Feature-Centric Efficient Subwindow Search

Alain Lehmann[1]
[1]Computer Vision Laboratory
ETH Zurich
lehmann@vision.ee.ethz.ch

Bastian Leibe[1,2]
[2]UMIC Research Centre
RWTH Aachen
leibe@umic.rwth-aachen.de

Luc van Gool[1,3]
[3]ESAT-PSI / IBBT
KU Leuven
vangool@esat.kuleuven.be

Abstract

Many object detection systems rely on linear classifiers embedded in a sliding-window scheme. Such exhaustive search involves massive computation. Efficient Subwindow Search (ESS) [11] avoids this by means of branch and bound. However, ESS makes an unfavourable memory tradeoff. Memory usage scales with both image size and overall object model size. This risks becoming prohibitive in a multiclass system.

In this paper, we make the connection between sliding-window and Hough-based object detection explicit. Then, we show that the feature-centric view of the latter also nicely fits with the branch and bound paradigm, while it avoids the ESS memory tradeoff. Moreover, on-line integral image calculations are not needed. Both theoretical and quantitative comparisons with the ESS bound are provided, showing that none of this comes at the expense of performance.

1. Introduction

In this work we address the task of efficient object detection, *i.e*. joint localisation and categorisation. The task consists of two stages: learning and recognition. The former deals with estimating an accurate object model. This can be done off-line and is not time critical. The latter is concerned with the on-line application of the learned model, *i.e*., the search for objects in images. We focus on this latter problem, where speed is of prime importance.

Many object detectors are based on either the sliding-window [5, 6, 7, 17, 18] or the Hough [2, 13, 14, 16] paradigm. The former considers all possible sub-windows of an image and a classifier decides whether they contain an object of interest or not. The latter starts with feature extraction and each feature casts votes for possible object positions. Recently, the PRincipled Implicit Shape Model (PRISM) [12] has been introduced which combines these two paradigms into a single framework. The core concept which allows the fusion of the two paradigms is a (visual) object footprint, which we will revisit. As a result, PRISM brings together the advantages of both paradigms. On the one hand, it reasons in a sliding-window based manner which overcomes various deficiencies of the Implicit Shape Model (ISM) [13]. In particular, PRISM allows for discriminative voting weights. On the other hand, the hypothesis score is expressed in a feature-centric form inspired by the Hough transform. This nicely combines with the ideas from Efficient Subwindow Search (ESS) [11] and leads to advantages at detection time. Our approach thus combines elements of what is currently the state-of-the-art in both the sliding-window and the Hough paradigm, as it is indebted to both ESS [11] and PRISM [12].

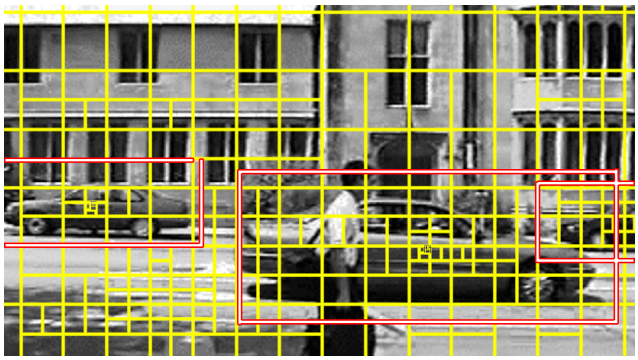

Figure 1. *Adaptive subdivision of the search space. The object center serves as hypothesis parametrisation. Each yellow box corresponds to a set of hypotheses (which comprises objects whose center is within the box). The discretisation around object centers is much finer than on background regions.*

ESS, which was recently introduced in [11], is an elegant solution to overcome exhaustive search of sliding-window systems. The key element is the use of bounds on sets of hypotheses. Embedded in a branch and bound search scheme [9], this allows for object detection in sub-linear time. However, ESS needs to compute integral images on-line prior to the actual search. These are very memory demanding and come at a computational cost during recognition. There are two integral images per spatial histogram bin (of the object model), which scale with the input image size. Hence, a single-class detector with 10×10 histogram bins (as presented in [11] p.6, but without spatial pyramid) already consumes on the order of 235MB memory for mod-

erately sized 640×480 images[1]. Depending on the application, the problem may be less the memory usage, but more the fact that memory has to be filled with data in an on-line pre-processing step, prior to the search. 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 overcomes the memory problem and avoids any on-line precomputation. Both memory usage and runtime are sublinear 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.

