Deformable Part Models with Individual Part Scaling

Charles Dubout
charles.dubout@idiap.ch
François Fleuret
francois.fleuret@idiap.ch

Computer Vision and Learning Group
Idiap Research Institute
Martigny, Switzerland

École Polytechnique Fédérale de
Lausanne, Switzerland

Abstract

Current deformable part models such as the ones introduced by Felzenszwalb *et al*. let the parts deform only at a fixed predetermined scale relative to that of the root of the models (typically at twice the resolution). They do so because it allows them to find the optimal placement of each part efficiently, using a fast 2D distance transform algorithm.

We demonstrate in this paper that if one settles for approximately optimal placements, it is possible to efficiently deform the parts across scales as well. Allowing parts to move in 3D increases the expressivity of the models, allowing them to compensate for a wider class of deformations, and might approximate an increase in the scanning resolution. As the number of parameters remains (nearly) constant, overfitting is not a problem.

1 Introduction

The Deformable Part Model (DPM) of Felzenszwalb *et al*. [7, 11] and its many variants, some of which are presented in §2, are considered one of the current state-of-the-art object detection methods. Indeed they are the winners of many Pascal VOC detection challenges [5, 6], and are the current top-performer on many other detection tasks, *e.g*. pedestrian detection [11], bird recognition [18], face detection and feature localization [23], or articulated pose recognition [19].

DPMs are discriminative sliding-window classifiers, predicting a score related to the presence or absence of an object for each possible position and scale of an image. These scores are computed by taking the sum of the inner products between the model's filters and the corresponding sub-windows of the image, placing each filter at an "optimal" image location. The strength of DPMs resides in their ability to represent an exponential number of templates by letting the part filters float around their reference location, and in finding the optimal part configuration at every possible root position efficiently using a generalized distance transform [8].

The bulk of their computational cost comes from the numerous convolutions they need to do between every feature pyramid levels and every part filters, each followed by a distance transform. Both can be computed in time linear with the area of the pyramid levels and take roughly the same amount of time [3]. Standard DPMs restrict the parts to move at a single scale, a limitation imposed by the distance transform as explained in §3.3. Our

extension removes this limitation without increasing the number of convolutions or distance transforms, by sacrificing the guarantee of the optimality of the part placements. Its cost was empirically found to be similar to that of a second distance transform, and it is easy to integrate into existing detection systems, the models staying the same except for the additional 3D components to the deformation cost vectors.

After a more formal presentation of DPMs in §3.1, we introduce our 3D model in §3.2, and our approximation to the generalized distance transform in §3.3. In §4 we present results showing an average relative accuracy improvement of 15% compared to standard models, and show that our approximation does not lead to any significant loss of performance by comparing against an exact baseline searching for the optimal part locations exhaustively. It is available under the GPL open source license at http://www.idiap.ch/scientific-research/resources.

2 Related Works

Many recent works build upon the original DPM of Felzenszwalb *et al*. and try to improve either its detection performance or its computational complexity. As an instance of a work belonging to the first category, [21] augments the HOG features usually used with LBP ones, in order to be sensitive to not only edges but also textures, which results in a 10% gain in average relative accuracy. Trying to simplify the original star-based part model, [22] represents objects by a mixture of hierarchical tree models organized on a 2D grid, where the nodes represent object parts, and solves the non-convex optimization problem using the Concave-Convex Computational Procedure (CCCP) [20]. Arguing that the most important component of DPMs is the mixture one, [2] proposes to improve their initialization by switching from aspect-ratio to appearance clustering, and reports that a mixture of monolithic models clustered by appearance can compete with DPMs.

A selection of the most relevant works aiming at making DPMs faster include [10], which sees the parts as classifiers in a cascade, and splits the detection process into two passes. The first pass evaluates the detector on a low-dimensional feature space (reduced from 32 HOG features to 5 using PCA), and the second pass with all the features. Making use of the linearity of the Fourier transform, [3] shows how to accelerate by one order of magnitude the many convolutions between the feature pyramid and the part filters, exactly and without the need to tune any parameter. Expressing the part filters as a sparse linear combination of a dictionary, [17] can also obtain large speedups when detecting multiple objects simultaneously, since in this case the dictionary typically can be made much smaller than the total number of filters while not sacrificing too much accuracy, as many objects share visually similar parts, *e.g*. wheels, limbs, corners, *etc*. Finally [13] applies the dual-tree branch and bound algorithm [14] to more efficiently optimize the objective function of [10], and rapidly approximates the inner products between filters and HOG features by quantizing the HOG cells onto a codebook and replacing their inner products with lookups of precomputed scores in [13].

