Active Detection via Adaptive Submodularity

Yuxin Chen[†] YUXIN.CHEN@INF.ETHZ.CH
Hiroaki Shioi[†‡] SHIOI@SPACE.RCAST.U-TOKYO.AC.JP
César Antonio Fuentes Montesinos[†] CESARF@STUDENT.ETHZ.CH
Lian Pin Koh[†] LIAN.KOH@ENV.ETHZ.CH
Serge Wich[*] SERGEWICH1@YAHOO.COM
Andreas Krause[†] KRAUSEA@ETHZ.CH

† ETH Zürich, Zürich, Switzerland
‡ The University of Tokyo, Tokyo, Japan
* Liverpool John Moores University, Liverpool, United Kingdom

Abstract

Efficient detection of multiple object instances is one of the fundamental challenges in computer vision. For certain object categories, even the best automatic systems are yet unable to produce high-quality detection results, and fully manual annotation would be an expensive process. How can detection algorithms interplay with human expert annotators? To make the best use of scarce (human) labeling resources, one needs to decide when to invoke the expert, such that the best possible performance can be achieved while requiring a minimum amount of supervision.

In this paper, we propose a principled approach to active object detection, and show that for a rich class of base detectors algorithms, one can derive a natural sequential decision problem for deciding when to invoke expert supervision. We further show that the objective function satisfies *adaptive submodularity*, which allows us to derive strong performance guarantees for our algorithm. We demonstrate the proposed algorithm on three real-world tasks, including a problem for biodiversity monitoring from micro UAVs in the Sumatra rain forest. Our results show that active detection not only outperforms its passive counterpart; for certain tasks, it also works significantly better than straightforward application of existing active learning techniques. To the best of our knowledge, our approach is the first to rigorously address the active detection problem from both empirical and theoretical perspectives.

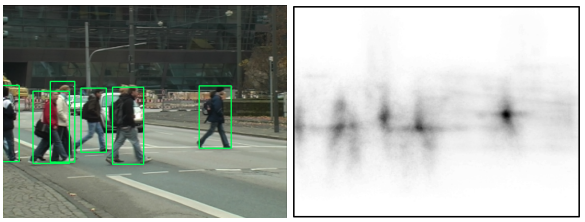
Proceedings of the 31st International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32.

1. Introduction

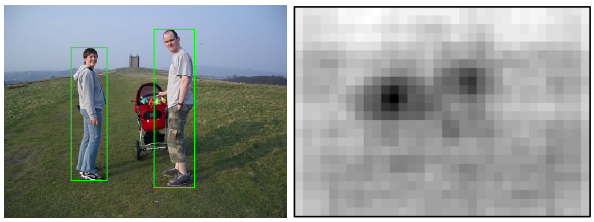
Object detection is one of the fundamental challenges in computer vision. Target objects in real-world images not only exhibit high variance in appearance, but also differ in various views, scales, illumination conditions, and background clutter. While object recognition algorithms have undergone rapid progress, for many practical tasks, a high-quality fully automatic detection system is still beyond our reach. A major problem for automatic object detection is the lack of sufficient training examples, as manual annotations are usually time-consuming and expensive, sometimes even impossible until the task is revealed. For example, consider a biodiversity monitoring task as in Fig. 1(a). In order to obtain accurate and timely data on the orangutan distribution in the surveyed area, ecologists launch micro UAVs, "conservation drones", to take high-quality photographs of orangutan habitat from above treetops. Frequently going through thousands of those photos to look for orangutan nests is an extremely tedious task for experts. On the other hand, an automatic detection system (e.g., the rightmost figure in Fig. 1(a)) tends to produce many false positive detections in the high-clutter background, given limited training samples obtained from the drone missions.

A natural step towards a sustainable and efficient system is to incorporate human supervision during the detection process. In such settings, the automatic system and the human expert collaborate in order to obtain the best performance: first, the system proposes candidate objects to the expert for verification, and then the expert provides feedback in order to guide the system to generate better detections. To make the best use of the scarce labeling resources, one needs to decide when to invoke the expert, or, in other words, in which order to query the candidates, such that the best possible performance could be achieved in exchange for the minimum amount of user supervision.

(a) *Orangutan nests detection for biodiversity monitoring (UAV-forest). (Left) Conservation drone (image courtesy of conservationdrones.org). (Middle) An arial image captured by the conservation drone, with two orangutan nests highlighted. (Right) The response image generated by a base detector.*



(b) *Pedestrian detection (TUD-crossing)*



(c) *Person detection (PASCAL VOC 2008)*

Figure 1. Different object detection tasks and the corresponding response images (in gray scale).

Similar problems (i.e., minimizing the number of user interactions) have been studied extensively as active learning problems in many other contexts, such as text classification (Tong & Koller, 2002), image recognition (Luo et al., 2004). However, comparing with the classical settings, the *active detection* problem studied in this paper is different in the following aspects:

1. In the classical active learning setting, the learner queries information at *training time*, and the goal is to select examples that are informative with respect to the set of classifiers. In other words, it aims to actively produce a classifier that works as well as possible. In contrast, in active detection, we assume that we already have access to a *base detector / classifier* that can produce certain response for target objects (c.f. Fig. 1), and the task is to apply the classifier to the multiple object detection problem. Particularly, we want to use human feedback to actively change the list of proposed detections, and produce as many positive detections as possible. In such cases, the human expert is involved *at test time* rather than training time. Note that one could have used active learning to train the base detectors, which is orthogonal to the active detection process.

2. In our setting, the system only queries objects which it believes to be in the *positive* class, and the human expert either confirms or rejects the proposed detection. As a result, the updates on our base detector often require asymmetric treatments on positive and negative feedbacks.

3. The queries proposed by an active detector are part of the detection process. On one hand, we seek to use the current base detector to detect as many true objects as possible; on the other hand, we would like to correct common mistakes made by the detector, and hope that we can improve the performance by adapting to external feedbacks in the detection process.

Rather than developing novel object recognition algorithms, in this paper, we focus our attention on the study of techniques for intelligently interacting with users. In particular, we propose a general framework for active detection problems, which brings together the quality of manual annotation and the scalability and speed of automatic detection, regardless of what base detectors have been employed. We show how one can, from a given base detector, derive a natural sequential decision problem. Further, its objective function satisfies *adaptive submodularity* (Golovin & Krause, 2011), a natural diminishing returns condition, quantifying how user labels explain the evidence obtained from the base detector. This insight allows us to use highly efficient greedy algorithms, with strong theoretical guarantees. To demonstrate the effectiveness of active detection, we carry out experiments on three different detection tasks using different base object detectors (see Fig. 1), and show that active detection does have substantial advantages over its passive counterpart. In addition, for the orangutan nest detection task, our algorithm significantly outperforms a natural baseline based on existing active learning techniques. To the best of our knowledge, our approach is the first to rigorously address the active detection problem from both empirical and theoretical perspectives.

In summary, our contributions are as follows:

- We propose a general framework for the active detection problem,
- prove theoretical performance guarantees for the proposed algorithm,
- show that different base detectors can be integrated into the framework, and
- demonstrate the effectiveness of our approach on three real-world detection tasks.

2. Related Work

Multiple object detection via submodular optimization Sliding-window based algorithms (Felzenszwalb et al., 2010) and patch based (e.g., Hough transform based) algorithms are two of the most widespread approaches for multiple object detection. These approaches produce responses with peaks at candidate object locations (see Fig. 1). When dealing with overlapping hypotheses, common object detection methods use non-maximum suppression or mode-seeking to locate and distinguish peaks. Such post-processing requires tuning of many parameters and is often fragile, especially when objects are located spatially close to each other. Recently, Barinova et al. (2012) proposes a new probabilistic framework for multiple object detection, with an objective function satisfying submodularity, which can be solved efficiently with a greedy algorithm for submodular maximization (Nemhauser et al., 1978). Hereby, submodularity captures diminishing returns in the detector response at nearby object locations. Our work is inspired by this framework. In contrast to their approach, however, we consider the active detection setting, where detection is interleaved with expert feedback.

