Attentional Object Detection: A Proposal

Sergey Karayev

updated: 22 April 2011

Abstract

This document tracks the development of ideas for efficient object detection using the idea of attention: a sequential process of looking for something somewhere. We follow the Deformable Part Model approach to object detection and extend it in several ways, mostly focusing on the role of context, toward the goal of Anytime detection performance. I expect a submission to NIPS'11 to be whittled down from the text and results here.

Contents

1 Introduction

The task of *object detection* as most commonly formulated entails simultaneous recognition and localization of "things" in a static image of a natural scene. Each object is most commonly assumed to belong to one of a fixed set of classes; its localization is approximated by placing a bounding box around pixels belonging to it. Datasets of such annotations by humans are used for evaluation of detection algorithms [1].

Most object detection systems can be subdivided into three parts: a way to propose regions of the image, a way to evaluate a given region, and a way to post-process the results. For example, the deformable parts model of Felzenszwalb *et al.* proposes image regions with an exhaustive sliding window, evaluates them with a window-sized gradient-based feature and a linear SVM (in addition, it fits parts to the model in a Latent-SVM formulation), and uses detection context to prune some detections in post-processing [2, 3]. Another approach, Multiple Kernels of Vedaldi *et al.*, proposes windows in a cascaded manner, beginning with *jump windows* of Chum and Zisserman [4], and evaluates them with bag-of-words non-linear SVM kernels [5]. Both systems have recently shown state-of-the-art performance on the most commonly used detection challenge dataset [1], and have pointed the direction for much subsequent research.

These approaches are quite slow. In the case of the deformable parts model, detection is on the order of a few seconds; recent work to speed it up with coarse-to-fine search and cascaded parts puts it on the order of less than a second (**is that with or without feature computation?**), at slight hit to the accuracy [6, 3]. Multiple Kernels consider increasingly fewer region proposals with increasingly more powerful SVMs; in the end, more than a minute is spent per image [7]. At these rates, real-time detection is problematic.

We argue that the solution to robust but fast object detectors in the notion of attention—meaning a sequential process of looking for something somewhere (as opposed to looking for everything everywhere). We suggest that to work on detection approaches scalable to many classes and to sequential frames of videos, the field needs to consider a new evaluation metric that encourages maximization of correct detections as early as possible in the detection process, given either a fixed or stochastic deadline.

This paper reviews past work toward this goal and formulates a novel method for maximizing performance of a detector with regard to the new evaluation. Along the way we explore and report on the effects of various speedups of current state-of-the-art detection methods. When given as much time as the current best detectors take, our system performs as well as them; when the detectors are given a fixed, short deadline, our system outperforms all others.

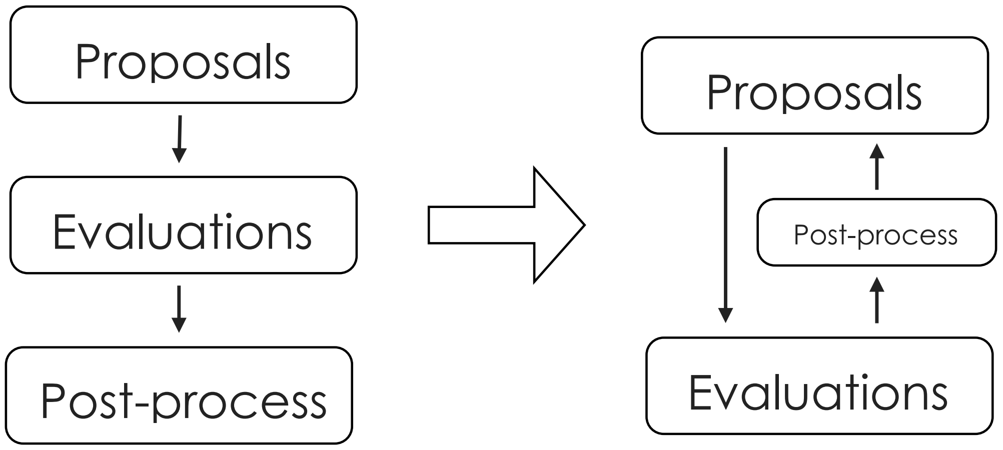
2 Related Work

The literature on object detection is vast. Here we briefly summarize work relevant to our contribution.