With this paper, we do not intend to propose a new object modeling scheme. Hence, it is not about improving object detection rates. Rather the contribution is the insight that the seminal 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. The structure of the paper is as follows: Sec. 2 revisits the PRISM framework. Integration of the ESS principle is shown in Sec. 3, along with theoretical comparison to ESS's bound. Sec. 4 gives implementation details followed by the evaluation in Sec. 5.

2. The PRincipled Implicit Shape Model

The task of object detection can be formulated as a search problem. Given a newly observed image I and a trained object model W, the goal is to find the best hypothesis $\lambda^* = \arg\max_{\lambda \in \Lambda} S(\lambda | I, W)$, where S is a score function and Λ is the search space of all potential object hypotheses. Note that the search space is large and finding the best-scoring hypothesis quickly is of great importance. The structure of S plays an important role when it comes to defining efficient search algorithms. We believe that the *feature-centric* definition of a score (as introduced in ISM [13]) is a powerful approach. However, ISM's probabilistic model has various shortcomings[2] and also does not allow for discriminative voting weights. The PRincipled Implicit Shape Model (PRISM) [12] overcomes these problems by starting from a sliding-window based reasoning. The remainder of this section defines the feature-centric score. The derivations are graphically supported by Fig. 2.

Score Function. The key element of PRISM is a footprint $\phi(\lambda, I)$ for a given object hypothesis λ extracted from the image I. This footprint represents all detected features

[1] 2 integral images · 640 × 480 pixel · 100 bins · 4 bytes ≈ 235MB. Using the 10 level pyramid increases memory usage to about 900MB.

[2] The most problematic point is to explain the summation over features by marginalisation: it implies that all features are *possible* realisations of a random variable, but *only one* can actually happen. See [12] for details.

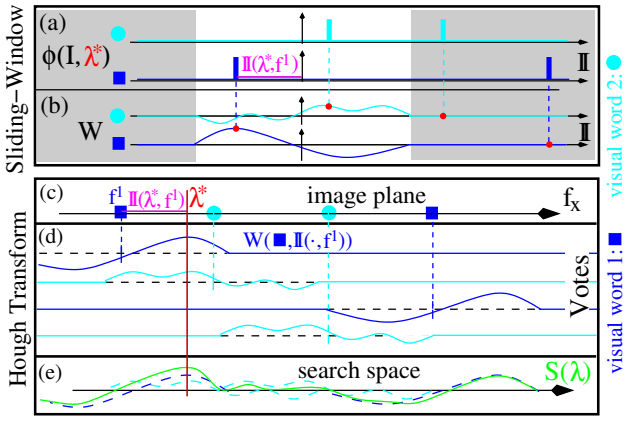

Figure 2. *PRISM: From the Sliding-Window (top) to the Hough Transform paradigm (bottom). Shift-invariant 1D-example with four extracted features (c) quantised into two visual words (blue rectangle and cyan circle). Sliding-window view (top): A* single *hypothesis λ^* is fixed (red). Its footprint (a) is a sum of dirac pulses which are positioned according to the invariant $\mathbb{I}(\lambda^*, f) = f - \lambda$ (depicted in magenta for f^1). Intuitively, 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). Various features do* not *affect the score (shaded). 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, whose intersection with the red line corresponds to the red dots.*

in a canonical reference frame. This leads to a translation- and scale-invariant characterisation of the object. A linear model is then used to rank this footprint. Thus, the hypothesis score is computed by the inner product

$$S(\lambda | I, W) = \langle \phi(\lambda, I), W \rangle \tag{1}$$

with a weight function W, *i.e.*, the object model. Classical sliding-window detectors [5, 6, 11] represent all features in an object-centric coordinate frame. Often, the relative position is discretised, which leads to histogram representations for ϕ and W. PRISM focuses on the definition of the footprint ϕ and avoids the discretisation. This allows to change from the sliding-window to a feature-driven Hough viewpoint. This switch is crucial for our search algorithm.