Another line of work [15, 16] aims at bridging the gap between 2D image positions and 3D real-world ones by learning 3D part deformation models. This is accomplished by learning a mixture model where each mixture component deals explicitly with a particular viewpoint, each trained using both real and 3D synthesized images. Even though it enables the model to map 2D part locations to 3D ones, the authors did not attempt to move the parts across scales, thus making their work completely orthogonal to ours.

3 Method

Standard DPMs comprise a root and several parts, all detected independently by a linear filter, and organized in a hierarchical structure. In the rest of this paper we restrict ourselves without loss of generality to star structures, for ease of notation. We also ignore the additional complexity introduced by the mixture over the models, since it is orthogonal to the method presented here.

3.1 Standard Models

Let H be a feature pyramid and $\mathbf{p} = (x, y, z)$ specify a 2D position (x, y) in the z-th level of the pyramid. Let $\phi(\mathbf{p})$ denote the vector obtained by concatenating the feature vectors in the sub-window of H centered at $\mathbf{p}$, of dimensions always clear from the context (the dimensions of the filter it is multiplied with), and $\phi_d(\mathbf{p})$ be the deformation features.

A model for an object with n parts is composed of a root filter $\mathbf{w}_0$ and n pairs $(\mathbf{w}_i, \mathbf{d}_i)$, where $\mathbf{w}_i$ is the filter of the i-th part and $\mathbf{d}_i$ is a vector specifying the deformation cost of the part placement.

An object hypothesis $\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n$ specifies the location of the center of each filter in a feature pyramid, where the parts are constrained to move in the same level as the root[1]. The score of a hypothesis is then given by the score of each filter at its respective location, minus a deformation cost that depends on the location of each part with respect to the root position,

$$S(\mathbf{p}_0, \ldots, \mathbf{p}_n) = \mathbf{w}_0^\mathsf{T} \phi(\mathbf{p}_0) + \sum_{i=1}^{n} \mathbf{w}_i^\mathsf{T} \phi(\mathbf{p}_i) - \mathbf{d}_i^\mathsf{T} \phi_d(\mathbf{p}_i - \mathbf{p}_0). \tag{1}$$

The deformation features are typically,

$$\phi_d(\mathbf{p}) = (1, x, y, x^2, y^2), \tag{2}$$

in which case it is possible to find the optimal location of each part $\mathbf{p}_i^\star(\mathbf{p}_0)$ as a function of the root position $\mathbf{p}_0$ efficiently (i.e. in time linear with the total number of locations), using a generalized distance transform [8],

$$\mathbf{p}_i^\star(\mathbf{p}_0) = \underset{\mathbf{p} \in \mathbb{Z}^2 \times \{z_0\}}{\operatorname{argmax}} \; \mathbf{w}_i^\mathsf{T} \phi(\mathbf{p}) - \mathbf{d}_i^\mathsf{T} \phi_d(\mathbf{p} - \mathbf{p}_0), \tag{3}$$

provided that the part locations are integral and limited to a single scale. The score of a root position is then,

$$S(\mathbf{p}_0) = \mathbf{w}_0^\mathsf{T} \phi(\mathbf{p}_0) + \sum_{i=1}^{n} \mathbf{w}_i^\mathsf{T} \phi(\mathbf{p}_i^\star(\mathbf{p}_0)) - \mathbf{d}_i^\mathsf{T} \phi_d(\mathbf{p}_i^\star(\mathbf{p}_0) - \mathbf{p}_0). \tag{4}$$

3.2 Extension to 3D

In our algorithm, we allow the parts to move freely across scales, and extend the deformation features of Equation (2) to include the z component of the disparity between root and parts positions,

$$\bar{\phi}_d(\mathbf{p}) = (1, x, y, z, x^2, y^2, z^2). \tag{5}$$

[1] Or at a scale which is an integral multiple of the root scale, *e.g.* twice the root resolution in [10], since in this case the root positions are a subset of the part ones. In any case this scale is predetermined.