An early success in efficient object detection used simple Haar features to build up a *cascade* of classifiers, which then considered image regions in a sliding window regime [8]. Plentiful later work improved the behavior of the cascade while maintaining the basic idea [9]. **more refs** This detection method is fast, but the simple features and classifiers used have not led to the best performing detectors.

The best performance has recently come from detectors that use gradient-based features to represent either local patches or object-sized windows; if local patches are used, it appears important to include additional feature channels such as shape and color; if object-sized windows are used, finer-scale "parts" in object representation boost performance significantly.

Figure 1: Most object detection approaches have three parts: region proposal, region evaluation, and post-processing of detections. The parts are in a sense sequential, which leads to the influence of cues like context being deferred to the post-processing stage, and to a dependency of early detections on the goodness of proposals. We present a system that considers region proposal and evaluation jointly, such that it is able to output the best detections early in the process.



Taking a grander view, most current detection systems can be conceptualized as having three parts, as visualized in Figure 1.

The region proposal is usually done exhaustively over the image space, and doing so in an efficient order has not received much attention in the literature. Using jump windows as region proposals is one common idea [4, 7]. For local features, a bounded search over the space of all possible windows works well (especially for single-object detection) [10]. The method requires derivation of bounds, which have not yet been developed for HOG-based detectors, limiting the method's usefulness in state-of-the-art systems. Similar ideas have been formulated in a decision theoretic way [11].

Region proposal through segmentation is another common idea [12]. **more here.**

Another idea is to use a class-independent measure of "objectness" to reject much of the hypothesis space before running any class-specific detectors [13, 14]. However, the latter approach has not been shown to be actually faster than exhaustive evaluation, either because the proposal step becomes too expensive [14], or because the system was not coded efficiently [13].

In search of efficient detectors, the general "cascade" idea of breaking up the evaluation of an image region into thresholded steps has been applied to current state-of-the-art systems [5, 2]. In the case of the Multiple-Kernel Detector [5], the cascade is necessary to keep detection of a single class to the order of a minute per image. In the case of the Deformable Parts Model [2], the cascade speeds up detection from several seconds to under a second per class, with little loss in accuracy.

Evaluation of regions involves feature computation, which is a time consuming step.

Efficient feature computation for HOG-based detectors has been explored in [15]. Another idea is coarse-to-fine model evaluation. A recent work applies coarse-to-fine evaluation and adds a feedback arrow from post-processing to proposals, obtaining an order of magnitude speedup in the deformable part models framework while losing some accuracy [6].

Other systems that add feedback arrows to the conceptual diagram in Figure 1 are often inspired by biological vision and sequential decision process ideas [16, 17, 18]. **more**

A close relative to this idea is inspired by the problem of tracking objects. A recent paper draws an interesting feedback arrow from Evaluations to Proposals by using iterative sampling of a hypothesized mixture of Gaussians to find multiple objects in the image [19].

Anytime performance in vision systems is a surprisingly little-explored idea. A pioneering recent paper picks features with maximum value of information in a Hough-voting framework, and explicitly evaluates itself with regard to time [20].

Multi-class detection has its own line of work, focusing largely on detection time sublinear in the number of classes through sharing features [21, 22, 23]. An interesting reinforcement learning approach in a cascade framework is taken in [24]. **more**

A recent post-processing extension to detection systems uses structured prediction to incorporate multi-class context as a principled replacement for the common step of non-maximum suppression [25]. This work aims to tie up a couple of loose threads of object detection: 1) joint multi-class detection, with the mutual exclusion and contextual effects it entails; 2) the hack of non-maximum suppression of single-class detections.