Learning object detectors with human in the loop Active learning has been used successfully to reduce labeling cost in classification (Joshi et al., 2012). However, it is more challenging for object detection problems. These approaches start off with few annotated images and then look at a pool of unlabeled examples, and find the ones which would most improve the performance of the classifier once their label has been obtained. Such a procedure has been shown to significantly reduce the number of required labels (Abramson & Freund, 2004; Kapoor et al., 2007; Bietti, 2012), and even work well in large scale (Vijayanarasimhan & Grauman, 2011) and in a distributed pattern (Vijayanarasimhan et al., 2010). However, in these works, active learning has only been applied at training time to produce a good base detector. To re-iterate, we consider the complementary setting, of taking any given base detector, and applying it in an active manner at test time, i.e., interleaving automatic detection with expert feedback. Moreover, many existing algorithms require retraining the model from scratch on new labels, whereas we choose to gradually update the base detector on new observations, which potentially could be much cheaper.

Human feedback has been used in different levels: e.g., image-level (Vijayanarasimhan & Grauman, 2008), object and attribute levels (Kovashka et al., 2011; Shrivastava et al., 2012), and part-level annotations (Wah et al., 2011). Parkash & Parikh (2012) consider attribute feedback (at training time), but on negative labels, the human expert also tries to communicate an explanation for why the learner's belief is wrong, and the learner can then propagate the feedback to many unlabeled images. Our work is similar in that we also treat positive and negative feedbacks differently, but we only require object level feedback (at test time) to update the prediction model.

Object recognition with test-time feedback Branson et al. (2010) and Wah et al. (2011) study fine-grained classification problems, where the goal is to recognize bird species, with the help from an expert answering actively selected questions pertaining to visual attributes of the bird. Similarly, Branson et al. (2011) focus on interactive labeling, where users can adjust incorrect labels (e.g., the expert can drag misplaced parts to correct locations). In these works, they consider posing multiple queries on part attributes for each test image, and allow symmetric updates for both positive and negative labels. In contrast, we focus on a different problem setting: active detection of multiple object instances (with asymmetric updates upon different labels), where user only provides a single "yes" or "no" feedback on each proposed candidate.

3. Active Detection as a Coverage Problem

Motivating Example: Hough-transform based method Hough-transform based detection algorithms work by transforming the input image into a new representation in a domain called the Hough space (Hough, 1959; Ballard, 1981). Each point in the Hough space corresponds to a hypothesis of existence of an object instance with some particular configuration. The Hough image is built by aggregating the contributions of the individual voting elements, taken from the image or some appropriate sets of features of it. The detections will then be identified as peaks in the Hough image, with the height of the peak as an indicator of the confidence in the detection. As an example, to detect lines in an image, one can search through the peaks in the 2-d Hough space (with each axis corresponding to one parameter of the line function), and find a subset of line parameters that have the highest accumulated votes. Similarly, to detect natural objects, one needs to create individual voting elements that vote for a certain configuration of the whole object. See Fig. 1(b) for an illustration.

More generally: Votes and Hypotheses Suppose we are given a finite set $\mathcal{H}$ of hypotheses $h_1, \ldots, h_n$, where each hypothesis $h_i \in \mathcal{H}$ represents a possible configuration of the target object at location $\mathbf{x}_i$. We use $Y_1, \ldots, Y_n \in \mathcal{O} = \{+1, -1\}$ to denote the (initially unknown) labels of the hypotheses, such that $Y_i = +1$ if hypothesis h_i is true (i.e., there exists an object at $\mathbf{x}_i$), and $Y_i = -1$ otherwise. We use $\mathbf{Y}_{\mathcal{H}} = \{Y_1, \ldots, Y_n\}$ to refer to the collection of all variables. Whenever a hypothesis h_i is selected, the corresponding variable Y_i is revealed as y_i. Similarly, if we select a set of hypotheses $\mathcal{A}$, the corresponding observations are represented as $\mathbf{y}_{\mathcal{A}} \in 2^{\mathcal{H} \times \mathcal{O}}$.

We further assume a finite set of evidence $\mathcal{V}$. Each item $v_i \in \mathcal{V}$ corresponds to a voting element that can cast votes for a set of hypotheses. A base object detector proposes a voting scheme that connects the hypotheses set $\mathcal{H}$ and the evidence set $\mathcal{V}$. The interaction between hypotheses and voting elements can be formally represented as a bipartite graph $\mathcal{G}(\mathcal{V}, \mathcal{H}, \mathcal{E})$; each edge $(v, h) \in \mathcal{E}$ is assigned with a score (e.g., confidence, probability estimation given by the base object detector) with which v votes for hypothesis h. We will give concrete examples in Section 5.

Active Detection as a Sequential Decision Problem We consider a sequential strategy, where the detector proposes a hypothesis $h_i \in \mathcal{H}$, and receives a label y_i from the expert. Whenever a label is revealed, we update the underlying bipartite graph, which represents the current state of the base detector. In particular, we perform the updates by only reducing the weights associated with the voting elements (i.e., covering the edges in $\mathcal{G}$), as observations will keep explaining the votes proposed by the base detector. Our goal, therefore, is to propose a strategy that can cover the entire set of edges as soon as possible.

4. The Active Detection Framework

We begin with the case where the votes generated by the base detector only have binary values, and then generalize to the setting with real-valued votes. In Section 4.3, we provide an efficient greedy solution to the active detection problem, and present our main results.

4.1. Binary Votes Setting

We first study a simple case, where the voting elements can only cast binary votes (i.e., 0/1) for the presence of an object with configuration/location $\mathbf{x}$ encoded by some hypotheses $h \in \mathcal{H}$.

Suppose the active learner proposes a hypothetical detection h_+, and receives a positive label from an external expert. Since a voting element has equal confidence for all its supporting hypotheses, the true hypothesis then fully explains the voting elements $v \in \mathcal{V}$ that voted for h_+, thereby "covering" all votes associated with those voting elements. We refer to the amount of edges covered by selecting h_+ with positive feedback as the *positive coverage* of h_+.

The perhaps more interesting case is when the active detector makes a false prediction. Let the *negative coverage* be the reduction of edge weights in $\mathcal{G}$ incurred by a false detection $h_- \in \mathcal{H}$. The construction of negative coverage is akin to that of positive coverage, but with one substantial difference: while in the positive case we cover the edges which are *neighbors* of the edges that directly vote for h_+ (i.e., stemming from the same voting elements), in the negative case we will reduce the weight of all edges that are *similar* to the ones pointing to the false hypothesis h_-. Concretely, we assume that the votes generated by the base detector are associated with some features, and thus can be clustered accordingly. The clustering associated with the base detector is denoted by means of a function $c : \mathcal{V} \times \mathcal{H} \to \mathbb{Z}^+$, which maps an edge $(v, h) \in \mathcal{E}$ to its cluster index. A false detection thereby "explains away" (covers) similar votes that share the same clusters with the potentially false vote(s). See Fig. 2 for an illustration.

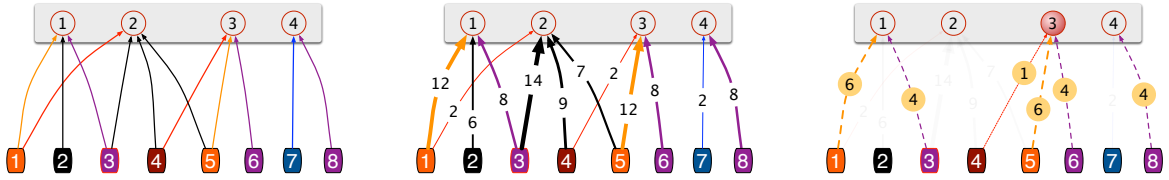
Formally, the fraction of an edge $(v, h) \in \mathcal{E}$ covered due to negative observations, could be modeled as a monotone increasing function $g : \mathcal{E} \times 2^{\mathcal{H} \times \mathcal{O}} \to [0, 1]$. In particular, we express the coverage as a function q of how frequent similar edges have been observed to vote for false hypotheses: $g(v, h, \mathbf{y}_{\mathcal{A}}) = q(n_{\texttt{neg}}(v, h, \mathbf{y}_{\mathcal{A}}))$, where $n_{\texttt{neg}}(v, h, \mathbf{y}_{\mathcal{A}}) \equiv |\{(h', -1) \in \mathbf{y}_{\mathcal{A}} : \exists v', c(v', h') = c(v, h)\}|$ is the number of false hypotheses that are being voted for by any edge in the same cluster as (v, h). In general, we want such a function to be concave within range [0,1], i.e., the edge should be largely covered even when $n_{\texttt{neg}}$ is small, and reach full coverage when $n_{\texttt{neg}}$ approaches infinity. An extreme choice would be $q(n) = \min(n, 1)$: the edge is fully covered as soon as it is in the same cluster of a vote for a negative hypothesis. A less aggressive choice of the concave function, which we adopt in our experiments, is:

$$q(n) = 1 - \gamma^n. \tag{4.1}$$

The negative discount factor γ controls the speed with which the weights will be discounted. If $\gamma = 0$, all the edges in the cluster c will be fully discounted once one of them votes for a negative hypothesis; if $\gamma = 1$, the edges will never be discounted.