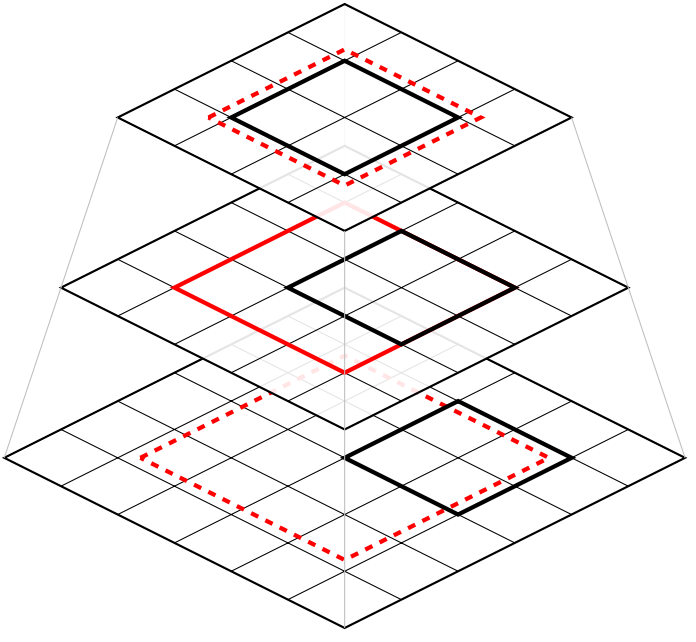

Figure 1: In the middle level of this illustration of a feature pyramid and drawn in solid red is the outline of a root. In the levels above and below and drawn in dashed red are the outlines of the same root scaled to correspond to the same rectangle in the image. In black is the outline of a part deforming across scales. The size of the part is always the size of its filter, here 2×2 HOG cells, which means that it becomes bigger relative to the root in the top level and smaller in the bottom one.

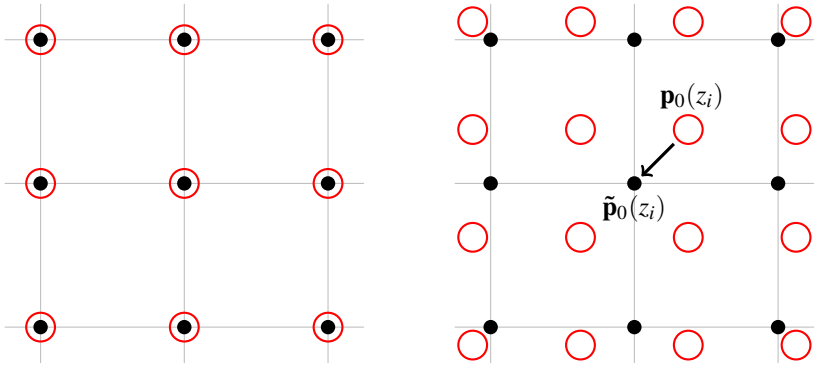

Figure 2: Lattices of part locations (in black) in a particular pyramid level. The red circles indicate root positions. In a) the part and root positions are at the same scale, as is always the case with standard models. In b) there is a mismatch between the scales of the two, and we show how we approximate a root position $\mathbf{p}_0(z_i)$ by rounding it to the closest integral position $\tilde{\mathbf{p}}_0(z_i)$ when looking for its optimal part placement.

