Shufflets: shared mid-level parts for fast object detection

Iasonas Kokkinos
Ecole Centrale de Paris INRIA
Center for Visual Computing Galen Team, INRIA-Saclay
iasonas.kokkinos@ecp.fr *

Abstract

We present a method to identify and exploit structures that are shared across different object categories, by using sparse coding to learn a shared basis for the 'part' and 'root' templates of Deformable Part Models (DPMs).

Our first contribution consists in using Shift-Invariant Sparse Coding (SISC) to learn mid-level elements that can translate during coding. This results in systematically better approximations than those attained using standard sparse coding. To emphasize that the learned mid-level structures are shiftable we call them shufflets.

Our second contribution consists in using the resulting score to construct probabilistic upper bounds *to the exact template scores, instead of taking them 'at face value' as is common in current works. We integrate shufflets in Dual-Tree Branch-and-Bound and cascade-DPMs and demonstrate that we can achieve a substantial acceleration, with practically no loss in performance.*

1. Introduction

Deformable Part Models (DPMs) [9, 8] have been established as a robust framework to tackle a broad range of problems in object recognition, but a main impediment to their broader application is their computation time. In this work we use the models of [8] and focus on accelerating multiple-category detection by sharing computation.

Part score computation is the first, and most time-consuming step in the pipeline of object detection with DPMs; at this stage Histogram-of-Gradient (HOG) features [3] are convolved with part filters to provide local part scores, which are then combined to deliver object scores. This second stage of part combination can be efficiently implemented using low-constant linear-time algorithms such as Generalized Distance Transforms (GDTs) [9] or faster Branch-and-Bound alternatives, such as Dual-Tree Branch-and-Bound (DTBB) [16]; the real bottleneck is the front-end convolution with the part filters.

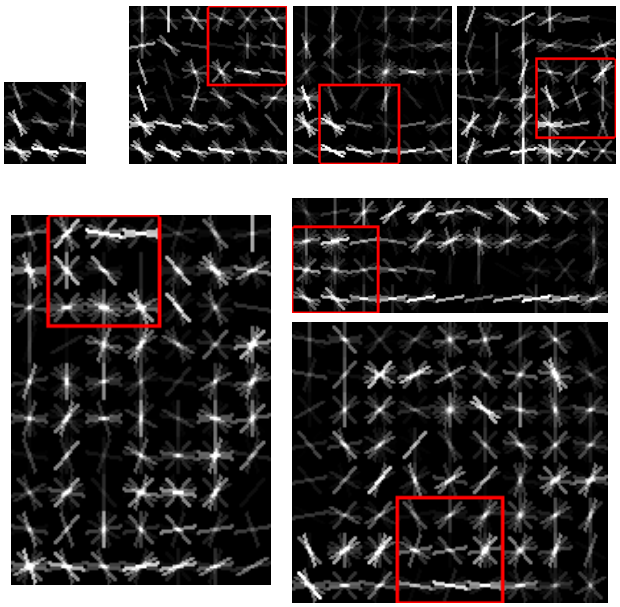


Figure 1: We introduce shufflets as a mid-level basis for object representation; what distinguishes shufflets is that the basis elements are 'shiftable', namely a single shufflet, like the corner in the top-left, can be used at many locations to reconstruct both part (top) and root (bottom) filters.

In this work exploit the redundancy that exists among the part filters of multiple categories to reduce the cost of computing part scores. For this, we learn a common, 'shared', basis to reconstruct the part and root filters; this basis serves as a mid-level interface between parts and HOG features.

This approach was recently advocated in [29, 12], while [17, 33, 27] have pursued similar ideas, either by replacing the 32-dimensional inner products of HOG cells with lookup-based approximations [17, 33] or by building a common basis for multiple part filters [27]. Along a complementary path, [6] developed a highly-optimized frequency domain acceleration technique for part score computation.

*This work was supported by grant ANR-10-JCJC-0205 (HiCoRe) and the EU-FP7 project RECONFIG.