Now we are ready to construct the *coverage function* $f^{(1)}_{v,h} : 2^{\mathcal{H} \times \mathcal{O}} \to \mathbb{R}_{\geq 0}$ for any edge $(v, h) \in \mathcal{E}$, in the binary votes setting. Given a set of hypotheses $\mathcal{A} \subseteq \mathcal{H}$, and corresponding observations $\mathbf{y}_{\mathcal{A}} \subseteq \mathcal{H} \times \mathcal{O}$, the amount by which a given edge (v, h) is covered is defined as

$$f^{(1)}_{v,h}(\mathbf{y}_{\mathcal{A}}) = \begin{cases} 1, & \text{if } \exists h' : (h', +) \in \mathbf{y}_{\mathcal{A}} \wedge (v, h') \in \mathcal{E}; \\ g(v, h, \mathbf{y}_{\mathcal{A}}), & \text{otherwise.} \end{cases} \tag{4.2}$$



(a) *Binary votes* (b) *Votes with real values* (c) *(Negative) edge coverage*

Figure 2. The voting scheme proposed by a base object detector as a bipartite graph. Edges are drawn between hypotheses (upper nodes) and voting elements (lower nodes). "Similar" edges share the same color. Fig. 2(a) and 2(b) show two toy examples with binary and real-value votes, respectively. For example, in Fig. 2(a), if hypothesis 2 is true, then all the edges associated with voting elements 1, 3, 4, 5 will be covered, as those voting elements are "explained away" from other hypotheses. Fig. 2(c) illustrates how we should update Fig. 2(b) from a negative feedback: if hypothesis 3 in Fig. 2(b) is false, then all the "similar" edges as highlighted are (partially) covered.

4.2. The General Case with Real-valued Votes

The previous approach is limited in that it only allows us to describe the support given by a voting element to a hypothesis as a binary relation. In practical settings, we would like to take the strength of confidence into account, i.e., each edge (v, h) is associated with a weight $w_{vh} \in \mathbb{R}_{\geq 0}$. For this more general scenario, we need to redefine "coverage", by allowing edges to be partially covered. Following the previous example, when an edge (v, h) is covered due to positive observation, it will also cover its neighbors in a magnitude that is at most its weight w_{vh}. Since we do not allow negative weights, if a neighbor edge (v, h') has a weight $w_{vh'} < w_{vh}$, then it is fully covered. Thus, an edge (v, h) covers another edge (v, h') in a magnitude given by $\min(w_{vh}, w_{vh'})$.

Taking negative coverage into account, the coverage function for an edge is defined as:

$$f_{v,h}(\mathbf{y}_\mathcal{A}) = g(v, h, \mathbf{y}_\mathcal{A}) \cdot w_{vh} + \min\left\{ \max_{(h',+1)\in\mathbf{y}_\mathcal{A}} w_{vh'},\ (1 - g(v, h, \mathbf{y}_\mathcal{A})) \cdot w_{vh} \right\} \quad (4.3)$$

We can interpret the first term on the RHS as the fraction of weight covered due to negative observations, and the second therm as the fraction of remaining weight (i.e., after negative discount) covered due to positive observations. Note that $w_{vh'}$ does not have a discount factor, since we know that the edge (v, h') represents the vote for a positive hypothesis, and thus it should be fully covered.

Connection with the binary votes setting. We can see that the coverage function with binary votes (Eq. 4.2) is a special case of the general coverage function (Eq. 4.3), when all non-zero weights are set to 1: Assume the edge (v, h) exists, i.e., $w_{vh} = 1$. If the maximum among the weights $w_{vh'}$ of the first term is 0, then the first term vanishes and we are left with $g(v, h, \mathbf{y}_\mathcal{A})$. Note that all the $w_{vh'}$ being 0 is equivalent to the second case of Equation 4.2. The only alternative is if $\max_{(h',+1)\in\mathbf{y}_\mathcal{A}} w_{vh'} = 1$. Since $1 - g \leq 1$, we have $f_{v,h}(\mathbf{y}_\mathcal{A}) = (1 - g) + g = 1$.

The objective function. Finally, we can define the objective function $F : 2^{\mathcal{H}\times\mathcal{O}} \to \mathbb{R}_{\geq 0}$ for the active detection problem, by summing up weights covered from all the edges in $\mathcal{E}$:

$$F(\mathbf{y}_\mathcal{A}) = \sum_{(v,h)\in\mathcal{E}} f_{v,h}(\mathbf{y}_\mathcal{A}) \quad (4.4)$$

The goal of active detection, therefore, is to adaptively select a minimum subset of hypotheses, such that the edges in the underlying bipartite graph can be fully covered.

4.3. Active Detection: A Greedy Solution

In this section, we show that the active detection problem defined in the previous section is an *adaptive submodular* optimization problem, and thus can be efficiently solved using a greedy algorithm. First, we show that the objective function (Eq. 4.4) satisfies submodularity:

Lemma 1. *F is monotone submodular.*

Formally, a function $f : 2^{\mathcal{H}\times\mathcal{O}} \to \mathbb{R}_{\geq 0}$ is submodular, if for all $(h, y_h) \in \mathcal{H} \times \mathcal{O}$ and $\mathbf{y}_\mathcal{A} \subseteq \mathbf{y}_\mathcal{B} \subseteq \mathcal{H} \times \mathcal{O}$, it holds that $f(\{(h, y_h)\} \cup \mathbf{y}_\mathcal{A}) - f(\mathbf{y}_\mathcal{A}) \geq f(\{(h, y_h)\} \cup \mathbf{y}_\mathcal{B}) - f(\mathbf{y}_\mathcal{B})$. In other words, adding a label helps more if we have observed few labels so far. The key idea of the proof is that we can decompose a voting element into many voting elements, each casts equal votes to its favorable hypotheses. Then we just need to prove F in the new evidence space to be submodular, which is straightforward. We defer the proof details to the supplementary material.

In active detection, since we have no access to the label of a hypothesis in advance, we are not able to select hypothesis-label *pairs* for each iteration. Instead, we consider the *conditional expected marginal gain* of a hypothesis h (considering *any* possible label):

$$\Delta_F(h \mid \mathbf{y}_\mathcal{A}) = \mathbb{E}_{\mathbf{y}_\mathcal{H}}[F(\mathbf{y}_\mathcal{A} \cup \{(h, y_h)\}) - F(\mathbf{y}_\mathcal{A}) \mid \mathbf{y}_\mathcal{A}]\,. \quad (4.5)$$

Function F together with prior distribution $P(\mathbf{Y}_\mathcal{H})$ is called adaptive submodular (Golovin & Krause, 2011), if, whenever $\mathbf{y}_\mathcal{A} \subseteq \mathbf{y}_\mathcal{B} \subseteq \mathcal{H} \times \mathcal{O}$, and $P(\mathbf{Y}_\mathcal{H}) > 0$, we have $\Delta_F(h \mid \mathbf{y}_\mathcal{A}) > \Delta_F(h \mid \mathbf{y}_\mathcal{B})$. Adaptive submodularity

Algorithm 1 The active detection algorithm

Input: Bipartite graph $\mathcal{G}(\mathcal{V}, \mathcal{H}, \mathcal{E})$, prior $P(\mathbf{Y}_\mathcal{H})$, discount factor γ, # of detections N
Output: Detections (with associated labels) $\mathbf{y}_\mathcal{A}$
$\mathcal{A} \leftarrow \emptyset$, $\mathbf{y}_\mathcal{A} \leftarrow \emptyset$
for $i = 1$ **to** N **do**
  for all h **in** $\mathcal{H}$ **do** %
  compute positive and negative coverage:
    $\Delta_+(h) \leftarrow \sum_{v \in \mathcal{V}} \sum_{h' \in \mathcal{H}} \min\{w_{vh},\ w_{vh'}\}$
    $\Delta_-(h) \leftarrow \sum_{c(v',h')=c(v,h)} \{\gamma \cdot w_{v'h'}\}$
  end for
  $h^* \leftarrow \arg\max_h \{P(y_h = +1)\Delta_+(h) + P(y_h = -1)\Delta_-(h)\}$
  Observe y_{h^*}
  for all edges (v, h) **in** $\mathcal{E}$ **do** %
  perform positive and negative updates:
    if $y^* = +1$ **then**
      $w_{vh} \leftarrow \max\{w_{vh} - w_{vh^*},\ 0\}$
    else if $c(v, h) = c(v, h^*)$ **then**
      $w_{vh} \leftarrow \gamma w_{vh}$
    end if
  end for
  $\mathcal{A} \leftarrow \mathcal{A} \cup \{h^*\}$, $\mathbf{y}_\mathcal{A} \leftarrow \mathbf{y}_\mathcal{A} \cup \{(h^*, y_{h^*})\}$
end for

characterizes a natural diminishing returns property: the gain of a new item, in expectation over its unknown label, can never increase as we gather more information.