Image-Object invariants. An important aspect of the footprint is to relate 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 extract a set of local features $\mathcal{F}$ [15]. Each feature is characterised by a descriptor, position, and scale, *i.e.*, $f = (f_c, f_x, f_y, f_s)$. Additionally, a feature weight $f_\omega > 0$ may be defined. In the remainder, we use SURF [3] features and f_c is an index to the best-matching visual word in a learned vocabulary. Moreover, in this work we set $f_\omega = 1$. As object parametrisation we use $\lambda = (\lambda_x, \lambda_y, \lambda_s)$, *i.e.*, the object's position and size,

respectively. Possible mappings of the observables (λ, f) into a translation- and scale-invariant space are

$$\mathbb{I}_f = [\frac{\lambda_x - f_x}{f_s}, \log \frac{\lambda_s}{f_s}], \quad \mathbb{I}_\lambda = [\frac{f_x - \lambda_x}{\lambda_s}, \log \frac{f_s}{\lambda_s}], \quad (2)$$

where the y-coordinate is analogous to x and dropped for brevity's sake. The former invariant considers a feature-centric coordinate frame, while the latter opts for an object-centric one. Their differences are discussed in [12]. In this work, we stick to the classical sliding-window invariant $\mathbb{I}_\lambda$.

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

$$\phi(\lambda, I) = \sum_{f \in \mathcal{F}} f_\omega \delta_{f_c, \mathbb{I}(\lambda, f)}. \quad (3)$$

In classical sliding-window systems, this 4D function is discretised and represented with a histogram. PRISM avoids this discretisation. Instead, the footprint is plugged directly into the score (Eq. (1)). Thus, the inner product of the two functions yields a score S for hypothesis λ as

$$S(\lambda) = \langle \phi(\lambda, I), W \rangle = \sum_{f \in \mathcal{F}} f_\omega W(f_c, \mathbb{I}(\lambda, f)), \quad (4)$$

i.e., as point evaluations of the weight function. This form makes the connection to the feature-driven Hough-transform explicit. The summand $f_\omega W(f_c, \mathbb{I}(\cdot, f))$ represents the "vote" cast by each feature. Although no longer really visible, this formulation is *equivalent* to a standard sliding-window classifier (*e.g.*, [5]) when considering the object-centric invariant $\mathbb{I}_\lambda$ and a histogram for W. One is however not restricted to this choice.

Non-Contributing Features. Objects have a finite extent. Thus, the weight function W has a compact support and is zero outside that range. As a consequence, many (far away) features do not contribute to the score, *i.e.*, $W(f_c, \mathbb{I}(\lambda, f)) = 0$. In other words, the "windowing" in sliding-window approaches is a consequence of the model 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 Sec. 4.

3. Efficient Search

Ignoring large parts of the search space with only little effort is a crucial factor for efficient search strategies. Cascades of (increasingly complex) classifiers [18] 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 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).

Recently, branch and bound [9, 11] showed to be 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 hypotheses and to ignore low-scoring sets. We adopt this idea and show the benefits of a feature-centric view of the score (Eq. (4)).
The branch and bound algorithm will be revisited in Sec. 4, while we now focus on the definition of a feature-centric bound. The latter is a key contribution of this paper. Sec. 3.1 presents a rather generic class of bounds, which makes no assumption about the weight function. From that we derive a concrete bound in Sec. 3.2 which assumes a histogram representation. We would like to emphasise that our bound is complementary to ESS's bound [11] as it makes different trade-offs. A detailed comparison is given in Sec. 3.3.