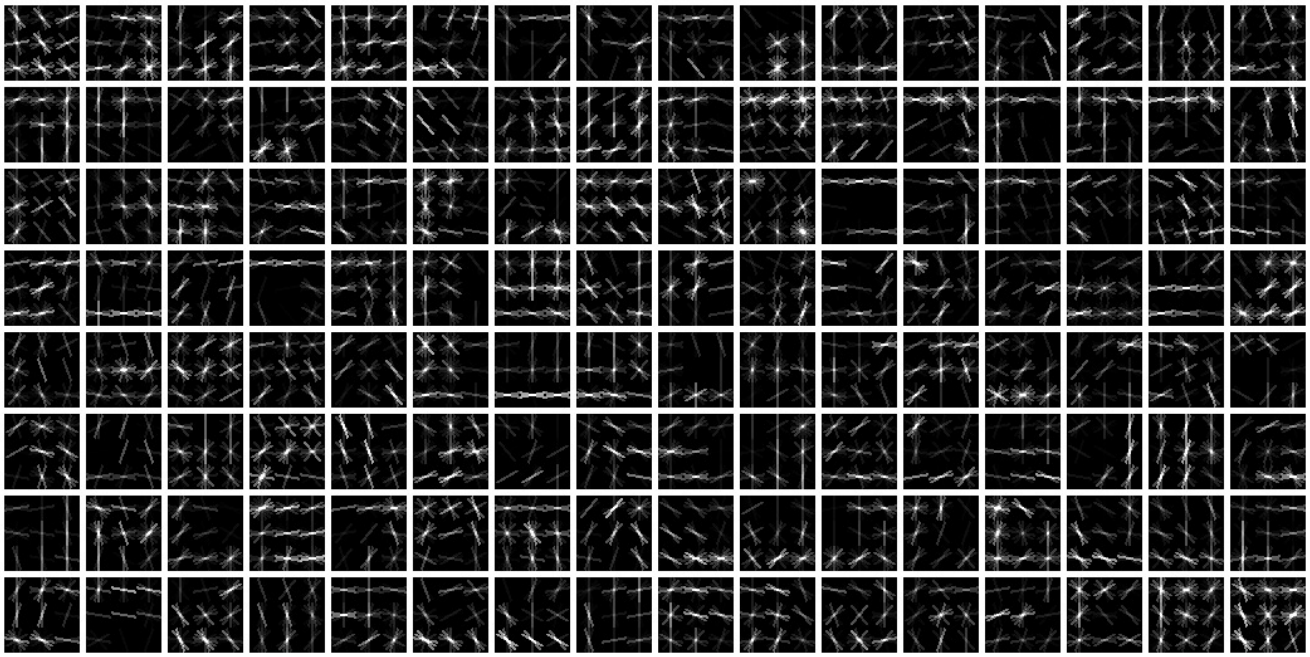


Figure 2: A dictionary of 128 3×3 shufflets learned with Shift-Invariant Sparse Coding (SISC).

More recently [4] used hashing to efficiently retrieve the approximate top-K scoring templates at every candidate part position. Our work presents advances with respect to these works in the following aspects:

First, regarding *quality of approximation*, compared to [29] we obtain a better approximation for the same number of terms, by allowing the mid-level parts to translate.

Second, regarding *detection accuracy*, unlike [27, 33, 29] who take the midlevel-based approximation 'at face value', we only use it as a rough initial estimate of the part scores, which is then refined around a subset of locations shortlisted based on the approximation. For this we use the probabilistic bounding technique of [17], and thereby attain virtually identical performance to the DPMs of [8].

Third, unlike [6] who compute the part scores densely we have the option of computing part scores 'on demand', i.e. at given locations, which is useful for cascaded detection.

Fourth, by virtue of being 'shiftable' our midlevel parts can be used as a *c*ommon basis to reconstruct both the part and the root filters. This is not straightforward for the techniques of [29, 27], which require predefining the filter size.

Finally, even though our method is linear in the number of categories -unlike the constant complexity of [4]- it has the advantage of coming with controllable accuracy, and we demonstrate that it yields virtually identical results as [8].

In particular we combine the proposed shufflets with the Dual-Tree Branch-and-Bound technique [16, 17] which was originally developed to accelerate the stages following part computation. We use shufflets to construct probabilistic upper bounds for the part scores, and use these bounds to drive branch-and-bound to a small set of candidate object locations. Our algorithm eventually computes the exact values of the part scores and the correct object score, but only around the locations shortlisted by the preliminary bounding stage. This spares us from computing the exact part scores at locations that do not merit it.

We quantitatively evaluate our method and demonstrate that it allows for a multi-fold speedup over the methods of [8, 11], while yielding virtually identical results. Our implementation will be available from [1].

2. Previous Work

The idea of using shared parts to detect multiple categories has its roots in the earliest days of recognition [7, 14, 37, 5], but its application to detection with statistical object models started later. A ground-breaking work was [32] who proposed the sharing of weak learners for multi-view and multi-class object detection, using region-based features as inputs to decision stumps. In [25] this idea was applied to the implicit shape models ISM [21] to learn a library of shared contour parts, while the ISM was also extended to multi-view [30] and multi-class [23] recognition by employing a shared dictionary across different views/categories; more recent work in [28] revisited the dictionary construction stage to ensure a more distinctive per-class distribution of votes for multi-class detection. However, since most of these works rely on voting, where correspondence information is lost, the performance of the raw, voting-based detector is substantially lower than that

of DPMs, as demonstrated in [28].

Data-driven approaches to discovering shared structures in object categories include [10], who used agglomerative clustering to recover hierarchical, shape-based representations, [31] where co-segmentation was used to extract shared structures among similar categories, the test hierarchy work of [36] as well as a host of works around convolutional learning of deep networks [35, 20, 15, 19]. These works have consistently demonstrated that the learned shared parts correspond to generic grouping rules at the lowest levels and more semantic configurations at the higher parts of the hierarchy.

Part sharing for DPMs has recently been explored from the multi-task/transfer learning perspective, examining the improvements in classification attained by transferring information across categories; in particular part sharing in [26] and root sharing in [2] were demonstrated to facilitate the learning of category models from smaller number of training images. The sparselets work of [29] introduced a sparse coding-based approach to express multiple part filters on a common basis and thereby accelerate detection with DPMs; in [12] this was shown to improve accuracy in large-scale category classification, if properly integrated during training. In [27] the *steerability* of the part filters was enforced during training, and was shown to result in both better training and faster testing. Even though shufflets could potentially be integrated in training, we have not explored this direction yet. In that respect when it comes to part sharing for DPMS, the most relevant works are those covered in the introduction.

3. Shufflets: shiftable mid-level parts

We start by describing the operation that we want to accelerate through our mid-level basis. The score of a filter p at position y is expressed as the inner product of a weight vector $\mathbf{w}_p$ and a HOG feature $\mathbf{h}_y$, $s_p(y) = \langle \mathbf{w}_p, \mathbf{h}_y \rangle$, with $\mathbf{w}_p \in R^D$. For a part filter of size $v \cdot h = (6 \cdot 6)$ we have $D = v \cdot h \cdot d = (6 \cdot 6 \cdot 32)$ where $d = 32$ is the size of each HOG cell. So every part requires roughly a thousand multiplications per candidate location.