Lemma 2. *F is adaptive submodular w.r.t. $P(Y_1, \ldots, Y_n)$ as long as $Y_1, \ldots, Y_n$ are independent.*

Proof. With a factorial distribution over the outcomes, the adaptive submodularity of F follows immediately from Lm. 1 and Thm. 6.1 of Golovin & Krause (2011). □

With the objective function defined in Section 4.2, we can associate the following greedy algorithm: It starts with the empty set, and at each iteration adds to the current set $\mathcal{A}$ the hypothesis h which maximizes the marginal improvement (Eq. 4.5). Once the label of h is observed, we update the bipartite graph $\mathcal{G}$ with the remaining edges that have not yet been explained by the current observations $\mathbf{y}_\mathcal{A}$. Algorithm 1 provides the details of the greedy algorithm. A major benefit of adaptive submodularity is that we can use a technique called *lazy evaluations* to dramatically speed up the selection process (Golovin & Krause, 2011). A further benefit is the following performance guarantee, which we obtain following the analysis in Golovin & Krause (2011).

Corollary 3. *Suppose $F : 2^{\mathcal{H} \times \mathcal{O}} \rightarrow \mathbb{R}_{\geq 0}$ is defined as Equation 4.4. Fix any value $Q > 0$ and $\beta > 0$, and let OPT_{wc} be worst-case cost of an optimal policy that achieves a maximum coverage value of Q for any realization of the variables $\mathbf{Y}_\mathcal{H}$. Let C_{greedy} be the cost of Algorithm 1 using a factorial prior on variables $Y_1, \ldots, Y_n$, until it achieves expected value $Q - \beta$. Then,*

$$C_{greedy} \leq \mathsf{OPT}_{wc} \left(\ln \left(\frac{Q}{\beta} \right) + 1 \right).$$

Moreover, it holds that under the algorithm's prior: $P(f(\mathbf{y}_\mathcal{A}) \geq Q) \geq 1 - \beta$.

Note that the above result provides guarantees even for worst-case realization of $\mathbf{Y}_\mathcal{H}$ (i.e., without assumptions on $P(\mathbf{Y}_\mathcal{H})$), as long as our algorithm uses any factorial prior. Further note that if we choose, in the extreme case, $\beta = \min_{\mathbf{Y}_\mathcal{H}} P(\mathbf{Y}_\mathcal{H})$, we actually guarantee that the algorithm achieves full coverage ($f(\mathbf{y}_\mathcal{A}) \geq Q$) for *all* realizations of $\mathbf{Y}_\mathcal{H}$. If we do not have a strong prior, we can obtain the strongest guarantees if we choose a distribution "as uniform as possible" (i.e., maximizes $\min_{\mathbf{Y}_\mathcal{H}} P(\mathbf{Y}_\mathcal{H})$), while still guaranteeing adaptive submodularity.

5. Experiments

In this section, we empirically evaluate our active detection approach on three (substantially different) data sets: an orangutan nest detection task for biodiversity monitoring, a pedestrian tracking task in a video sequence, and a standard object detection task for the PASCAL VOC Challenge. For each data set we employ different base detector that is most tailored for the task. Our emphasis is on comparing the active detection algorithm with classical passive detection algorithms (and active detection baseline, if applicable), as well as empirically quantifying the improvement by the active detection framework over the base detectors.

Orangutan Nest Detection on UAV-recorded Forest Images The first application is an interactive orangutan nests detection system for biodiversity monitoring. To estimate the distribution of critically endangered Sumatran orangutans (*Pongo abelii*), ecologists deploy conservation drones above orangutan habitat in surveyed areas, so that they can obtain timely and high-quality photographs of orangutan nests high in the tree canopies (Koh & Wich, 2012). Our test set contains 37 full-resolution (4000×3000 pixels) images from two separate drone missions launched in September 2012, in Sumatra, Indonesia. Each of the target images contains at least one orangutan nest, and there are a total number of 45 nests in the data set, with a minimum size of 19×19 pixels. Selected examples of the nest and non-nest image patches are shown in Fig. 3.

As we can see from the examples, the positive class has high intra-class variation. For efficiency considerations, we reduce the resolution of the original images by half. We then extract all 45 examples of orangutan nests of size 9×9 pixels, as well as 148 background image patches, as the labeled set. Each training example is represented as

a 9-d vector which consists of statistics (mean, maximum and minimum) of three color channels in a patch. Based on these features, we train a linear discriminant classifier (LDA) in order to classify orangutan nests vs. background.

Figure 3. Positive (upper) and negative (lower) examples of orangutan nests in the UAV-recorded forest data set.

The base detector we employ is a sliding-window based system. As we do not have sufficient (positive) training data, we use all the labeled images other than those in the current test image as training set. At runtime, each image patch located by the current sliding window (of size 9×9) is evaluated with a pre-trained classifier, and used as a voting element that casts equal votes to its surrounding area (i.e., 9×9 pixels). The confidence of votes from theses voting windows are determined by their distances to the classifier's decision boundary; positive windows that are further away from the decision boundary have higher confidence when voting for a nest hypothesis.

To cluster similar voting elements, we apply k-means algorithm on the set of voting windows. Moreover, as negative detections often occur adjacently (e.g., branches are usually connected), we also use a *local clustering* algorithm (i.e., segmenting nearby regions), to avoid overwhelming false detections. The precision-recall curve for active detection is demonstrated by the red line in Fig. 4(a).

Our first baseline is the "passive" version of Alg. 1, where the algorithm assumes all detections to be "true", and thus only performs positive updates. The only difference between Alg. 1 and the passive baseline is that for active detection, we actively update the order of the sequence, while for the passive baseline we do not (note that the passive approach also needs expert to verify the detections, only that it happens after all detections are made). We can see from Fig. 4(a) that, at 80% recall, active detection ($\gamma = 0.5$) obtains almost twice the precision (0.27 vs. 0.15) as the passive approach. As another baseline, we compare with an active learning heuristic, where the base classifier is retrained after each query. More specifically, at each iteration, the baseline active detector generates a response image by applying the new classifier, and removes the surrounding areas of previous candidates by non-maximal suppression. The next candidate is then located through mode-seeking. As shown in Fig. 4(a), although the active baseline (blue curve) comes with no guarantees, it still outperforms the passive approach due to extra feedbacks, but generally performs worse than Alg. 1.

