceedings of the DAGM symposium*.

Leibe, B., Seemann, E., & Schiele, B. (2005). Pedestrian detection in crowded scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Leibe, B., Leonardis, A., & Schiele, B. (2008). Robust object detection by interleaving categorization and segmentation. *International Journal of Computer Vision*, *77*(1–3), 259–289.

Liebelt, J., Schmid, C., & Schertler, K. (2008). Viewpoint-independent object class detection using 3D feature maps. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Lindeberg, T. (1994). *Scale-space theory in computer vision*. Amsterdam: Kluwer Academic.

Maji, S., & Malik, J. (2009). Object detection using a max-margin hough transform. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Maji, S., Berg, A. C., & Malik, J. (2008). Classification using intersection kernel support vector machines is efficient. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Mallat, S., & Zhang, Z. (1993). Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, *41*(12), 3397–3415.

Mikolajczyk, K., & Schmid, C. (2004). Scale and affine invariant interest point detectors. *International Journal of Computer Vision*, *60*(1), 63–86.

Mikolajczyk, K., & Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*(10), 1615–1630.

Ommer, B., & Buhmann, J. M. (2007). Learning the compositional nature of visual objects. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Opelt, A., Pinz, A., & Zisserman, A. (2006). A boundary-fragment-model for object detection. In *Proceedings of the European conference on computer vision*.

Perona, P., & Malik, J. (1990). Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *12*(7), 629–639.

Philbin, J., Chum, O., Isard, M., Sivic, J., & Zisserman, A. (2008). Lost in quantization: Improving particular object retrieval in large scale image databases. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Sandler, T., Talukdar, P. P., Ungar, L. H., & Blitzer, J. (2008). Regularized learning with networks of features. In *Proceedings of the advances in neural information processing systems*.

Schneiderman, H. (2004). Feature-centric evaluation for efficient cascaded object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (Vol 2, pp. 29–36).

Schneiderman, H., & Kanade, T. (2004). Object detection using the statistics of parts. *International Journal of Computer Vision*, *56*(3), 151–177.

Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge: Cambridge University Press.

Sudderth, E. B., Torralba, A., Freeman, W. T., & Willsky, A. S. (2005). Learning hierarchical models of scenes, objects, and parts. In *Proceedings of the IEEE international conference on computer vision*.

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

Williams, C. K. I., & Allan, M. (2006). On a connection between object localization with a generative template of features and pose-space prediction methods (Tech. Rep. 0719). University of Edinburgh.

Yeh, T., Lee, J. J., & Trevor, Darrell T. (2009). Fast concurrent object localization and recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.