Our goal is to recover a basis $\mathbf{b}$ of B vectors to approximate a large set of weight vectors $\mathbf{w}_p, p = 1 \dots P$. Once this basis is available, we can perform the high-dimensional inner products once with the basis elements and then linearly combine their scores to approximate all of the the part scores, by linearity:

$$\mathbf{w}_p \simeq \sum_{b \in I_p} \alpha_p(b) \mathbf{b}_b \tag{1}$$

$$\langle \mathbf{w}_p, \mathbf{h}_y \rangle \simeq \sum_{b \in I_p} \alpha_p(b) \langle \mathbf{b}_b, \mathbf{h}_y \rangle \tag{2}$$

where α_p is the expansion vector for the p-th part and I_p indicates the non-zero elements of the vector for part p.

When working with multiple categories/viewpoints the cost of computing the convolution with the midlevel parts will eventually become amortized. This approach can thus be understood as cutting the computation cost down from a thousand operations per location to the number of nonzero-elements in α_p, $L = \|\alpha_p\|_0$.

Coming now to finding a good basis, $\mathbf{b}$, we consider first the basis that minimizes the ℓ_2 norm of the distortion, while using expansion vectors of bounded ℓ_0 norm. This results in the following sparse coding problem:

$$(P1) \qquad \min_{\mathbf{b}, \alpha_1 \dots \alpha_P} \sum_{p=1}^{P} \|\mathbf{w}_p - \mathbf{b}\alpha_p\|_2 \tag{3}$$

$$\text{s.t.} \qquad \|\alpha_p\|_0 \leq L, \quad p = 1, \dots P \tag{4}$$

$$\|\mathbf{b}_b\|_2 \leq 1, \quad b = 1, \dots B \tag{5}$$

where $\mathbf{b}_b$ is the b-th row of the basis. The criterion in Eq. 3 penalizes the distortion of the basis elements, the constraint in Eq. 4 enforces the sparsity of the expansion vector and the constraint in Eq. 5 ensures that the basis elements will have unit ℓ_2 norm. We can find (local) minima of this non-convex optimization problem with alternating optimization: for given expansion coefficients the optimal basis elements can be recovered through matrix inversion, and for given basis elements an approximate optimizer of α_p can be obtained by Orthogonal Matching Pursuit (OMP) (we use the implementation of [22]). This standard sparse coding formulation was used for sparselets [29].

We propose instead to use Shift-Invariant Sparse coding, which intuitively amounts to having 'shiftable' basis elements. This exploits the fact that the weight vectors are not arbitrary high-dimensional vectors, but rather have an underlying spatial structure.

Namely, instead of using a basis with arbitrary vectors of dimension $v \cdot h \cdot d$, we first consider a kernel $\mathbf{k}$ for our basis elements on a smaller domain of size $v' \cdot h' \cdot d$. We then consider every displacement of this kernel that allows the displaced kernel to be fully contained in the original basis domain. This provides us with $(v - v' + 1) \cdot (h - h' + 1)$ translated replicas of $\mathbf{k}$, denoted as $\mathbf{k}^{\nu,\eta}$. Using these in Eq. 1 and Eq. 2 we have:

$$\mathbf{w}_p \simeq \sum_{b \in I_p} \alpha_p(b) \mathbf{k}_b^{\nu_b, \eta_b} \tag{6}$$

$$\langle \mathbf{w}_p, \mathbf{h}_y \rangle \simeq \sum_{b \in I_p} \alpha_p(b) \langle \mathbf{k}_b^{0,0}, \mathbf{h}_{y+(\nu_b, \eta_b)} \rangle, \tag{7}$$

where $\mathbf{k}_b, \nu_b, \eta_b$ denote the kernel and displacements corresponding to the b-th basis element while $\mathbf{h}_{y+(\nu_b,\eta_b)}$ corresponds to HOG features extracted at a position displaced by (ν_b, η_b); we have used $\langle \mathbf{k}_b^{\nu_b,\eta_b}, \mathbf{h}_y \rangle = \langle \mathbf{k}_b^{0,0}, \mathbf{h}_{y+(\nu_b,\eta_b)} \rangle$ to obtain the bottom expression.

This illustrates the merit of this scheme: during the approximation of $\mathbf{w}_p$ we are free to use any of the replicas $\mathbf{k}^{\nu,\eta}$ of $\mathbf{k}$; but at test time we only need to convolve with a single replica $\mathbf{k}^{0,0}$, and then use appropriately displaced convolution scores.

Turning to learning the kernels, $\mathbf{k}_k, k = 1,\dots,K$, we consider the following optimization problem:

$$(P2) \quad \min_{\mathbf{k},\alpha_1\dots\alpha_P} \sum_{p=1}^{P} \|\mathbf{w}_p - \mathbf{b}\alpha_p\|_2 \tag{8}$$

$$\text{s.t.} \quad \|\alpha_p\|_0 \leq L, \quad p = 1,\dots,P \tag{9}$$

$$\|\mathbf{k}_i\|_2 \leq 1, \quad i = 1,\dots,K \tag{10}$$

$$\mathbf{b}_b = \mathbf{k}_b^{v_b,h_b}, \quad b = 1,\dots B \tag{11}$$

where the last constraint indicates that the replica $\mathbf{k}_b^{v_b,h_b}$ of kernel $\mathbf{k}_b$ is used to form the b-th basis element. As in P1, we face a non-convex optimization problem, so we need to resort to alternating optimization. For known basis elements, the expansion coefficients are obtained through OMP, as in P1. However the estimation of the basis elements is more challenging, since we no longer optimize over an arbitrary basis $\mathbf{b}$, but rather constrain the basis to be 'shiftable', through the set of constraints in Eq. 11.

This results in a problem where the different dimensions of the kernel $\mathbf{k}$ become coupled in the objective. Its direct solution is non-trivial, but an efficient technique for solving this optimization problem was introduced in [13] by casting it in the frequency domain - there the translations are turned into multiplications with complex exponentials, and the set of constraints in Eq. 11 becomes decoupled in the different frequencies; we refer to [13] for further details, since we rely entirely on their approach for basis learning.