An illustration of the consequences of allowing parts to move across scales is available in Figure 1. Ideally, one would like to now find the optimal location of each part in this way,

$$\bar{\mathbf{p}}_i^\star = \underset{\mathbf{p}\in\mathbb{Z}^3}{\operatorname{argmax}}\, \mathbf{w}_i^\mathsf{T}\phi(\mathbf{p}) - \mathbf{d}_i^\mathsf{T}\bar{\phi}_d(\mathbf{p}-\mathbf{p}_0(z_i))\,, \tag{6}$$

where $\mathbf{p}_0(z_i) = (\lambda^{z_i-z_0}x_0, \lambda^{z_i-z_0}y_0, z_0)$ are the coordinates of the root position in the i-th part's level z_i, λ being the scaling factor between two successive levels of the feature pyramid. The x and y coordinates of the root position need to be rescaled since they are defined in a different level than the part's coordinates, each level z of the feature pyramid storing features extracted from the image scaled by a factor λ^z. We compare the root and part locations in the part's level, but the particular level at which they are compared does not matter, since each deformation cost $\mathbf{d}_i$ can be scaled accordingly during training.

3.3 Approximation to the Generalized Distance Transform

Unfortunately, $\mathbf{p}_0(z_i)$ is likely to be non-integral, and the generalized distance transform thus cannot be used directly anymore. Another issue is that since pyramid levels are of varying sizes, one cannot extend the distance transform to work across levels in the same way as it works across 2D locations. In order to cope with the first issue, we approximate the root position at the scale of the i-th part by the closest integer one,

$$\tilde{\mathbf{p}}_0(z_i) = \underset{\mathbf{p}\in\mathbb{Z}^2\times\{z_i\}}{\operatorname{argmin}}\, ||\mathbf{p}-\mathbf{p}_0(z_i)||\,. \tag{7}$$

Using this approximate root position, we can now again use the generalized distance transform in order to find the optimal part location for the approximate root position,

$$\tilde{\mathbf{p}}_i^\star(\mathbf{p}_0) = \underset{\mathbf{p}\in\mathbb{Z}^3}{\operatorname{argmax}}\, \mathbf{w}_i^\mathsf{T}\phi(\mathbf{p}) - \mathbf{d}_i^\mathsf{T}\bar{\phi}_d(\mathbf{p}-\tilde{\mathbf{p}}_0(z_i))\,. \tag{8}$$

We expect this location to coincide most of the time with the optimal one for the real root position, the difference between the real and the approximate one being at most 0.5 along the x and y axes. However, the optimal score returned by the transform will generally not match the score of any real root and part configuration, and might even be higher than the true optimal one, so we recompute it in constant time using this time the real root position $\mathbf{p}_0(z_i)$,

$$\tilde{S}(\mathbf{p}_0) = \mathbf{w}_0^\top \phi(\mathbf{p}_0) + \sum_{i=1}^{n} \mathbf{w}_i^\top \phi(\tilde{\mathbf{p}}_i^\star(\mathbf{p}_0)) - \mathbf{d}_i^\top \bar{\phi}_d(\tilde{\mathbf{p}}_i^\star(\mathbf{p}_0) - \mathbf{p}_0(z_i)), \tag{9}$$

in order to obtain a lower bound on the true optimal score.

The second issue was that the generalized distance transform cannot be extended to work across pyramid levels. Since we can generally expect the number of scales in which parts will deform to be small (much smaller than the number of possible 2D locations at each scale), we brute-force search the optimal level z_i of each part, and for each level use an efficient 2D distance transform. Brute-force searching the optimal level also enables the use of costs other than the quadratic one of Equation (5), as long as they remain separable in all dimensions. The results of the transform of Equation (8) can be reused for each root level with common part levels, such that by precomputing them for every part level in advance, the total number of transforms is reduced from being quadratic to linear in the total number of pyramid levels.

4 Experiments

To evaluate our approach to increase the expressivity of DPMs by allowing parts to also move across scales, we trained a mixture of 6 models on all 20 classes of the Pascal VOC 2007 challenge [5]. We compare both against standard 2D models as well as "exact" 3D ones, searching exhaustively for the optimal part placement instead of using our approximation of §3.3. We based our implementation on our publicly available system [3, 4], which we extended to also deal with the training of the models.