Context has a long history in vision. One source of context is the scene or non-detector cues; for the PASCAL VOC, these are quantitatively considered in [26]. Another source is inter-object context, used for detection in a random field setting in [27]. **more**

Brief review of attentional modeling work [28, 29, 30]. The concept of saliency has been used for object classification [31].

todo: active recognition [32, 33].

The two papers closest to our contribution are principled multi-class structured prediction [25] and the detection under bounded resources work [20]. We significantly extend the first work, which assumes that all classes have been evaluated at all regions, to the online case of picking regions and classes to evaluate. We are guided by the same motivation of Anytime performance as the second work, but in a significantly more powerful detection regime. Although we rely on the commonly used deformable parts model detector, our method can use any feature extraction and classification method.

3 Time-sensitive evaluation

Average Precision (AP) has become a standard evaluation for detector performance on challenging datasets. AP is the area under the Precision vs. Recall curve, which is obtained by varying the threshold on the confidence of the detector.

Just as we care about the performance of our system at different thresholds of detection, we should also care about performance as the system is given different amount of time to output detections. Accordingly, we plot the P-R *surface* instead of the P-R curve, with time as one of the axes. The surface can be integrated along the Recall dimension to yield an AP vs. time curve.

The goal of our scheme is to schedule computation optimally, so that the largest part of the detection performance is recovered early on. In the limit of infinite time, we still want optimal performance—the performance of our system should not degrade as it is given more time. Accordingly, our goal is *Anytime* performance starting at some fixed deadline.

todo: think about the difference between per-dataset AP and average per-image AP. see relevant section in [25]: the latter is higher and makes more sense for us.

4 Problem Formulation

Our goal is to output the best detections at some deadline F, and to continue having the best detections at every time step after that. Specifically, we seek to maximize the AP of a detector for all classes in all images of a test dataset. **but see todo note in section 3.**

In the limit of infinite time, our program will look everywhere for everything, but given a short deadline, it should first look in the most promising places for the most likely objects. To take advantage of semantic and spatial context, where it looks at time t should depend on the results gathered by $t-1$.

In each of the I images we consider, we look for K classes of objects, and have detectors d_k, with each one having some number P_k of part fittings, cascade stages, or other partial classifiers c_{kp}.

We represent an image as a collection of overlapping windows at different scales: a multi-scale pyramid of N locations $l_i = (x, y, s)$ (the discretization and feautirzation of the image pyramid is discussed in section 5). We assume that each location ℓ can be the center of at most one bounding box, containing an object belonging to one of K classes. While this assumption may not hold in cases of occlusion, it does hold for the data we consider and at the levels of discretization we use. **is this true?**

There are two ways we could think about our task. The first can be called "assigning content to locations." It fits the sliding-window paradigm. The second can be called "assigning locations to content." It is inspired by particle filtering and the like [19]. We'll mostly talk in the paradigm of the former, but these are two sides of the same coin.

We first describe a partially observable Markov decision process representing our problem. Although we do not commit to using the methodology of solving POMDPs, this section is important, as it describes the specific models we assume, which even a hard-coded solution will use.

We can consider three variants of object detection as a sequential decision problem:

1. Looking for everything at a location: Selecting a location i, and running a black box program P there that evaluates all detectors as it sees fit.
2. Looking for something at a location: Selecting a location i and a detector d_k. Picking the sequence of partial classifiers for d is then a black box subproblem.
3. Looking for partial evidence of something at a location: Selecting a location i and a partial classifier c_{kp}.

We will be working mostly at the level of the third problem in the following sections, although our initial implementations may start out at lower levels.