Our main difference with the method of [13] is that we use the ℓ_0 instead of ℓ_1 sparsity penalty on the expansion vectors. This more aggressive sparsity cost reflects our setup, where we only care about the number of operations, rather than the magnitude of the coefficients.

We demonstrate in Table 1 the decrease in reconstruction error attained by our method, when compared to the simpler sparse coding technique employed in [29]. Actually we used the sparse coding results of [29] as initialization for our optimization procedure, which practically ensures a better approximation quality. For most categories we have substantial reductions, while paying the same computation budget at test time. Furthermore, the approximation of the root scores can be effortlessly achieved by our method; qualitative results are provided in Fig. 3.

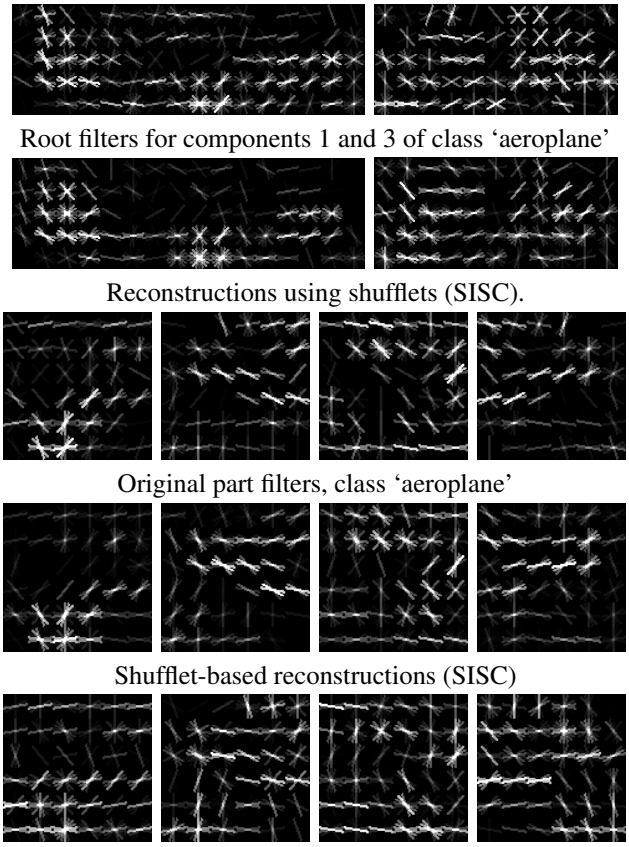


Figure 3: Shufflet-based reconstructions of filters for class 'aeroplane', using a dictionary with 128 elements, 16 coefficients for parts and 32 coefficients for root filters. Comparing to the sparselet-based reconstructions of [29], we obtain better localized and sharper reconstructions.

4. From approximate scores to score bounds

We now describe how shufflets can be used to provide not only an estimate of the part score, but also a probabilistic upper bound to it. This allows us to avoid a pitfall of most of the existing works on fast approximate part score computation, which take the approximated score 'at face value', which incurs performance degradation as faster -and cruder- approximations are used.

Instead, as in [17], we consider that we should be using the shufflet-based scores only as proxies to the exact part scores; these proxies serve to reduce the set of locations that are considered for a subsequent, more refined estimation of the exact part score. For this, we efficiently compute upper bounds to the part scores, and use them in bounding-based detection with DPMs. This guarantees accuracy, and allows us to get the best of both worlds, namely a fast 'bottom-up' detection of objects with quickly computable scores, which is then complemented by an exact, 'top-down' score refinement around promising locations [17, 18].

Turning to how this can be achieved, we construct bounds on the approximation error:

$$\boldsymbol{\epsilon} = s - \hat{s} = \langle \mathbf{w}, \mathbf{h} \rangle - \langle \hat{\mathbf{w}}, \mathbf{h} \rangle \tag{12}$$

between the exact score s obtained by convolving with the

Method (L/B)	SC (16/64)	SISC (16/64)	SC (32/64)	SISC (32/64)	SC (16/128)	SISC (16/128)	SC (32/128)	SISC (32/128)	SC (16/256)	SISC (16/256)	SC (32/256)	SISC (32/256)
aeroplane	32.97	31.31	24.48	25.98	31.88	28.92	23.57	19.39	29.32	26.05	21.53	19.58
bicycle	67.02	59.57	48.66	47.10	65.74	51.59	47.63	33.65	60.70	48.37	43.67	32.87
bird	19.28	17.47	15.55	15.41	18.88	16.72	14.84	12.23	17.62	15.63	13.43	13.03
boat	12.67	12.09	9.74	10.32	12.06	11.62	9.17	8.16	10.91	10.84	8.16	8.70
bottle	36.57	33.83	28.78	28.59	35.10	31.25	27.18	21.65	32.98	28.24	24.57	21.44
bus	25.82	24.75	19.58	19.99	25.06	23.00	18.52	15.24	23.98	20.98	17.56	15.70
car	208.87	141.75	155.65	112.86	199.92	123.87	146.17	77.07	171.89	118.32	127.80	84.07
cat	27.38	25.14	21.37	21.22	27.24	23.62	20.48	16.50	25.80	21.86	18.96	17.16
chair	49.91	45.97	39.80	39.31	49.06	42.61	38.47	29.81	48.00	38.09	35.74	29.44
cow	19.59	18.78	15.23	15.66	19.35	17.72	14.50	12.11	18.17	16.56	13.34	13.03
diningtable	18.62	17.67	14.08	14.54	18.10	16.67	13.62	11.46	17.65	15.66	12.78	12.13
dog	26.63	23.95	20.89	20.72	25.93	22.80	20.30	16.22	25.06	21.01	18.70	17.06
horse	46.79	45.78	35.75	36.13	47.61	41.04	35.10	26.95	44.59	36.47	32.53	26.20
motorbike	46.98	42.53	35.07	34.48	45.94	38.96	34.06	26.17	43.10	34.31	31.65	25.38
person	597.17	359.05	455.53	270.02	589.53	326.24	431.17	202.72	540.43	326.58	403.42	229.58
pottedplant	18.91	17.51	15.46	15.48	18.38	16.82	14.70	12.25	16.67	15.60	12.94	13.03
sheep	12.51	11.68	9.86	10.00	11.97	11.23	9.16	7.85	11.01	10.47	8.00	8.46
sofa	20.70	19.57	15.82	16.03	20.50	18.45	15.21	12.55	19.61	17.09	14.12	13.28
train	31.97	31.34	24.23	25.23	31.65	28.87	23.35	19.37	30.69	25.97	22.38	19.63
tvmonitor	32.99	30.54	25.32	25.67	32.03	28.48	24.36	19.54	30.69	25.40	22.56	19.95