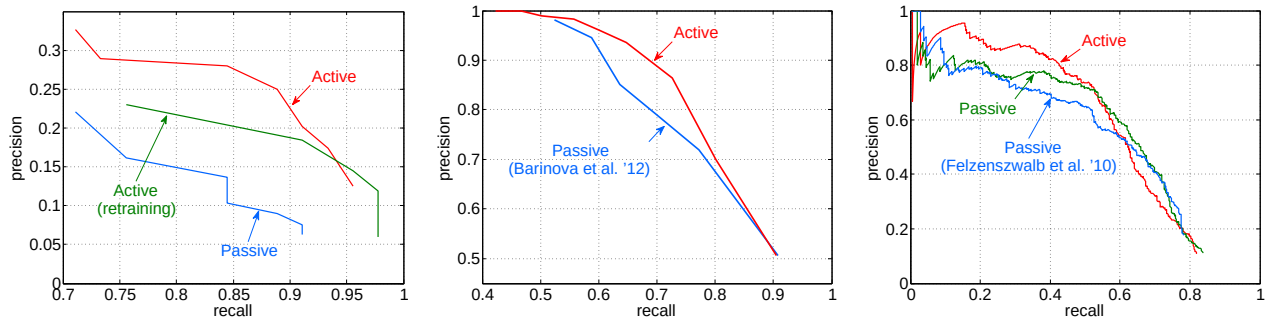
Pedestrian Detection on TUD-crossing Image Sequence Hough-based approaches offer seamless integration with the active detection framework. To demonstrate how user supervision can help such systems, we apply Alg. 1 to the TUD-crossing sequence, based on the *Hough Forest* detector proposed in Gall & Lempitsky (2009). We use a discount factor $\gamma = 0.01$ to penalize votes that are "similar" with any of the incorrect votes. Here votes are considered "similar" if they are (1) from similar image patches (i.e., sharing the same leaf in Hough forest), and (2) pointing to locations that have the same offset to the voting elements. We also use "local clusters" to update the bipartite graph when observing a false hypothesis, similar as the case for nest detection: edges that share the same voting element are considered within the same local cluster, and thus will be discounted if any of them points to a false hypothesis.

Since the background clutter does not change much across frames, for active detection we choose to share the cluster updates through the entire video sequence, rather than discard the information acquired from user feedback and start from scratch (i.e., reset the negative count for each cluster) for each new frame. As baseline, we compare active detection with the state-of-the-art passive detection results on this data set, which is given by Barinova et al. (2012). We find that training a Hough forest detector is very expensive (e.g., it takes > 1 hour to train a forest with 15 trees), making it highly inefficient to frequently retrain the model. Therefore, we skip the active baseline for this task.

For evaluation of both algorithms, we apply the Hungarian algorithm (Kuhn, 1955) to match the set of detections with the ground truth annotations, based on the Jaccard similarity ($\leq 40\%$ are considered as false detections) between bounding boxes[1]. We test the candidate algorithms on 41 frames of the TUD-crossing sequence (by sampling every 5th frame of the full video sequence) in the single scale scenario, and show the results in Fig. 4(b). The curves are generated by varying the stopping threshold on the marginal gain of new hypotheses. We limited the maximum number of detections to be 10 for both systems (given there are at most 8 objects per frame) in order to have a fair comparison. As can be seen, with user supervision, our framework considerably outperforms the baseline detection algorithm.

Object detection on PASCAL VOC Data Set The third data set differs from the previous two in the sense that it contains object classes that exhibit much richer structural features (e.g., the "person" class includes examples of a high variability of poses and shapes). The state-of-the-art results for this dataset are obtained by the sliding-window

[1]Greedy matching is problematic for data sets that exhibit sufficient overlap between objects, because once a detection is matched to an object, it cannot switch to another even if the second is a better matching.



(a) *UAV-forest images* (b) *TUD-crossing sequence* (c) *PASCAL VOC*

Figure 4. Performance of the active detection on three different tasks

based, multi-scale, deformable parts model (MDPM) of Felzenszwalb et al. (2010). To convey the idea that our framework can incorporate different base detectors, we build our bipartite graph upon an earlier release (*voc-release3*) of their system, as it already incorporates most of the important innovations of the MDPM, without extra expensive components (e.g., grammar models as in Girshick et al. (2011)) that are designed specifically for certain tasks.

In MDPM, each category is modeled by a "root" filter that describes the shape of the object, and a fixed number of part filters that describe important sub-areas of the object at a higher resolution. For multi-scale detection, we keep a *feature pyramid* that consists image *cells* (of size 8×8, represented by a 31-d HOG descriptor (Dalal & Triggs, 2005)) from a pile of rescaled versions of the image. A hypothesis z is then characterized by a triplet (x, y, s) corresponding to the location and scale of an object, and is scored jointly by both root filter and associated parts filters.

To build a bipartite graph, we assume that voting elements correspond to image patches (i.e., cells in the feature pyramid), and will cast equal votes for a hypothesis h given that they are inside its associated window. The total sum of votes h receives from the voting cells amount to the score given by the underlying MDPM. To handle the deformable parts, we further assume two types of hypotheses: "root hypotheses" that represent the existence of an object, and "part hypotheses" as intermediate nodes in the bipartite graph, that can be voted by (part) cells. Each hypothesis node in the bipartite graph will eventually receive (direct) votes from the root cells, as well as (indirect) votes from the part cells, that are weighted by the deformation coefficient (Felzenszwalb et al., 2010) of the part window.

Edge similarity is measured based on two sets of features: the filter type of the window associated with h, and the HOG descriptor of the voting cell v. To construct clusters, we first group the windows by filter type, and then employ a hierarchical clustering method to retrieve similar edges (i.e., small cosine distance between HOG descriptors).

Fig. 4(c) shows our results on the *Person* category of the *VOC2008* data set (4133 test images). Each detector makes 16 detections per image. Unlike the nest detection task, retraining a deformable parts model after each detection is a prohibitive task (it would have taken weeks to retrain $\approx$ 66K MDPMs). Without a better choice for active baselines, we show how human feedbacks can be used to improve the base detector. The red curve shows the performance of our active detector (AUC 0.566), which only uses root filters, yet it already outperforms the baseline system (Felzenszwalb et al. (2010), AUC 0.516) that utilizes both root and parts filters. Note that in principle we can incorporate feedback of part filters as well. We also find that the passive approach under our framework (i.e., "active detection" assuming all detections are true) also considerably outperforms the baseline (AUC 0.544). One possible reason is that, when identifying multiple objects, our framework does not suffer from problems caused by the non-maximum suppression approach, and thus has better recall.

6. Conclusion

In this paper, we propose an active detection framework that enables turning existing base detectors into automatic systems for intelligently interacting with users. Our approach reduces active object detection to a sequential edge covering optimization problem. We show that the objective function satisfies adaptive submodularity, allowing us to use efficient greedy algorithms, with strong theoretical performance guarantees. We demonstrate the effectiveness of the active detection algorithm on three different real-world object detection tasks, and show that active detection not only works for various base detectors, but also provides substantial advantages over its passive counterpart.

Acknowledgments. This research was supported in part by SNSF grant 200021_137971, DARPA MSEE FA8650-11-1-7156, ERC StG 307036 and a Microsoft Research Faculty Fellowship. LPK and SW thank the Leuser International Foundation and Sumatran Orangutan Conservation Program for their support.

References

Abramson, Y. and Freund, Y. Active learning for visual object recognition. Technical report, UCSD, 2004.

Ballard, D.H. Generalizing the hough transform to detect arbitrary shapes. *Pattern recognition*, 1981.

Barinova, O., Lempitsky, V., and Kholi, P. On detection of multiple object instances using hough transforms. *PAMI*, 2012.

Bietti, A. Active learning for object detection on satellite images. Technical report, Caltech, 2012.

Branson, S., Wah, C., Schroff, F., Babenko, B., Welinder, P., Perona, P., and Belongie, S. Visual recognition with humans in the loop. In *ECCV*, pp. 438–451, 2010.

Branson, S., Perona, P., and Belongie, S. Strong supervision from weak annotation: Interactive training of deformable part models. In *ICCV*, pp. 1832–1839, 2011.

Comaniciu, D. and Meer, P. Mean shift: A robust approach toward feature space analysis. In *PAMI*, 2002.

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

Felzenszwalb, P.F., Girshick, R.B., McAllester, D., and Ramanan, D. Object detection with discriminatively trained part-based models. *PAMI*, 2010.

Gall, J. and Lempitsky, V. S. Class-specific hough forests for object detection. In *CVPR*, 2009.

Girshick, R., Felzenszwalb, P., and McAllester, D. Object detection with grammar models. *IEEE TPAMI*, 2011.

Golovin, D. and Krause, A. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *JAIR*, 2011.

Hough, P.V.C. Machine Analysis of Bubble Chamber Pictures. In *International Conference on High Energy Accelerators and Instrumentation*, 1959.

Joshi, A.J., Porikli, F., and Papanikolopoulos, N.P. Scalable active learning for multiclass image classification. *PAMI*, 2012.

Kapoor, A., Grauman, K., Urtasun, R., and Darrell, T. Active learning with gaussian processes for object categorization. In *ICCV*, 2007.

Koh, L.P. and Wich, S.A. Dawn of drone ecology: low-cost autonomous aerial vehicles for conservation. *Trop Conserv Sci*, 2012.