4.1 Implementation

Our DPM implementation uses the same modified Histogram of Oriented Gradients (HOG) features [11] and the same initialization of the parts locations, sizes, deformation cost, and left/right pose assignments as in [9, 10]. It similarly initializes the parts at twice the resolution of the root (one octave below), and we configured it to always compute 5 scales per octave in the feature pyramid. The only additional parameter relative to the initialization of the 3D models that we needed to specify was the initial deformation cost of the parts, corresponding to the z-coordinate of each vector $\mathbf{d}_i$. We set their linear components to 0 and their quadratic one to 0.01, such that the initial dispersion of parts across scales was approximately centered and of standard deviation 1 level.

While brute-force searching the optimal scale of each part, during both training and testing, we restricted the search to a 7 levels window (± 3 levels) centered on the level one octave below the root, which corresponds to $z_i \in [z_0 - 5 - 3, \ldots, z_0 - 5 + 3]$, meaning that we allowed parts to grow or shrink at most by a factor of $2^{\frac{3}{5}} \approx 1.5$ compared to their reference size. This setting proved sufficient for parts to fully exploit their additional freedom along the z-axis given our initialization of the deformation cost. Since there is some randomness

	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table
voc-release4 (AP)	28.9	60.2	1.7	8.3	20.6	53.5	51.3	6.9	18.7	20.1	13.8
2D DPM (AP)	30.3	57.7	4.4	11.4	25.2	55.0	53.3	11.1	19.2	22.5	24.4
3D DPM (AP)	33.5	59.4	6.9	13.1	28.7	59.0	52.9	19.5	20.6	26.8	25.9
Rel. gain (%)	10.6	2.8	55.4	15.1	13.9	7.2	-0.8	75.9	7.3	19.4	6.3

	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean
voc-release4 (AP)	3.3	54.5	47.6	38.8	5.8	14.3	28.1	37.3	39.0	**27.6**
2D (AP)	5.0	59.2	48.5	35.7	8.6	18.8	28.4	42.3	42.2	**30.2**
3D (AP)	6.5	61.2	48.9	32.9	11.1	21.5	31.7	44.5	43.8	**32.4**
Rel. gain (%)	29.2	3.5	0.8	-7.7	29.5	14.3	11.6	5.3	3.9	**15.2**

Table 1: Pascal VOC 2007 challenge Average Precision (area under the Precision/Recall curve) comparison for the models of [9] as well as our 2D and 3D models. What we call relative gain is the improvement of 3D models over 2D ones.

involved in the initialization of the models, we always initialized the seed of the random number generator to the same value while training 2D and 3D models.

To demonstrate the performance of our implementation, we also evaluated the models included in [9], which achieve close to state-of-the-art detection results, using the same evaluation parameters as our models. These evaluation parameters might not be optimal for those models, and we therefore include their results as a reference point only.

4.2 Results

The performances of all 3 kinds of models on the Pascal VOC 2007 challenge are displayed in Table 1. The scoring function of the Pascal VOC development kit computes the average precision score for each model by sampling the Precision/Recall curve in eleven points of recall $0.0, 0.1, 0.2, \ldots, 1.0$. In order to increase its precision, which is particularly important for difficult classes obtaining less than 10% AP, we modified it to take into account all points of the PR curve. This modification explains why the scores we obtain on some difficult classes are lower than usually reported, and why some authors obtain scores above 9% AP while correctly detecting only one object on the whole data-set (since the first point is sampled with recall 0.0, as long as the first detection is correct the AP is guaranteed to be at least $\frac{1}{11}$).

The average precision of our 3D models improves over the 2D ones for 18 of the 20 classes, increasing on average by more than 15%. The average time taken by our implementation to detect objects in an image using one of the 2D models was 77 ms. Out of those 77 ms, we measured that 22 were spent computing distance transforms. When using a 3D model, the average total time increased to 99 ms, corresponding to a doubling of the transform time.

Apart from increasing the expressivity of the models, allowing parts to move across scales might also improve performance by simulating a scan of the image at a higher resolution. This may happen because HOG grids in neighboring pyramid levels have always the same step size (typically 8 pixels), but slightly different dimensions. This difference intertwines their positions on the image as in Figure 2 b), and may make the whole process similar to searching for objects on several slightly misaligned HOG grids.