Table 1: Total reconstruction errors (scaled by 100) of the part filters for twenty categories, as a function of dictionary size, B, and expansion length, L. We compare Sparse Coding [29] and our Shift-Invariant Sparse Coding-based method; we observe that our method systematically achieves lower reconstruction errors at the same level of model complexity.

filter $\mathbf{w}$ and its shufflet-based approximation, $\hat{s}$, obtained from the approximate weight vector $\hat{\mathbf{w}}$.

We could construct a deterministic bound for $\boldsymbol{\epsilon}$ by using Holder's inequality, but this is practically too loose to be useful. Instead, we treat $\boldsymbol{\epsilon}$ as a random variable, and construct *probabilistic bounds* that are valid with controllable accuracy. We use Chebyshev's inequality [24] which ensures that a zero-mean random variable X with second moment $V = E\{X^2\}$ satisfies:

$$P(|X| > \alpha) \leq \frac{V}{\alpha^2}. \tag{13}$$

This means that with probability larger than V/α^2, X will be contained in $[-\alpha, \alpha]$, or equivalently X will be contained in $[-\sqrt{V/p_e}, \sqrt{V/p_e}]$ with probability of error smaller than p_e. We can use this fact to bound $\boldsymbol{\epsilon}$ probabilistically: with a probability of error p_e we will have $\boldsymbol{\epsilon} \in [-\sqrt{m/p_e}, \sqrt{m/p_e}]$, where m is the estimated second moment of $\boldsymbol{\epsilon}$ - which we will describe below. Since $\boldsymbol{\epsilon} = s - \hat{s}$, with probability $1 - p_e$ we will have:

$$\hat{s} - \sqrt{\frac{m}{p_e}} \leq s \leq \hat{s} + \sqrt{\frac{m}{p_e}}, \tag{14}$$

which provides upper and lower bouns for s in terms of $\hat{s}$. What remains is to express the second moment of $\boldsymbol{\epsilon}$ in a manner that can be efficiently computed. For this, expanding Eq. 12 we get:

$$\boldsymbol{\epsilon} = \langle \mathbf{w} - \hat{\mathbf{w}}, \mathbf{h} \rangle = \sum_c \epsilon_c, \quad \text{where} \tag{15}$$

$$\epsilon_c = \sum_{d=1}^{D} (\mathbf{w}[c,d] - \hat{\mathbf{w}}[c,d]) \mathbf{h}[c,d] \tag{16}$$

with c ranging over the C HOG cells comprising a part filter, and d ranges over the 32 HOG cell dimensions. Namely, for every cell c we are taking the inner product between then part score approximation error $\mathbf{w}[c,\cdot] - \hat{\mathbf{w}}[c,\cdot]$ and the associated HOG cell, $\mathbf{h}[c,\cdot]$. We assume that the reconstruction error terms $\mathbf{w}[c,d] - \hat{\mathbf{w}}[c,d]$ are zero-mean, independent and identically distributed (iid) random variables, and use the ℓ_2 norm of the reconstruction error vector to form an empirical estimate of their second moment, i.e. $V_c = \frac{1}{D} \sum_{d=1}^{D} (\mathbf{w}[c,d] - \hat{\mathbf{w}}[c,d])^2$. We can then see ϵ_c in Eq. 16 as the weighted average of D iid variables, which means that its second moment will be:

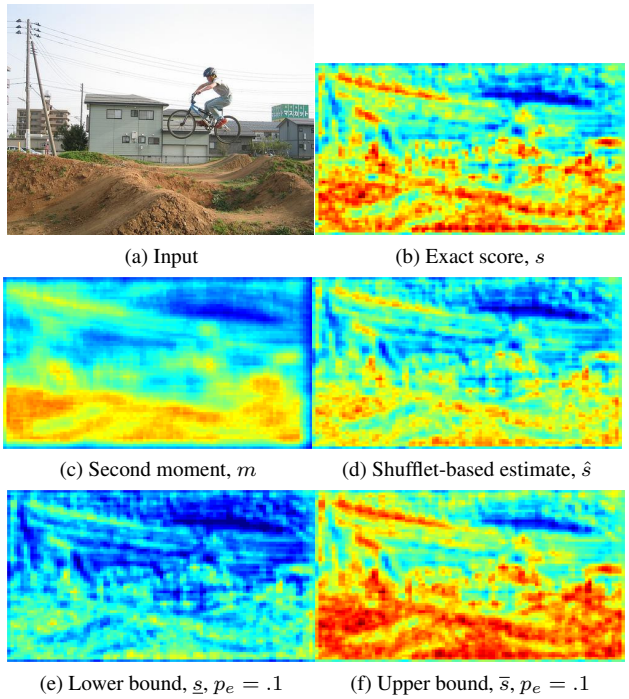
$$E\{\epsilon_c^2\} = V_c \|\mathbf{h}[c]\|_2^2. \tag{17}$$

where $\mathbf{h}[c]$ indicates the vector formed from the c-th HOG cell. What we gain in this way is that instead of computing a 32-dimensional inner product per HOG cell, we only need to multiply the two quantities on the right of Eq. 17; V_c can be precomputed, while the right term is computed once per HOG cell, and reused across all parts.