Kovashka, A., Vijayanarasimhan, S., and Grauman, K. Actively selecting annotations among objects and attributes. *ICCV*, 0:1403–1410, 2011.

Kuhn, Harold W. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

Luo, T., Kramer, K., Goldgof, D.B., Hall, L.O., Samson, S., Remsen, A., and Hopkins, T. Active learning to recognize multiple types of plankton. In *ICPR*, 2004.

Nemhauser, G.L., Wolsey, L.A., and Fisher, M.L. An analysis of approximations for maximizing submodular set functions–i. *Mathematical Programming*, 1978.

Parkash, A. and Parikh, D. Attributes for classifier feedback. In *ECCV*, pp. 354–368, 2012.

Shrivastava, A., Singh, S., and Gupta, A. Constrained semi-supervised learning using attributes and comparative attributes. In *ECCV*, pp. 369–383, 2012.

Tong, S. and Koller, D. Support vector machine active learning with applications to text classification. *JMLR*, 2002.

Vijayanarasimhan, S. and Grauman, K. Multi-level active prediction of useful image annotations for recognition. In *NIPS*, pp. 1705–1712, 2008.

Vijayanarasimhan, S. and Grauman, K. Large-scale live active learning: Training object detectors with crawled data and crowds. In *CVPR*, 2011.

Vijayanarasimhan, S., Jain, P., and Grauman, K. Far-sighted active learning on a budget for image and video recognition. In *CVPR*, pp. 3035–3042, 2010.

Wah, C., Branson, S., Perona, P., and Belongie, S. Multiclass recognition and part localization with humans in the loop. In *ICCV*, pp. 2524–2531, 2011.

A. Proofs

Proof of Lemma 1 . We first decompose each voting element into a set of voting elements, each carrying equal weights for all of its outgoing edges. Let the new voting elements set be $\mathcal{V}'$. Then for all $v \in \mathcal{V}'$ and $h_1, h_2 \in \mathcal{H}$, it holds that

$$(w_{vh_1} > 0) \wedge (w_{vh_2} > 0) \Rightarrow w_{vh_1} = w_{vh_2}.$$

We then show that the function $f_{v,h}$ is monotone submodular for edges in the new bipartite graph.

Let $\Delta_f((h_s, y_s) \mid \mathbf{y}_\mathcal{A}) = f_{v,h}(\mathbf{y}_\mathcal{A} \cup \{(h_s, y_s)\}) - f_{v,h}(\mathbf{y}_\mathcal{A})$ be the marginal gain of $f_{v,h}$ over set $\mathbf{y}_\mathcal{A}$ by selecting (h_s, y_s). We need to show that for each $\mathbf{y}_\mathcal{A} \subseteq \mathbf{y}_\mathcal{B} \subseteq \mathcal{H} \times \mathcal{O}$, and $(h_s, y_s) \in \mathcal{H} \times \mathcal{O}$ it holds that

$$\Delta_f((h_s, y_s) \mid \mathbf{y}_\mathcal{A}) \geq \Delta_f((h_s, y_s) \mid \mathbf{y}_\mathcal{B}) \geq 0.$$

The proof falls naturally into three parts.

- If $y_s = +1$, then $g(v, h, \mathbf{y}_{\mathcal{A} \cup \{h_s\}}) = g(v, h, \mathbf{y}_\mathcal{A}) \leq 1$. Let $\beta_\mathcal{A} = 1 - g(v, h, \mathbf{y}_\mathcal{A})$. We have

$$\begin{aligned}
&\Delta_f((h_s, +1) \mid \mathbf{y}_\mathcal{A}) \\
=&f_{v,h}(\mathbf{y}_\mathcal{A} \cup \{(h_s, +1)\}) - f_{v,h}(\mathbf{y}_\mathcal{A}) \\
=&\min\left\{\max\left(\max_{h':(h',+1)\in\mathbf{y}_\mathcal{A}} w_{vh'},\ w_{vh_s}\right),\ \beta_\mathcal{A} \cdot w_{vh}\right\} \\
&- \min\left\{\max_{h':(h',+1)\in\mathbf{y}_\mathcal{A}} w_{vh'},\ \beta_\mathcal{A} \cdot w_{vh}\right\} \\
=&\begin{cases} 0 & \text{if } \exists h' : (h', +1) \in \mathbf{y}_\mathcal{A} \wedge w_{vh'} > 0; \\ \min\left(w_{vh_s},\ \beta_\mathcal{A} \cdot w_{vh}\right) & \text{otherwise.} \end{cases}
\end{aligned}$$

  For both cases on the RHS, it holds that $\Delta((h_s, +1) \mid \mathbf{y}_\mathcal{A}) \geq \Delta((h_s, +1) \mid \mathbf{y}_\mathcal{B}) \geq 0$.

- If $y_s = -1$, and $\exists h' : (h', +1) \in \mathbf{y}_\mathcal{A} \wedge w_{vh'} > 0$, then the edge (v, h) is fully covered by $\mathbf{y}_\mathcal{A}$. Thus $\Delta_f((h_s, -1) \mid \mathbf{y}_\mathcal{A}) = \Delta_f((h_s, -1) \mid \mathbf{y}_\mathcal{B}) = 0$.

- Otherwise, $\max_{h':(h',+1)\in\mathbf{y}_\mathcal{A}} w_{vh'} = 0$. Therefore,

$$\begin{aligned}
&\Delta_f((h_s, -1) \mid \mathbf{y}_\mathcal{A}) \\
=&w_{vh} \cdot \big(g(v, h, \mathbf{y}_{\mathcal{A} \cup \{h_s\}}) - g(v, h, \mathbf{y}_\mathcal{A})\big) \\
\geq&w_{vh} \cdot \big(g(v, h, \mathbf{y}_{\mathcal{B} \cup \{h_s\}}) - g(v, h, \mathbf{y}_\mathcal{B})\big) \\
=&\Delta_f((h_s, -1) \mid \mathbf{y}_\mathcal{B})
\end{aligned}$$

  The inequality holds because g is constructed as a concave function. Since g is also non-decreasing, we have $\Delta_f((h_s, -1) \mid \mathbf{y}_\mathcal{A}) \geq 0$.

By definition F is a non-negative linear combination of monotone submodular functions, and hence is also monotone submodular. □

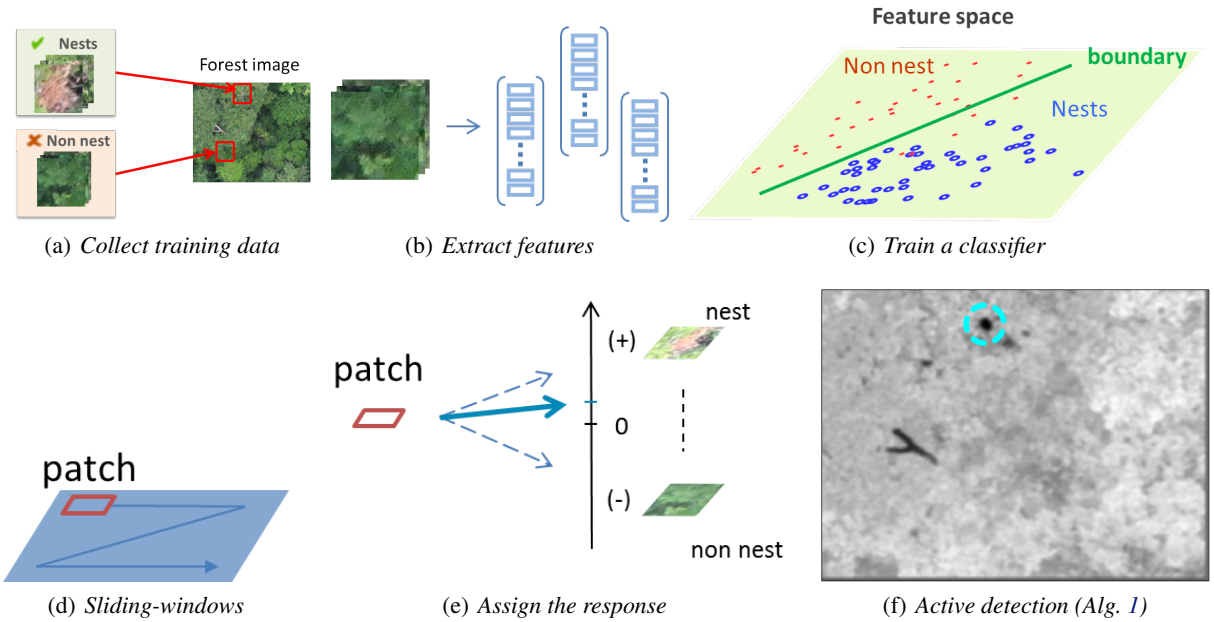
B. Implementation Details

As discussed, our active detection framework builds upon a bipartite graph between voting elements $v \in \mathcal{V}$ that is represented by image features, and hypotheses $h \in \mathcal{H}$ that encode the existence of objects. To benefit the most from user supervision, it is of crucial importance to apply specialized base detectors, optimized for different tasks. This section establishes the connections between several commonly used detectors and our active detection framework, and provides necessary implementation details for different applications discussed in Section 5.