	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table
Brute-force (AP)	50.1	69.0	18.0	20.9	38.8	72.7	59.1	33.0	28.8	46.5	47.3
Approx. DT (AP)	49.9	69.2	17.9	21.5	38.5	72.7	59.1	32.4	28.7	46.3	47.3

	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean
Brute-force (AP)	18.7	77.2	60.3	32.9	23.9	37.6	44.3	56.0	64.3	**45.0**
Approx. DT (AP)	18.4	77.2	60.3	33.0	24.6	37.2	44.3	56.0	64.2	**44.9**

Table 2: Pascal VOC 2007 challenge AP comparison on the first 100 images of each class for the exact as well as our approximation to the generalized distance transform method of §3.3 using our 3D models.

A comparison of our approximate method versus the "exact" one which brute-force searches the optimal location of each part is shown in Table 2. We evaluated both methods only on the first 100 images of each class because of the prohibitive time taken by the exhaustive search. These results demonstrate the accuracy of the approximation.

5 Conclusion

The idea motivating our work is to make full use of all the convolutions between pyramid levels and part filters evaluated in DPMs, reusing them to deform parts across multiple scales. The extension we presented increases on average the detection accuracy of the models by 15% for a moderate augmentation of its total computational cost, the number of convolutions and distance transforms remaining constant. Despite relying on an approximation to the generalized distance transform, our approach obtains scores virtually equal to its exact but much slower counterpart.

Acknowledgements

Charles Dubout was supported by the Swiss National Science Foundation under grant 200021-124822 – VELASH, and François Fleuret was supported in part by the European Community's 7th Framework Programme under grant agreement 247022 – MASH.

References

[1] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 886–893, 2005.

[2] S. K. Divvala, A. A. Efros, and M. Hebert. How important are "deformable parts" in the deformable parts model? In *European Conference on Computer Vision*, pages 31–40, 2012.

[3] C. Dubout and F. Fleuret. Exact acceleration of linear object detectors. In *European Conference on Computer Vision*, pages 301–311, 2012.

[4] C. Dubout and F. Fleuret. Exact acceleration of linear object detectors. http://www.idiap.ch/scientific-research/resources/exact-acceleration-of-linear-object-detectors, 2012.

[5] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html, 2007.

[6] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2011 (VOC2011) Results. http://www.pascal-network.org/challenges/VOC/voc2011/workshop/index.html, 2011.

[7] P. F. Felzenszwalb and D. Huttenlocher. Pictorial structures for object recognition. In *International Journal of Computer Vision*, 2005.

[8] P. F. Felzenszwalb and D. P. Huttenlocher. Distance Transforms of Sampled Functions. Technical report, Cornell Computing and Information Science, 2004.

[9] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. Discriminatively trained deformable part models, release 4. http://people.cs.uchicago.edu/ pff/latent-release4/.

[10] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2241–2248, 2010.

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

[12] I. Kokkinos. Rapid deformable object detection using dual tree branch and bound. In *Advances in Neural Information Processing Systems*, 2011.

[13] I. Kokkinos. Bounding part scores for rapid detection with deformable part models, 2012.

[14] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Efficient subwindow search: A branch and bound framework for object localization. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 31, pages 2129–2142, 2009.

[15] B. Pepik, P. Gehler, M. Stark, and B. Schiele. 3D^{2}PM - 3D deformable part models. In *European Conference on Computer Vision*, 2012.

[16] B. Pepik, M. Stark, P. Gehler, and B. Schiele. Teaching 3d geometry to deformable part models. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[17] H. Pirsiavash and D. Ramanan. Steerable part models. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[18] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

[19] Y. Yang and D. Ramanan. Articulated pose estimation using flexible mixtures of parts. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[20] A. Yuille, A. Rangarajan, and A. L. Yuille. The concave-convex procedure (cccp. In *Advances in Neural Information Processing Systems*, 2002.

[21] J. Zhang, K. Huang, Y. Yu, and T. Tan. Boosted local structured hog-lbp for object localization. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1393–1400, 2011.

[22] L. L. Zhu, Y. Chen, A. Yuille, and W. Freeman. Latent hierarchical structural learning for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1062–1069, 2010.

[23] X. Zhu and D. Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2879–2886, 2012.