4.1 POMDP

We face a Partially Observed Markov Decision Process (POMDP), which we define as the tuple $(S, A, O, b_o, T, \Omega, R, \gamma, F)$, where:

- S is a set of discrete states, A is a set of discrete actions, and O is a set of continuous observations.
- b_o is the initial belief state.
- $T(s, a, s') = P(s_{t+1} = s | a_t = a, s_t = s)$ is the distribution describing the probability of transitioning from state s to state s' upon taking action a.
- $\Omega(o, s, a) = P(o_{t+1} = o | a_t = a, s_{t+1} = s)$ is the distribution describing the probability of observing o from state s after taking action a.
- $R(s, a)$ is the reward signal received when executing action a in state s.
- F is the deadline to termination.

Our goal is to learn the optimal policy π, which is a mapping from belief states to actions.

4.2 State representation

For all approaches described later, the state $s \in S$ is fully defined by the time T, and the class of the bounding box centered at each location. We augment the K classes with a "background" class. Note that each object in the image is assigned to exactly one location. The number of states is exponential in N, which is on the order of 10000 in our applications. We are dealing with a static image and our actions do not change the underlying state, so $T(s, a, s') = \delta_{s,s'}$ where δ is the Kronecker delta function.

The belief state is supposed to be a distribution over "physical" states. We model the belief state as a Conditional Random Field (CRF) over the N locations in the image pyramid, where a node $y_i \in Y$ is an integer $0 \dots K$, according to the class whose bounding box we believe is centered at location i (background is 0).

For each node y_i, there are $\sum_{k=1}^{K} P_k$ nodes z_{ikp}, which represent a classifier's confidence in the corresponding y_i node **(in some way)**. Two nodes y_i and z_{ikp} are connected with weight w_{kp}. This weight in a sense represents our confidence in the classifier $c_k p$. We also leave open the possibility of extra nodes $z_{ikp'}$ to model contextual cues or *a priori* beliefs.

The connections between two nodes y_i and y_j are defined by the spatial context feature d_{ij} (represented in Figure 2) and the weights w_{y_i,y_j}, which encode valid geometric configurations of object classes y_i and y_j.

This way of modeling the underlying state of the image is similar to the one used in [25]. In fact, we rely on their method of greedy forward search to do inference in the CRF and learn the parameters; inference of the physical state from our belief state is described in section 4.5.

4.3 Actions and Observations

Actions are defined by a choice of location i and partial detector c_{kp}, making for $N \sum_{k=1}^{K} P_k$ possibilities.

Upon executing an action, the observation o we get back is the decision of the classifier (or its score, if provided). The observation updates the corresponding node z_{ikp}. We can propagate this update through the CRF with an iteration of belief propagation, or alternatively, we can fully "resolve" our CRF by running a greedy forward search (details in section 4.5). We can do this at every action, every fixed number of actions, or upon taking a special action.

Figure 2: Figure from [25].

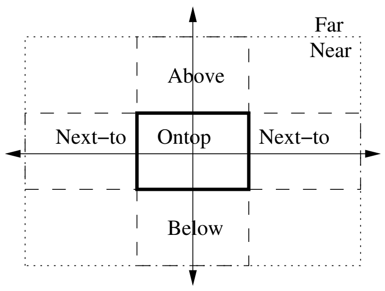


Fig. 3 A visualization of our spatial histogram feature d_{ij}. We consider the location of the center of window j with respect to a coordinate frame defined by window i, denoted by the thickly outlined box. The dashed and dotted rectangles represent regions over which the center of window j are binned. The relative location of j must either be `far` or `near`. For near windows, we consider `above`, `ontop`, `below`, and symmetric `next-to` bins as shown. To allow our model to reproduce the behavior of baseline modules that perform NMS with a criteria of 50% relative overlap, we also include a binary overlap feature. This makes d_{ij} a 7 dimensional sparse binary vector.

We may also want to think of executing actions in batches, such as "evaluate c_{kp} at all locations," or "evaluate all c at a location i." We can represent such sets of actions as a special action.

4.4 Deadline

We express F in units of expected time per action, such that taking action a with time cost τ advances T by τ. We determine τ by averaging the wall clock time of executing a; this is done for all a prior to running the POMDP learner.