Pushing this on, we can consider that the individual error contributions in Eq. 15 are independent over c, which means that we can express the second moment of $\boldsymbol{\epsilon}$ as follows:

$$E\{\epsilon^2\} = \sum_c E\{\epsilon_c^2\} = \sum_c V_c \|\mathbf{h}[c]\|_2^2 \tag{18}$$

The last expression provides us with the quantity m used in Eq. 14 to construct the upper and lower bounds to the score. The quantities involved in this bounding scheme are illustrated in Fig. 4 for a single part.



(a) Input (b) Exact score, s

(c) Second moment, m (d) Shufflet-based estimate, $\hat{s}$

(e) Lower bound, $\underline{s}$, $p_e = .1$ (f) Upper bound, $\overline{s}$, $p_e = .1$

Figure 4: Part score approximation and bounding: our goal is to rapidly bound the value of the part score s shown on the top right. The bound we propose in Eq. 14 is formed from two quantities, a shufflet-based approximation $\hat{s}$, in (d) and an estimate of the approximation error's second moment, shown in (c). These two are combined to form an interval that contains the actual value with a certain probability of error p_e. The values of the lower and upper bounds for $p_e = 0.1$ are shown in (e) and (f) respectively.

4.1. Integration with DPM detection

We now describe how we integrate the bound obtained above with the Dual-Tree Branch-and-Bound (DTBB) method [16, 17]; for lack of space, we refer to [16, 17] for details, and intend to provide a thorough presentation in a forthcoming technical report.

In [16] Branch-and-Bound is used to bypass Generalized Distance Transforms (GDTs); this can result in substantial speedups for the part combination phase. However in [16] we consider the part scores to be computed in advance of DTBB, while this is actually the main bottleneck in detection with DPMs. Instead, following [17], we now integrate into DTBB our efficient upper bound of the part scores.

In DTBB we rely on upper bounding the quantity:

$$\mu_d^s = \max_{x \in X_d} \max_{x' \in X'_s} s[x'] + B[x', x], \tag{19}$$

which indicates the maximal contribution from a set of part positions X'_s to a set of object positions X_d; $s[x']$ is the appearance term at part location x' and $B[x', x]$ is the geometric consistency between part x' and the object location x. Since $\max_{x \in X} f[x] + g[x] \leq \max_{x \in X} f[x] + \max_{x \in X} g[x]$, we can bound Eq. 19 as:

$$\mu_d^s \leq \underbrace{\max_{x' \in X'_s} s[x']}_{\overline{S}} + \max_{x \in X_d} \max_{x' \in X_d} B[x', x] \tag{20}$$

Further $\lambda_d^s = \min_{x \in X_d} \max_{x' \in X_s} s[x'] + B[x', x]$ is used in [16] and is lower bounded as:

$$\underbrace{\min_{x' \in X'_s} s[x']}_{\underline{S}} + \min_{x \in X_d} \max_{x' \in X_s} B[x', x] \leq \lambda_d^s. \tag{21}$$

The computation of the bounds relevant to the geometric term, $B[x', x]$ exploits the fact that X_s, X_d are rectangular, and is detailed in [16]. Coming to bounding the unary terms, the approach of [16] has been to compute the exact part scores at every location $s[x]$ and then obtain $\overline{S}, \underline{S}$. As the domains X_s are organized in a kd-tree the latter maximization can be rapidly performed in a fine-to-coarse manner, but the computation of $s[x]$ was not avoided.

Instead we propose to accelerate the computation of $\overline{S}, \underline{S}$ by initially sidestepping the computation of $s[x]$ using the probabilistic bound of Eq. 14: the terms $\underline{s} = \hat{s}[x] - \sqrt{m_x/p}$ and $\overline{s} = \hat{s}[x] + \sqrt{m_x/p}$ involved in Eq. 14 are with probability $1 - p_e$ lower and upper bounds of $s[x]$ respectively. Based on these we can upper and lower bound S as follows: $\overline{S} = \max_{x' \in X'_s} s[x'] \leq \max_{x' \in X'_s} \overline{s}[x']$ and $\underline{S} = \min_{x' \in X'_s} s[x'] \geq \min_{x' \in X'_s} \underline{s}[x']$, and thereby use $\overline{s}, \underline{s}$ as surrogates for $s[x]$ in DTBB.

We note that we use $\overline{s}, \underline{s}$ as surrogates for s only in the first phase of DTBB. As soon as Branch-and-Bound converges to singleton intervals, we evaluate the exact part scores, $s[x]$; as we show in the experimental section, this boosts performance when compared to using only the midlevel-based approximation. This more elaborate computation however is performed around a drastically reduced set of points, namely around those image locations that survive the first, quick bounding phase. Our method thereby combines the speed of shared part filter computation with the accuracy of DPMs.

4.2. Combination with Cascaded DPMs

In [8] the authors exploit the fact that the DPM score is expressed as the accumulation of the part scores to devise a cascaded detection algorithm: after computing the contribution of each part to the overall object score, the computation stops for any location where the sum falls below a conservative threshold.

In order to accelerate the first stage of their algorithm, [8] downproject the HOG features to a lower, five-dimensional space obtained through PCA. This results in fewer multiplications per HOG cell-bin, but can distort the obtained

result. The remedy used in [8] is to use separate conservative thresholds for the PCA-based part scores, and estimate them from the training data.