3.1. Score Function on Hypothesis Sets

Let us extend the definition of the score function (Eq. (4)) to sets of hypotheses $\Omega \subset \Lambda$. Jensen's inequality yields the generic upper bound

$$S(\Omega) \stackrel{def}{:=} \max_{\lambda \in \Omega} S(\lambda) \leq \sum_{f \in \mathcal{F}} f_\omega \max_{\lambda \in \Omega} W(f_c, \mathbb{I}(\lambda, f)), \quad (5)$$

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. 3). Given a feature f, a set Ω (in the hypothesis space) maps to a region $\bar{\Omega}_f$ (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*, where W acts as "database" and $\bar{\Omega}$ is the query. The goal is to design a method to answer 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) can be computed *off-line* during learning because the database W is independent of the test data. The next subsection shows a concrete implementation of such maximum queries.

3.2. Maximum Query using Integral Images

We now present *a* possible method to efficiently process the maximum queries of Eq. (5). In this concrete setup, we discretise the weight function W and represent it by a histogram. This is a common setup [5, 6, 11] and learning can

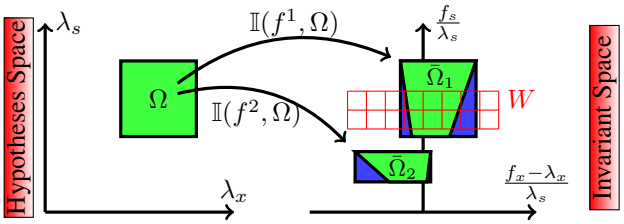

Figure 3. *Given (two example) features f^i, a hypothesis* set Ω *is mapped to the invariant space by means of the invariant $\mathbb{I}$. Both the original set Ω and the mapped one $\bar{\Omega}$ are coloured in green. The weight histogram W (only one slice corresponding to the matching visual word) 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) allows for fast evaluation of a bound on this maximum using integral images.*

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. 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 consider an upper bound for the maximum using mean μ and variance σ^2, *i.e.*,

$$\begin{aligned} \max W|_{\bar{\Omega}} &= \mu(W|_{\bar{\Omega}}) + (\max(W|_{\bar{\Omega}}) - \mu(W|_{\bar{\Omega}})) \\ &\leq \mu(W|_{\bar{\Omega}}) + g(|\bar{\Omega}|)\sigma(W|_{\bar{\Omega}})) \end{aligned} \tag{6}$$

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 $g(n) = \sqrt{n-1}$. 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 (Sec. 5). As the region $\bar{\Omega}$ has a complicated boundary (green polygon in Fig. 3), we slightly loosen the bound by expanding the region to the smallest enclosing rectangle. This allows to efficiently compute μ, σ^2 by means of integral images. Note that these integral images only depend on the object model and can be pre-computed *during learning*. Although we also rely on integral images, our approach differs from ESS in various ways. Therefore, we will briefly revisit ESS's bound and discuss the relation to our framework thereafter. For a detailed description of ESS, we refer to the original paper [11].

3.3. 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, they are represented by histograms. The following description is accompanied by the visualisation in Fig. 4. Let us denote a bin of a histogram by b and, given a hypothesis λ, its back-projection onto the image

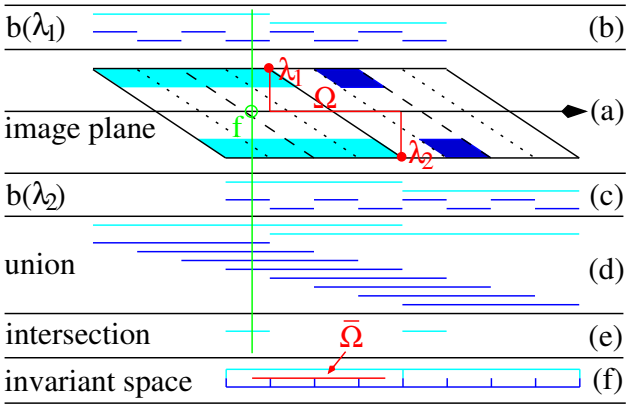

Figure 4. *1D illustration of ESS's bound computation. The setup shows a shift-invariant detector with a fine (dark blue) and a coarse histogram (light blue). For visibility, we show just one single feature f (green) and consider the set of hypotheses $\Omega = [\lambda_1 \dots \lambda_2]$ (red). (a): 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^-).*