Toward our goal is Anytime performance starting at some fixed deadline F (see section 3), our training procedure enforces the deadline stochastically, terminating the agent at any random $T \geq F$.

4.5 Outputting Detections and Rewards

Y in our belief state represents the detections, and Z provides confidences for Y.

Recall that we cannot have detections of two different classes at the same location, and that the PASCAL evaluation will penalize all but one detections of the same class at overlapping locations. For this reason, most detection systems that "assign content to locations" run a post-processing non-maximum suppression step [2]. In a multi-class setting, it is better to resolve such ambiguities with a principled system to model both inter- and intra-class, and short- and long-range interactions.

Our belief state is modeled after such a system, and we simply run a greedy forward search to maximize the energy of our CRF to pick our final detections [25]. The energy of the CRF is given as

$$S(Y,Z) = \sum_{i,j} w^T_{y_i,y_j} d_{ij} + \sum_{i,kp} w_{kp} z_{ikp} \tag{1}$$

In brief, the greedy search starts out with an empty set of detections, and proceeds to pick those y_i that maximize $\Delta S(Y,Z)$. It was shown to be optimal for nearly all images ($\approx 98\%$) in the PASCAL set (and compared against Loopy BP and Tree-ReWeighted BP) [25].

The theoretical explanation for this comes from the work on *submodular functions*. The rough idea is that to maximize functions where adding an element to a large set has less effect than adding an element to a small set, greedy algorithms often have theoretical guarantees of near-optimal performance. The more pairwise connections are non-positive in the CRF, the more submodular it is. **Mention percentage of our learned connections that are non-positive to justify this.**

After every action, $R(s,a)$ is assigned according to the AP of these detections. The program is terminated at time F.

4.6 Some Example Policies

To get a feel for how different detection strategies fit into our framework, we go through some examples, in order of increasing complexity.

Sliding window Here, we take actions that run whole detectors and not their individual classifiers. We featurize the belief state with just two things: the location i and class k of the last action. The policy simply increments these in order, cycling through either locations or classes.

Sliding window, cascaded Now we pick the actual classifier, not just the detector, so let's store i and c_{kp} of the last action. The policy can use the thresholds that the Cascaded DPM detector learned: if observation o is past threshold for c_{kp}, then the next action will be c_{kp+1} at the same location; otherwise, we move on to another location or class.

Saliency-driven Once again we take actions that run whole detectors. Our policy first computes a simple saliency map for the image (for example, as in [13]). The belief state is featurized by the location of the current unexamined (picked less than K times) max in the saliency map and the class k of the last action. The policy is to simply sample locations in order of saliency, running all detectors at a location before proceeding to the next.

Class prior-driven On the training set, we compute K canonical object likelihood maps. We featurize the belief state with the current unexamined (picked less than 1 time) max of each likelihood map. Our policy is to run the detector for the class k at the location i with the largest likelihood among these maps.

Root model score-driven, cascaded The first stage of the cascaded DPM detector is a PCA-reduced root filter. Our policy has two stages. First, it follows a modified **sliding window** policy to run the fast root filter everywhere in the image, for all classes. Then, it follows a modification of the **Class prior-driven** policy, with the class priors given by the root filter scores, and classifier actions picked as in **Sliding window, cascaded**.

Coarse-to-fine, inspired by [6] As above, except in between the stages, we resolve our belief state CRF (which basically does NMS in addition to other things). This is similar to what they do in [6]: run NMS after root filter scoring, then run higher-resolution parts and resolve them again, although we don't do the latter.

Coarse-to-fine, with smart featurization As explained in section 5, we can use low-resolution templates and the lower-scale part of the image pyramid to get estimates on the detections in the higher-scale part of the pyramid.

Our policy first computes only the lower scales of the feature pyramid, and otherwise follows the **Root model score-driven, cascaded** policy with a low-res root filter.