We can replace the PCA-based approximation of [8] with our faster, shufflet-based approximation. Instead of using $(36 \cdot 5)$ operations per location, we only need L operations, with L being 16, or 32. Even though we have the option of using our bounding-based scheme to compute upper bounds throughout, we can now avoid it altogether, since the empirically computed thresholds for cascades can take into account the distortion due to the shufflet approximation. This makes the part score computation even faster.

5. Results

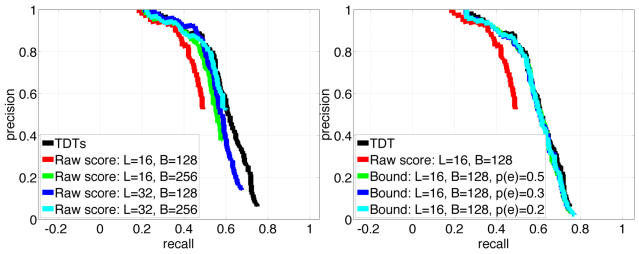
We start with a qualitative illustration of the merit of our approach using the bicycle class of the PASCAL VOC challenge; identical qualitative results have been obtained for the other classes, but are omitted to save space. We report Average Precision scores on all classes in Table 2.

On the left side of Fig. 5 we provide precision-recall curves for bicycle detection when using (a) the 'golden standard' Truncated Distance Transform (TDT) detector of [11] and (b) the raw shufflet score for different combinations of basis size, B, and expansion size, L (we use $2L$ terms for the larger root filters). As expected, increasing the size of the basis and the expansion improves performance, but there still remains a gap in performance.

However when taking even the weakest combination of B, L parameters ($B = 128, L = 16$) and combining it with the probabistic bounding scheme of the previous section, we observe on the right side that the performance of the shufflet-based detector becomes practically indisitnguishable from that of [11]. Moreover we observe that the exact value of the probability of error p_e is not that important, and the performace is robust even for relatively large values of p_e.

Coming to timing results, we provide in Table 3 timings gathered from 1000 images of the PASCAL VOC dataset, and averaged over all 20 categories. The first row indicates the time spent to compute part scores by the different methods, and the following rows indicate detection times. For more conservative threshold values the part score is fully evaluated at more points and the merits of having a quick first fast pass get eliminated. Actually for a threshold of -1.0 it turns out that the TDT-based implementation can be almost as fast; but our shufflet-based approximation turns out to be faster than both DTBB and TDT for moderate values of the threshold θ. In particular for $\theta = -.7$, or $\theta = -.5$ the shufflet-based variant of cascades requires approximately half the time of the PCA-based cascade, and a small fraction of the time of TDT-based detection.

We note that these timings are (i) for a single-threaded implementation and (ii) do not include steps that are shared by all classes, namely HOG pyramid construction, and shufflet convolutions. The latter typically cost 0.5 seconds and 1.1 seconds respectively (for 128 bases), but become quickly amortized when working with multiple categories. Finally, these timings may vary due to processor specifications; we will provide code to allow for the reproduction of the experiments on different machines at [1].



(a) Shufflet parameter variation (b) Raw vs bounding-based score

Figure 5: Precision-recall curves for bicycles on PASCAL VOC 2007. The 'golden standard' is the black curve (TDT, of [11]). The left plot shows the performance of the raw, shufflet-based score for different shufflet parameter combinations, ans the right plot shows the performance of the bounding-based scheme.

	TDT [11]	BB [16]	**Shuff-3**	**Shuff-1**
Part terms	2.26 ± 0.77	1.69 ± 0.18	0.41 ± 0.10	0.41 ± 0.10
$\theta = -0.5$	0.41 ± 0.06	0.21 ± 0.06	0.54 ± 0.22	0.82 ± 0.22
Sum	2.67 ± 0.83	1.90 ± 0.23	0.95 ± 0.24	1.23 ± 0.29
$\theta = -0.7$	0.41 ± 0.06	0.42 ± 0.10	0.74 ± 0.25	0.83 ± 0.26
Sum	2.67 ± 0.83	2.10 ± 0.24	1.15 ± 0.34	1.24 ± 0.35
$\theta = -1.0$	0.41 ± 0.06	1.31 ± 0.31	1.53 ± 0.24	1.71 ± 0.27
Sum	2.67 ± 0.83	3.00 ± 0.42	1.74 ± 0.34	2.12 ± 0.37

	TDTs	C-DPM	**C-Shufflet**
$\theta = -0.5$	2.67 ± 0.82	0.56 ± 0.07	0.23 ± 0.04
$\theta = -0.7$	2.67 ± 0.82	0.72 ± 0.09	0.34 ± 0.04
$\theta = -1.0$	2.67 ± 0.82	1.04 ± 0.16	0.57 ± 0.10

Table 3: Means and standard deviation timings, in seconds, of the considered approaches. The top table indicates full-blown detection, the bottom table indicates cascaded detection. TDT stands for truncated distance transform - [11], BB for Dual Tree Branch-and-Bound [16], and Shuff-{3,1} are the shufflet-based bounds for $p_e = .3$ and $.1$ respectively, with $L = 16, B = 128$. In the bottom table we compare the original Cascade-DPM model of [8] with our shufflet-based cascade.