B.1. UAV-forest

The goal of this task is to estimate the distribution of orangutan in a tropical forest in Sumatran, Indonesia. Ideally, the application can also be used as a general tool for biodiversity monitoring, e.g., detecting wildlife activities captured by conservation drones, such as chimpanzees and elephants. Therefore, our attention is focused more on the improvement through user interactions, than that of novel features or classifiers, in a more-readily generalizable setting.

As Fig 3 shows, most of the positive samples have a round shape and their color is very similar to the branches. Unlike pedestrians or cars, they do not contain any meaningful sub-areas. Hence it is difficult to employ the more complicated deformable parts model, as we did in Section 5 for the *VOC2008* data set. In addition, the *UAV-forest* data set suffers from insufficient training examples (e.g., there're at most two orangutan nests per image), making it a challenging task to build a high-quality object detector.



(a) *Collect training data* (b) *Extract features* (c) *Train a classifier*

(d) *Sliding-windows* (e) *Assign the response* (f) *Active detection (Alg. 1)*

Figure 5. System flowchart for orangutan nest detection (*UAV-Forest*).

Local clustering Besides constructing global clusters to group similar voting elements, we use an additional local clustering step to to avoid proposing multiple queries on the same object. As an example, see the response image demonstrated at Fig. 5(f). Once we query some hypothesis at the "Y" branch and get to know its label, we will either cover (when the detection is true), or discount (when the detection is false) the weights for the entire local region. In our experiments, we use mean-shift (Comaniciu & Meer, 2002) for local clustering. We adaptive the code from the *EDISON*[2] system, and set the parameters as follows: a spatial width $h_s = 25$, a feature (range) width $h_r = 10$, and a minimum region area $M = 169$. We adjust the spatial width so as to avoid discounting too much regions by a single observation, while being able to maintain a relatively low computational cost.

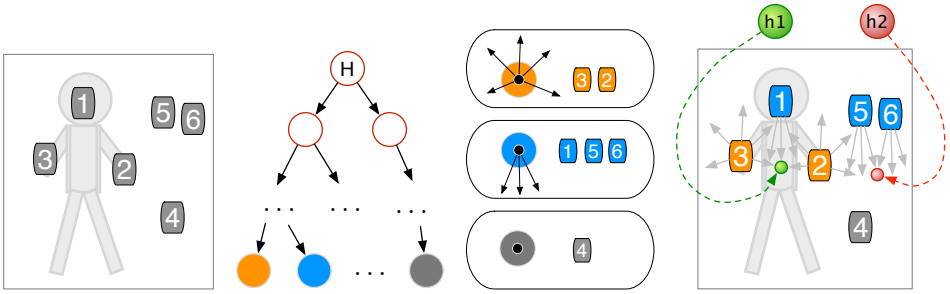
Stopping Criteria We study two stopping criterions to decide when to quit the detection process. The first one, which is used to generated the results in Fig. 4(a), is to stop when the marginal gain by selecting a hypothesis falls below some certain value. The second choice is to stop when the accumulated coverage of the votes exceeds a certain fraction of the total votes in the images. While the first one seems to be a practical choice, the second is in direct connection with the adaptive minimum-cost coverage problem, where one stops once the total utility exceeds a fixed quota.

Detection Process Alg. 2 summarizes the preprocessing steps for active detection. It constructs a bipartite graph that represents the base object detector, which could be used as input for Alg. 1. The system flowchart for this application is illustrated in Fig. 5.

[2]http://coewww.rutgers.edu/riul/research/code/EDISON/index.html

Algorithm 2 Subroutine for constructing Bipartite graph (*UAV-forest*).

Input: Training set $S_{\text{train}} = (I_{\text{train}}, Y)$, Test image I_{test}
Output: Bipartite graph $\mathcal{G}$
$\mathcal{V} \leftarrow \emptyset, \mathcal{H} \leftarrow \emptyset, \mathcal{E} \leftarrow \emptyset$ *% initialize the coverage set*
$(X, Y) \leftarrow \text{EXTRACTFEA}(S_{\text{train}})$ *% feature extraction (training)*
$M \leftarrow \text{LDA}_{\text{TRAIN}}(X, Y)$ *% train a linear classifier*
for all sliding windows v **on** I_{test} **do**
  $\mathcal{V} \leftarrow \mathcal{V} \cup \{v\}$ *% add voting element*
  $x_{\text{test}} \leftarrow \text{EXTRACTFEA}(v)$
  $s \leftarrow \text{LDA}_{\text{TEST}}(M, x_{\text{test}})$ *% distance to decision boundary*
  for all hypotheses h **near** v **do**
    $w_{vh} \leftarrow f(s)$ *% f customizes edge weight*
    if $h \notin \mathcal{H}$ **then** $\mathcal{H} \leftarrow \mathcal{H} \cup \{h\}$ **end if** *% add hypothesis*
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{(v, h, w_{vh})\}$ *% add edge*
  end for
end for



(a) *Input image* (b) *Hough forest* (c) *Clusters* (d) *Hough votes & detection*

Figure 6. Active object detection with Hough forest (*TUD-crossing*)

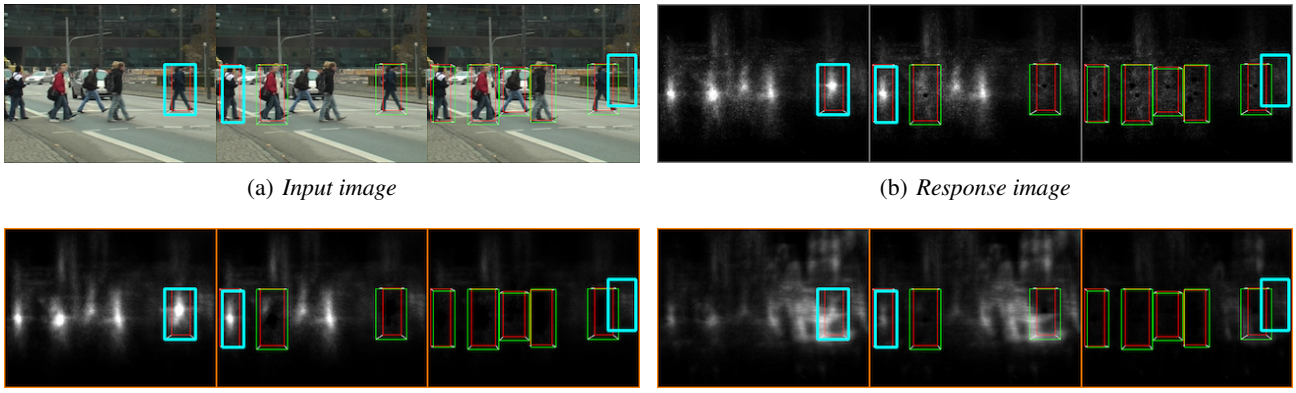
B.2. TUD-crossing and Hough-based Detector

As one of our motivating examples, Hough based approaches (Barinova et al., 2012) offer seamless integration with the active detection framework. Essentially, they produce a set of individual voting elements that are integrated to reason about the existence of a full object. In Gall & Lempitsky (2009), a direct mapping between the appearance of an image patch and its Hough votes is learned through a random forest framework, as illustrated in Fig. 6. Each tree in the forest is constructed based on a set of patches that are sampled from the training collection of images (Fig. 6(a)), and each leaf node in the tree stores the proportion of image patches that belong to an object, and their corresponding offset vectors from the object centroid (Fig. 6(b)). At runtime, patches of the test image are fed to the forest, and passed through the branches of the Hough trees. Then the information stored at the leaf nodes of the trees is used to cast probabilistic votes about the existence of the object at some location (Fig. 6(c), 6(d)).