Multi-scale models *As explained in section 5, it has been shown that one should not run high-resolution part-based models to look for small objects in the image pyramid [34]. Their approach learns when to use a low-resolution model and when to use a high-res model, and trains the combined model jointly. Deciding when to use the low-res model could be part of a policy.*

5 Location Discretization and Feature Computation

Location Discretization The fewer possible discrete locations $l_i = (x, y, s)$, with $i = 1 \dots N$, we consider, the smaller our state space and therefore the better our reinforcement learning solution for a given amount of computation. Featurization also takes less time as tolerable coarseness increases. At the same time, we lose power to match ground truth detections.

There are three factors in the discretization process: the stride of the center point of the window (x, y), and the number of octaves and scales per octave of s.

todo: include a figure plotting oracle performance vs. decreasing coarseness of the image pyramid discretization.

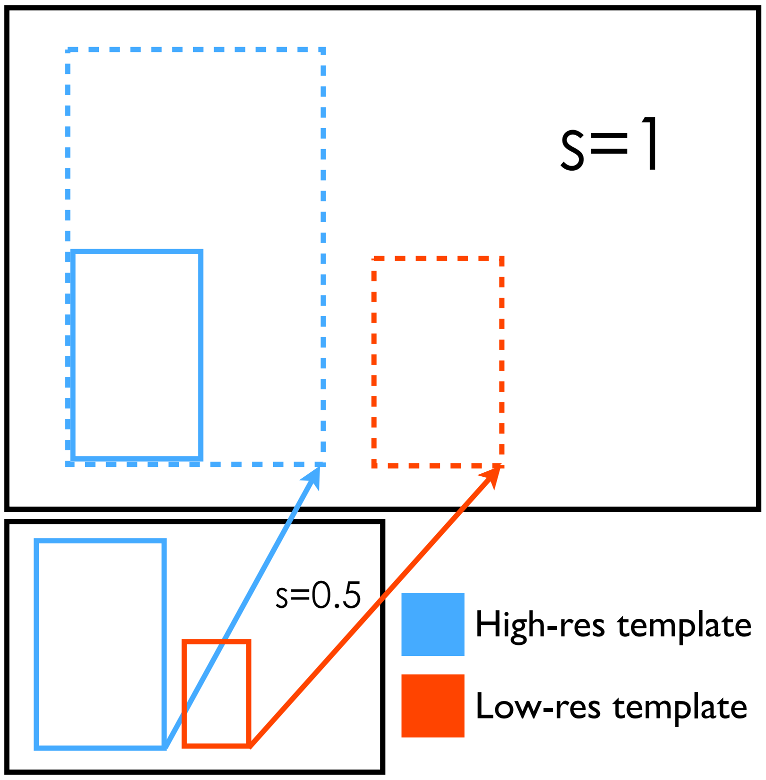
Coarse-to-Fine Evaluation The standard scale-invariant approach to detection with a template is to evaluate a template of fixed size over different scales of the image pyramid. A recent report makes the observation that this approach using high-resolution, part-based models performs worse than scale-variant detection that degrades to using low-resolution, rigid models for detection at small scales of the image [34]. They demonstrate state-of-the-art results on a pedestrian detection benchmark. The authors also note that context is most helpful for small-object detections.

We note also that a low-resolution model used at scale $s/2$ provides an approximation to the output of a high-resolution model at scale s, as illustrated in Figure 3. This may allow us to stagger feature computation, first computing just the small-scale part of the pyramid and running low-res models, and then computing the the large-scale part of the pyramid as time allows.

Feature Computation **todo: how can we include feature computation as part of the action space?**

should look at Piotr Dollar's work on efficient HOG computation [15].

Figure 3: Illustration of the idea of using low-res models to approximate the output of high-res models, allowing us to only compute the image pyramid at small scales.



6 Loose Ideas

6.1 Local context feature

We could try augmenting the DPM model with a local context window, for example by learning another HOG template on a window that is slightly larger than the object window. Has this been done, and why not? It's also what Allie is thinking of doing for shadow (shadow context around object windows).