6. Conclusion

In this work we have introduced *shufflets*, a shiftable basis for mid-level part representation, demonstrated its usefulness for part sharing in DPMs, and introduced probabilis-

	airplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	d.table	dog	horse	m.bike	person	p.plant	sheep	sofa	train	tv
[8]	0.331	0.593	0.103	0.155	0.265	0.520	0.537	0.224	0.200	0.243	0.268	0.124	0.565	0.483	0.433	0.133	0.208	0.358	0.450	0.421
Our work	0.328	0.588	0.101	0.153	0.261	0.513	0.532	0.221	0.201	0.241	0.269	0.122	0.561	0.481	0.428	0.130	0.202	0.352	0.448	0.418

Table 2: Average precision results on Pascal VOC, comparing the implementation of [8] to our shufflet-based cascade.

tic bounds to accommodate the effects of distortions due to approximations on this basis, thereby enabligh fast and accurate detection with DPMs. In future work we intend to explore how this basis can be exploited during training [2, 12, 27], incorporated in hierarchical models [18, 34] and used for scalable object detection [4], while also exploring connections with convolutional models [35, 20, 15, 19].

References

[1] http://vision.mas.ecp.fr/Personnel/iasonas/dpms.html. 2, 7

[2] Y. Aytar and A. Zisserman. Tabula rasa: Model transfer for object category detection. In *ICCV*, 2011. 3, 8

[3] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 1

[4] T. Dean, M. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. Yagnik. Fast, accurate detection of 100,000 object classes on a single machine. In *CVPR*, 2013. 2, 8

[5] S. Dickinson, A. Pentland, and A. Rozenfeld. 3-d shape recovery using distributed aspect matching. *PAMI*, 1992. 2

[6] C. Dubout and F. Fleuret. Exact acceleration of linear object detectors. In *ECCV*, 2012. 1, 2

[7] G. Ettinger. Large hierarchical object recognition using libraries of parameterized sub-parts. In *CVPR*, 1988. 2

[8] P. Felzenszwalb, R. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *CVPR*, 2010. 1, 2, 6, 7, 8

[9] P. Felzenszwalb and D. Huttenlocher. Pictorial Structures for Object Recognition. *IJCV*, 2005. 1

[10] S. Fidler and A. Leonardis. Towards Scalable Representations of Object Categories:Learning a Hierarchy of Parts. In *CVPR*, 2007. 3

[11] R. Girshick, P. Felzenszwalb, and D. McAllester. Discriminatively trained deformable part models, release 5. http://people.cs.uchicago.edu/ rbg/latent-release5/. 2, 7

[12] R. Girshick, H. O. Song, and T. Darrell. Discriminatively activated sparselets. In *ICML*, 2013. 1, 3, 8

[13] R. Grosse, R. Raina, H. Kwong, and A. Ng. Shift-invariant sparse coding for audio classification. In *UAI*, 2007. 4

[14] D. Jacobs. Groper: A grouping based object recognition system for two-dimensional objects. In *IEEE Workshop on Computer Vision*, 1987. 2

[15] K. Kavukcuoglu, P. Sermanet, Y.-L. Boureau, K. Gregor, M. Mathieu, and Y. LeCun. Learning convolutional feature hierarchies for visual recognition. In *NIPS*, 2010. 3, 8

[16] I. Kokkinos. Rapid deformable object detection using dual-tree branch-and-bound. In *NIPS*, 2011. 1, 2, 6, 7

[17] I. Kokkinos. Bounding Part Scores for Rapid Detection with Deformable Part Models. In *2nd Parts and Attributes Workshop, ECCV*, 2012. 1, 2, 4, 6

[18] I. Kokkinos and A. Yuille. Inference and Learning with Hierarchical Shape Models. *IJCV*, 2011. 4, 8

[19] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS*, 2012. 3, 8

[20] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML*, 2009. 3, 8

[21] B. Leibe, A. Leonardis, and B. Schiele. Combined Object Categorization and Segmentation with an Implicit Shape Model. In *ECCV*, 2004. SLCV workshop. 2

[22] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *ICML*, 2009. 3

[23] K. Mikolajczyk, B. Leibe, and B. Schiele. Multiple object class detection with a generative model. In *CVPR*, 2006. 2

[24] M. Mitzenmacher and E. Upfal. *Probability and computing - randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005. 5

[25] A. Opelt, A. Pinz, and A. Zisserman. Boundary-fragment-model for object detection. In *CVPR*, 2006. 2

[26] P. Ott and M. Everingham. Shared parts for deformable part-based models. In *CVPR*, 2011. 3

[27] H. Pirsiavash and D. Ramanan. Steerable part models. In *CVPR*, 2012. 1, 2, 3, 8

[28] N. Razavi, J. Gall, and L. V. Gool. Scalable multi-class object detection. In *CVPR*, 2011. 2, 3

[29] H. O. Song, S. Zickler, T. Althoff, R. B. Girshick, M. Fritz, C. Geyer, P. F. Felzenszwalb, and T. Darrell. Sparselet models for efficient multiclass object detection. In *ECCV*, 2012. 1, 2, 3, 4, 5

[30] A. Thomas, V. Ferrari, B. Leibe, T. Tuytelaars, B. Schiele, and L. V. Gool. Towards multi-view object class detection. In *CVPR*, 2006. 2

[31] S. Todorovic and N. Ahuja. Learning subcategory relevances for category recognition. In *CVPR*, 2008. 3

[32] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing Visual Features for Multiclass and Multiview Object Detection. In *CVPR*, 2004. 2

[33] A. Vedaldi and A. Zisserman. Sparse Kernel Approximations for Efficient Classification and Detection. In *CVPR*, 2012. 1, 2

[34] Y. J. Xi Song, Tianfu Wu and S.-C. Zhu. Discriminatively trained and-or tree models for object detection. In *CVPR*, 2013. 8

[35] M. D. Zeiler, G. W. Taylor, and R. Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *ICCV*, 2011. 3, 8

[36] L. Zhu, Y. Chen, A. Torralba, W. T. Freeman, and A. L. Yuille. Part and appearance sharing: Recursive compositional models. In *CVPR*, 2010. 3

[37] S. C. Zhu and A. Yuille. FORMS: A Flexible Object Recognition and Modeling System. *IJCV*, 20(3), 1996. 2