grid by $b(\lambda)$. ESS rewrites the hypothesis score (Eq. (4)) as

$$S(\lambda) = \sum_b Q_b^+(b(\lambda)) - \sum_b Q_b^-(b(\lambda)) \tag{7}$$

where $Q_b^\pm(q) = \sum_f [f_w W^\pm(f_c, b)] \mathbb{1}_{f \in q}$ and $\mathbb{1}$ is an indicator which is 1 if $f \in q$ and 0 otherwise. The expressions $Q_b^\pm$ are (weighted) range sum queries. They can be efficiently evaluated using integral images if the query region $q := b(\lambda)$ is rectangular. The upper bound of ESS is obtained by considering upper bounds on Q_b^+ and lower bounds on Q_b^-. The former 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 $q_b^+ := \cup_{\lambda \in \Omega} b(\lambda)$. In the latter case the feature has to fall into $b(\lambda)$ for *all* $\lambda \in \Omega$. Thus, the query for Q_b^- is just the intersection $q_b^- := \cap_{\lambda \in \Omega} b(\lambda)$, which may be empty.

Comparison. We now compare our bound (*i.e.*, Eqs. (5, 6)) to the one of ESS. Contrary to ESS, our integral images do *not* depend on the size of the test image, but only on the model 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. 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 Sec. 5, where we show that feature-centric ESS can get along with $25\times$ 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. 4): $f \in q_b^+$ 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. 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

$$\max_{\lambda \in \Omega} W = \begin{cases} n \cdot \mu(\max(W|_{\bar{\Omega}}, 0)) & n > 1 \\ W(f_c, B) & n = 1 \end{cases} \quad (8)$$

(instead of Eq. (6)) 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, 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 Sec. 5.

4. Implementation

We now give implementation details of our algorithm, which is to be seen mainly as an illustration of the feature-centric ESS idea, the key contribution of this paper. As mentioned earlier, our system is similar to ESS [11] as we use a histogram representation and the object-centric invariant $\mathbb{I}_\lambda$. We also employ branch and bound to search for objects, but use the feature-centric bound Eqs. (5, 6) instead of ESS's bin-centric one. Consequently, our approach avoids any online pre-processing, which is a clear advantage. In the sequel, we discuss the following three aspects. Firstly, we briefly revisit the branch and bound algorithm. 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. 1). Eventually, the size of a hypothesis set becomes very small, *i.e.*, we converge to a single object which is reported as detection.

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. 3). The sub-divisive nature of our 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).

Multiple Objects. Sliding window based reasoning treats all hypotheses independently. Thus, there is no limit on the number of detectable objects. However, it does not account for already explained evidence and tends to re-detect the same object. Limiting each feature to explain only one single object overcomes this problem. In this work, we consider a greedy strategy: once the best object is found, we eliminate all features which explain this object, and restart the search for more objects. This procedure suppresses the score of nearby hypotheses and relates to usual non-maximum suppression post-processing. Consequently, 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 [11], as all integral images would have to be recomputed from scratch.

5. Experiments & Discussion

We evaluated our algorithm on two benchmark datasets. As in [11], we use the UIUC cars database [1] 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 [8] which includes 153 training images and 115 test im-