6.2 Fast multi-class approximation of the right class

If we consider a window and have N detectors, how can we efficiently figure out which of the N detectors have the greatest likelihood of giving a high score to this window?

One idea is to use classifier dimensionality reduction. Let's assume we are using SVM classifiers and classifying vectors of the same dimension. Do we really need to run all of them in their full dimensions, to have a good guess as to which are going to score high? Could we not use dimensionality reduction on all the learned classifiers to come up with a very low-dimensional evaluation that would approximate the full-dimensional score? Then we could only run those classifiers that are past threshold on this low-dimensional approximation.

Another idea is to forget about the specific classifier we are using and just try to partition the feature space into class clusters, roughly. We can use K-means, for example. The cluster(s) that a specific window then falls into determine the more complex classifiers that we will evaluate.

A similar idea is explored in the paper [24]. They use boosting to train a cascade of classifiers to classify all object classes. Then they learn a policy to partition the output of the detectors into class-specific clusters. The policy is a decision tree. The leafs of the decision trees specify the expected class of that window, and they use that to run a class-specific classifier cascade.

7 Results

We want to show two things: 1) given the same amount of time as our baseline systems, we perform at least as well as them; 2) in the AP vs. time evaluation, we do better than all baselines.

Cascaded DPM Baseline [3] We should do as well as it does in the same time (≈0.5 sec), and better than it for shorter times than that.

Coarse-to-fine Baseline [6] Same thing.

We should think of ways to make the comparison fair.

Should we run the multi-class NMS [25] over the detections of the baselines? Since our approach relies on it, it seems that we should; on the other hand, it's part of the special sauce of our approach.

To make sliding window detectors run faster, we can modify the stride or scale quantization of their sliding window. So, if we want to output detections in 0.1 seconds, we can modify the Cascaded DPM code, for example, to cover the image in that long.

We should also evaluate on some kind of video object detection, to showcase the attentional priors. Could be an easy CVPR'12 submission.

References

[1] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL VOC Challenge 2010 Results." http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html, 2010. 2

[2] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models.," *PAMI*, vol. 32, pp. 1627–45, Sept. 2010. 2, 3, 7

[3] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester, "Cascade object detection with deformable part models," in *CVPR*, pp. 2241–2248, IEEE, June 2010. 2, 11

[4] O. Chum and A. Zisserman, "An Exemplar Model for Learning Object Classes," in *CVPR*, pp. 1–8, Ieee, June 2007. 2, 3

[5] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman, "Multiple kernels for object detection," *ICCV*, pp. 606–613, Sept. 2009. 2, 3

[6] M. Pedersoli, A. Vedaldi, and J. Gonzalez, "A Coarse-to-fine approach for fast deformable object detection," in *CVPR*, 2011. 2, 4, 9, 11

[7] S. Vijayanarasimhan and K. Grauman, "Large-Scale Live Active Learning: Training Object Detectors with Crawled Data and Crowds," in *CVPR*, 2011. 2, 3

[8] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in *CVPR*, 2001. 2

[9] L. Bourdev and J. Brandt, “Robust Object Detection via Soft Cascade,” in *CVPR*, pp. 236–243, Ieee, 2005. 2

[10] C. Lampert and M. Blaschko, “A Multiple Kernel Learning Approach to Joint Multi-class Object Detection,” *Lecture Notes in Computer Science: Pattern Recognition*, vol. 5096, 2008. 3

[11] R. Sznitman and B. Jedynak, “Active testing for face detection and localization.,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, pp. 1914–20, Oct. 2010. 3

[12] O. Russakovsky and A. Y. Ng, “A Steiner tree approach to efficient object detection,” in *CVPR*, 2010. 3

[13] B. Alexe, T. Deselaers, and V. Ferrari, “What is an object?,” in *CVPR*, IEEE, 2010. 3, 8

[14] I. Endres and D. Hoiem, “Category Independent Object Proposals,” in *ECCV*, pp. 575–588, 2010. 3