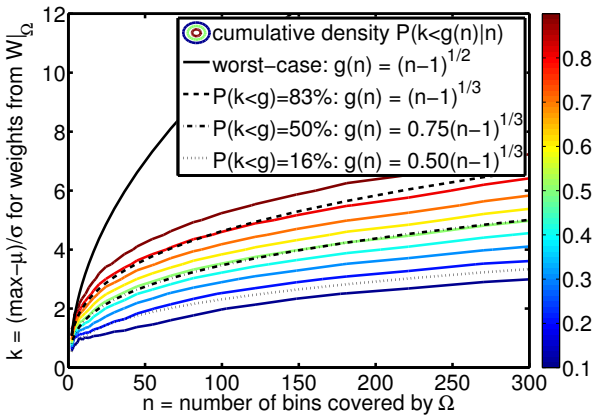

Figure 5. *Confidence* $P(\frac{\max-\mu}{\sigma} < g(n)|n)$ *in the bound Eq. (6) given* $g(n)$*. For fixed* n*, we can estimate this cumulative density by examining all possible rectangular sub-regions of* W *consisting of* n *bins. The iso-probability lines (as function of* n*) are colour coded with 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*.*

ages. In order to compare our results to the published baseline performance, we use the same evaluation criterion as in [1, 8]. 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.

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 [11] 1.4% and ISM [13] 5%). On TUD motorbikes we achieve an equal error rate of 19%, *i.e*. close to the 18% reported by the original authors [8]. Although we demonstrate state-of-the-art performance on both datasets, we would like to emphasise that this paper is about efficient search, and *not* about learning robust models, *e.g*. [4]. In the sequel, we focus on the analysis of runtime properties and on the comparison to ESS [11].

Approximate Correction Term. As mentioned in Sec. 3.2, the worst case correction term $g(n) = \sqrt{n-1}$ is very pessimistic. 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. Fig. 5 shows the estimation for our UIUC cars model. For all following experiments, we consider $\tilde{g}(n) := \alpha\sqrt[3]{n-1}$ as it approximates the iso-probability lines very well, *i.e*., the confidence in this "probabilistic bound" is (almost) independent of n. Next, we study the effect of α.

Runtime Comparison. We now 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

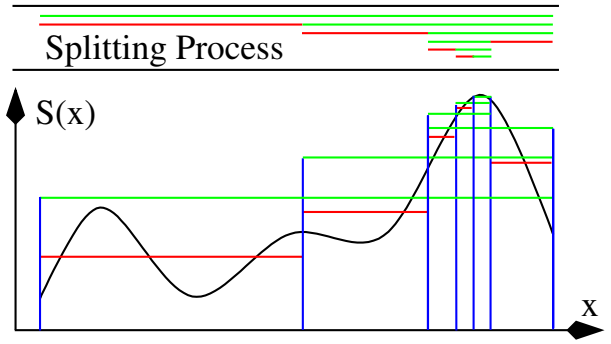

Figure 6. *Branch and bound search using the mean (*i.e.*,* $g(n) = 0$*) instead of an upper bound. The splitting process is shown on top: green lines represent hypotheses sets which are refined, while red lines have low priority and are not split any further. The actual priority (*i.e.*, mean value) 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.*

baseline. As baseline we consider a system which has access to the true maximum of Eq. (6). The results for both datasets are reported in Fig. 7. As the behaviour is similar for both datasets, we limit our discussion to UIUC cars (Fig. 7(left)). For comparison, we emulate the bound of ESS [11] in our framework. ESS performs as well as the baseline, but needs about $1.7\times$ 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[3] detects objects in about $0.8s$. 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 Eq. (5) in our reasoning. This compensates for a slight underestimation of the max term Eq. (6). Secondly, branch and bound does not break down completely if we use a *priority function* which is *not* an upper bound. Fig. 6 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. Fig. 8 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

[3] available at `www.vision.ee.ethz.ch/lehmanal/iccv09`

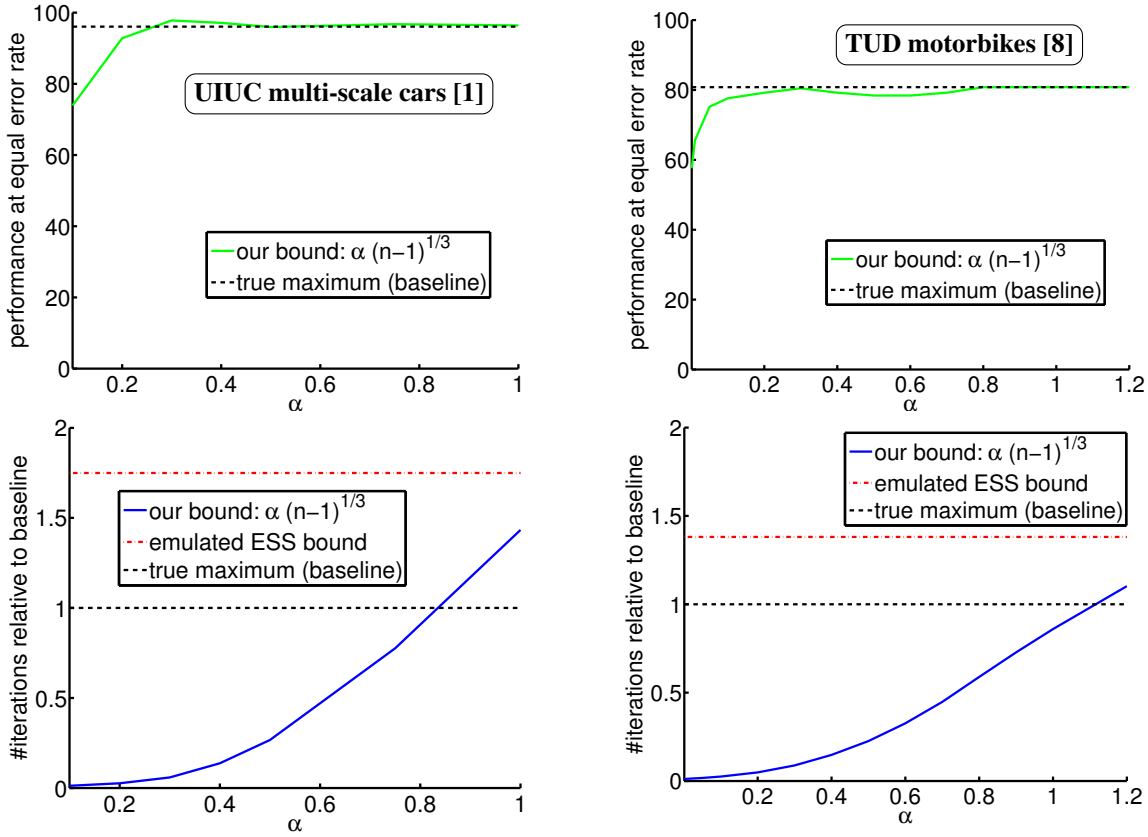

Figure 7. *Evaluation of the probabilistic bounds in terms of performance at equal error rate (top), and number of iterations relative to the baseline (bottom). As baseline we consider a system which has access to the true maximum in Eq. (6). The results are shown for UIUC multi-scale cars (left) and 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.*

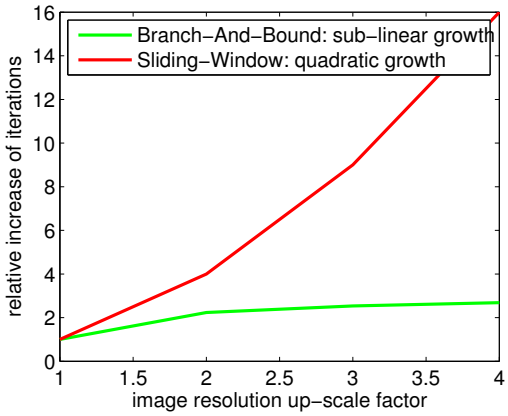

Figure 8. *Runtime/memory dependence on image resolution: Our adaptive search (green) versus quadratic growth of classical sliding-window search (red).*

quadratic growth inherent in an exhaustive, sliding-window based search. 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 $|\Lambda|$ 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. The number of features is affected by the visual content (*i.e.*, more complicated scenes require more work), while sublinearity can be interpreted as follows. There is a logarithmic-time lower bound for general search algorithms [10]. However, this bound assumes the data to be sorted, which is not (fully) true for image data. Thus, sub-linear time seems to be very reasonable. Interestingly, neither the model size (*i.e.*, number of bins) nor the image resolution affect runtime directly.