[15] P. Dollár, S. Belongie, and P. Perona, “The fastest pedestrian detector in the west,” in *BMVC*, Citeseer, 2010. 4, 9

[16] N. Butko and J. Movellan, “Optimal scanning for faster object detection,” *CVPR*, pp. 2751–2758, June 2009. 4

[17] J. Vogel and N. de Freitas, “Target-directed attention: Sequential decision-making for gaze planning,” *ICRA*, pp. 2372–2379, May 2008. 4

[18] L. Paletta, G. Fritz, and C. Seifert, “Q-learning of sequential attention for visual object recognition from informative local descriptors,” in *ICML*, (New York, New York, USA), pp. 649–656, ACM Press, 2005. 4

[19] G. Gualdi, A. Prati, and R. Cucchiara, “Multi-stage Sampling with Boosting Cascades for Pedestrian Detection in Images and Videos,” in *ECCV*, pp. 196–209, 2010. 4, 5

[20] S. Vijayanarasimhan and A. Kapoor, “Visual Recognition and Detection Under Bounded Computational Resources,” in *CVPR*, pp. 1006–1013, 2010. 4

[21] A. Torralba, K. P. Murphy, and W. T. Freeman, “Sharing visual features for multiclass and multiview object detection.,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, pp. 854–69, May 2007. 4

[22] X. Fan, “Efficient Multiclass Object Detection by a Hierarchy of Classifiers,” in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, pp. 716–723, Ieee, 2005. 4

[23] N. Razavi, J. Gall, and L. V. Gool, “Scalable Multi-class Object Detection,” in *CVPR*, 2011. 4

[24] R. Isukapalli and A. Elgammal, “Learning Policies for Efficiently Identifying Objects of Many Classes,” *18th International Conference on Pattern Recognition (ICPR’06)*, no. 1, pp. 356–361, 2006. 4, 11

[25] C. Desai, D. Ramanan, and C. Fowlkes, “Discriminative models for multi-class object layout,” in *2009 IEEE 12th International Conference on Computer Vision*, pp. 229–236, Ieee, Sept. 2009. 4, 5, 6, 7, 8, 11

[26] S. Divvala, D. Hoiem, J. Hays, A. Efros, and M. Hebert, “An empirical study of context in object detection,” in *CVPR*, pp. 1271–1278, Ieee, June 2009. 4

[27] A. Torralba, K. P. Murphy, and W. T. Freeman, “Contextual Models for Object Detection Using Boosted Random Fields,” *MIT Computer Science and Artificial Intelligence Laboratory Technical Report*, 2004. 4

[28] L. Itti and C. Koch, “Computational modelling of visual attention.,” *Nature reviews. Neuroscience*, vol. 2, pp. 194–203, Mar. 2001. 4

[29] S. S. Chikkerur, T. Serre, C. Tan, and T. Poggio, “What and where: A Bayesian inference theory of attention,” *Vision Research*, May 2010. 4

[30] T. Judd, K. Ehinger, F. Durand, and A. Torralba, “Learning to predict where humans look,” *2009 IEEE 12th International Conference on Computer Vision*, pp. 2106–2113, Sept. 2009. 4

[31] C. Kanan and G. Cottrell, “Robust Classification of Objects , Faces , and Flowers Using Natural Image Statistics,” in *CVPR*, 2010. 4

[32] J. Denzler and C. M. Brown, “Information Theoretic Sensor Data Selection for Active Object Recognition and State Estimation,” *PAMI*, vol. 24, no. 2, pp. 145–157, 2002. 4

[33] A. Andreopoulos and J. K. Tsotsos, “A theory of active object localization,” in *2009 IEEE 12th International Conference on Computer Vision*, pp. 903–910, Ieee, Sept. 2009. 4

[34] D. Park, D. Ramanan, and C. Fowlkes, “Multiresolution models for object detection,” in *ECCV*, 2010. 9