Memory Comparison. Both ESS and our approach use integral images. However, there is a crucial difference due to the feature-/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) on-line

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. 8, 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 $860^2 \times 10^2 \times 2 \times 4B \approx 564$MB. On the other hand, our system stores 2×4 Bytes per contributing feature and it requires about $1.8 \cdot 10^6$ integral image evaluations. Thus we use roughly 20MB, *i.e*. 25 times less memory.

An implication of the memory bottleneck is that ESS uses only $2D$ integral images for their range queries. That implies that the feature scale f_s is completely ignored[4]. This is a limitation, as assigning different weights to (relatively) large-/small-scale features is desirable. Moreover, it may lead to 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. In the experiments we use a spatial histogram with $10 \times 10 \times 3$ bins. Therefore, there is no bias towards larger detection windows.

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 with $O(sl(|\Lambda|)\bar{F})$. Hence, the number of classes affects runtime/memory only through the size of the search space, where we expect sub-linear scaling. In contrast, runtime and memory usage of ESS scale as $O(|I|BC + sl(|\Lambda|)B)$, where B denotes the number of spatial bins and C denotes the number of classes (and views). Thus, the pre-processing introduces a linear dependence which cancels the sub-linear behaviour of the actual search and emphasises the memory bottleneck. 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/view setups.

6. Conclusion

In this work, we successfully applied the ideas of Efficient Subwindow Search (ESS) to the PRincipled Implicit Shape Model (PRISM). We thus combined elements of what is currently the state-of-the-art in sliding-window and Hough based object detection. Branch and bound in combination with a feature-centric score (instead of ESS's window-centric one) seems to be very promising. Most importantly, our approach avoids any online preprocessing which may cancel the sub-linear runtime of the actual search. Moreover, our system 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/views.

[4]Using only $2D$ integral images corresponds to using $\mathbb{I}_\lambda$ and a histogram which extends infinitely in the scale-ratio dimension ($\mathbb{I}_{\lambda,s}$).

Acknowledgments

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

References

[1] S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. *PAMI*, 26(11):1475–1490, 2004.

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

[3] H. Bay, A. Ess, T. Tuytelaars, and L. van Gool. Surf: Speeded up robust features. *CVIU*, 110(3):346–359, 2008.

[4] M. B. Blaschko and C. H. Lampert. Learning to localize objects with structured output regression. In *ECCV*, 2008.

[5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[6] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *CVPR*, 2008.

[7] V. Ferrari, F. Jurie, and C. Schmid. Accurate object detection with deformable shape models learnt from images. In *CVPR*, 2007.

[8] M. Fritz, B. Leibe, B. Caputo, and B. Schiele. Integrating representative and discriminant models for object category detection. In *ICCV*, 2005.

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

[10] D. E. Knuth. *Art of Computer Programming, Volume 3: Sorting and Searching*. Addison-Wesley Professional, 1998.

[11] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *CVPR*, 2008.

[12] A. Lehmann, B. Leibe, and L. van Gool. Prism: Principled implicit shape model. In *BMVC*, 2009.

[13] B. Leibe, A. Leonardis, and B. Schiele. Robust object detection by interleaving categorization and segmentation. *IJCV*, 77(1-3):259–289, 2008.

[14] J. Liebelt, C. Schmid, and K. Schertler. Viewpoint-independent object class detection using 3d feature maps. In *CVPR*, 2008.

[15] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. *IJCV*, 60(1):63–86, 2004.

[16] A. Opelt, A. Pinz, and A. Zisserman. A boundary-fragment-model for object detection. In *ECCV*, 2006.

[17] H. Schneiderman and T. Kanade. Object detection using the statistics of parts. *IJCV*, 56(3):151–177, 2004.

[18] P. A. Viola and M. J. Jones. Robust real-time face detection. *IJCV*, 57(2):137–154, 2004.