Clustering the edges. Other than the direct benefit of a ready-to-use voting scheme, the Hough forest framework also provides a natural and intrinsic characterization of the "similarity" of votes. If two image patches fall into the same leaf node, then by default they are clustered together (see Fig. 6(c)). Therefore, when receiving negative feedback from an external expert, the active detector can efficiently update the bipartite graph, by adjusting the weight (i.e., probabilities) of simlar votes that share the same leaves for all the trees in the forest. Note that we are actually clustering edges in the bipartite graph, not voting elements. Therefore, the updates should be focused only on the votes that are pointing to the same directions as the false votes. As an example, in Fig. 6(d), if the active detector proposes h_2, and finds it to be a false detection, then we will discount the weights for (1) the rightmost edge for voting element v_5, and (2) the leftmost edge for voting element v_6. In other words, we just need to discount the two corresponding edges in the blue cluster (Fig. 6(c)) – such that all blue voting elements are updated.

Fig. 7, 8, 9 demonstrate the dynamics of the active detection process. In the following context, we use a discount factor

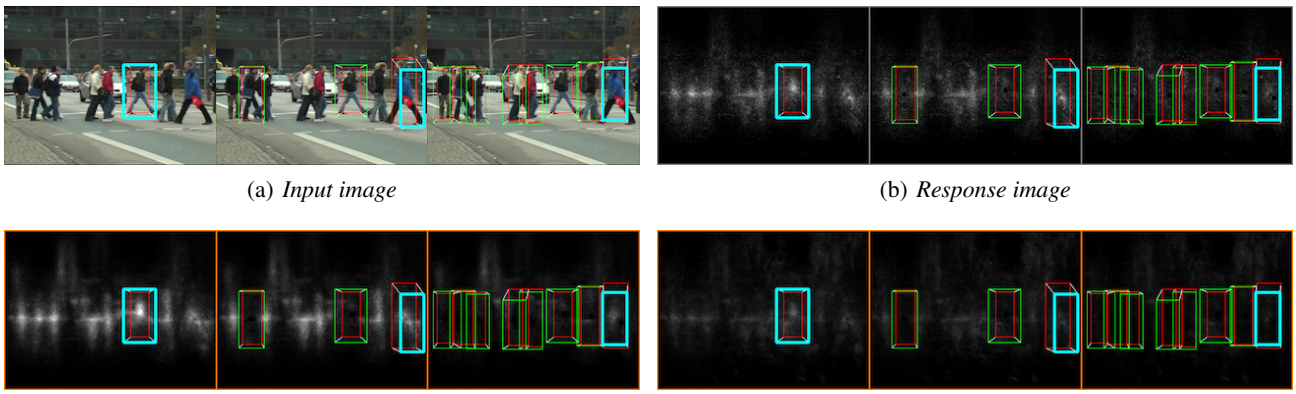
$\gamma = 0.1$ for negative updates.



(a) *Input image* (b) *Response image*

(c) *Positive coverage* (d) *Negative coverage*

Figure 7. The active detection results on the 16th frame of the *TUD-crossing* sequence, at the 1st, 3rd, and 9th iteration. The current detection is highlighted by cyan bounding box. Red bounding boxes indicate the groundtruth labels of pedestrians, and green bounding boxes are the detections made by the active detector (Alg. 1). For illustration purpose, we only show the bounding boxes of the true detections, and that of the current detection (regardless of its label). Each column illustrates the dynamics of the corresponding items: (a) Detections on the input image. (b) Response (Hough) image. (c) Utility obtained given the current detection is true (i.e., total sum of edge weights covered by a observation a positive label at given locations). (d) Utility obtained given the current detection is false.



(a) *Input image* (b) *Response image*

(c) *Positive coverage* (d) *Negative coverage*

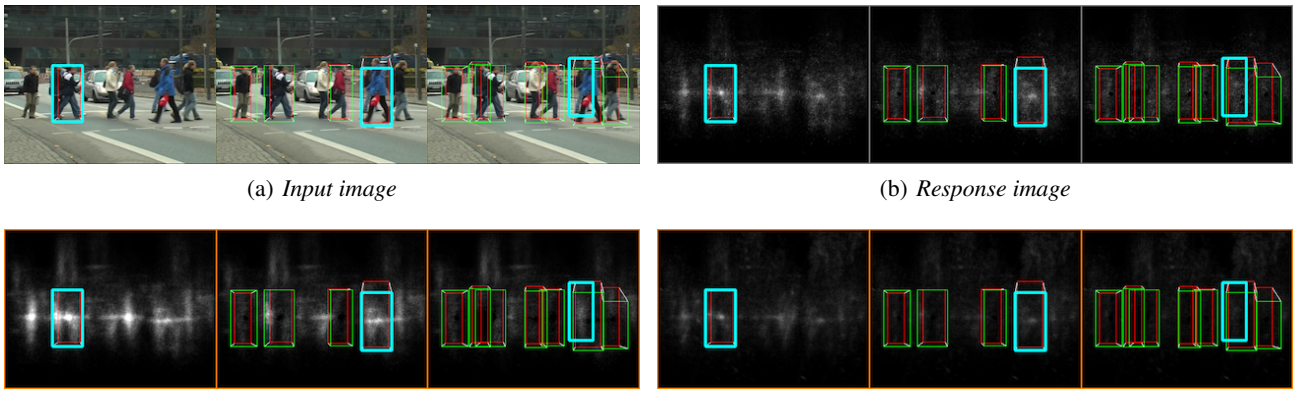
Figure 8. The active detection results on the 41th frame of the *TUD-crossing* sequence, at the 1st, 3rd, and 9th iteration.

We start active detection from the 16th frame of the video sequence (Fig. 7(a)). Each time we finish detecting one frame, we take the next frame that is 5 frames away from the current one as input. That is, the sequence of test images are frame 16, 21, 26, 31, 36, 41, 46, 51, etc.

In Fig. 7, we show three detections made by the active detector. As we can see from the third image of Fig. 7(a), the active detector makes a false prediction at the 9th iteration, where it mistakes the pole as a pedestrian. According to the negative update rule in the active detection framework, we will then discount all the similar votes, which indicates that, in the following frames, the negative coverage by selecting a similar hypothesis (e.g., detecting the pole again) will be discounted.

Fig. 8 demonstrates the above changes. Similar as Fig. 7, we show three detections made by the active detector, on the 41st frame of the video sequence (i.e, the 6th test image). On the first image of Fig. 8(d), we can see clearly that the negative coverage of the pole is dramatically reduced, comparing with that of Fig. 7(d). Therefore, the active detector will be less likely to select that false positive, and thus shows better performance.

Another interesting result we found in Fig. 8 is, by using the Hungarian matching algorithm, we can actually switch the associated ground truth object of the detections made in the previous iterations, to a better matching. For example, in the



(a) *Input image* (b) *Response image*

(c) *Positive coverage* (d) *Negative coverage*

Figure 9. The active detection results on the 51th frame of the *TUD-crossing* sequence, at the 1st, 4rd, and 8th iteration.

second and third image of Fig. 8, the algorithm proposes two detections (illustrated as cyan bounding boxes), both could be matched to the same pedestrian. When the dynamic matching algorithm finds a better matching at the 9th iteration (third image), it discards the predictions made at the 3rd iteration (second image), and therefore gives better detection results.

Furthermore, we show in Fig. 9 how our active detection framework can detect highly overlapped objects, without invoking the non-maximum suppression mechanism. In the first image, it starts to detect the 8th input image (i.e, the 51st frame). When it find the pedestrian (in blue jacket) closer to the camera at the 4th iteration (second image), the algorithm doesn't remove the entire surrounding region. Instead, it only covers its direct votes, and all neighboring votes. Those edges that are not in any form associated with the hypothesis that represent the pedestrian detected, are not affected. As a result, in the third image (8th iteration), the algorithm successfully identifies the pedestrian further from the scene – who is nearly 90% occluded. In this scenario (i.e., dealing with highly overlapped objects), our algorithm behaves in a similar manner manner as that of Barinova et al. (2012), but it has the advantage that, it allows us to address the active detection problem – by invoking the adaptive submodularity property of the objective function – in a principled way.

B.3. PASCAL VOC and Deformable Parts Model

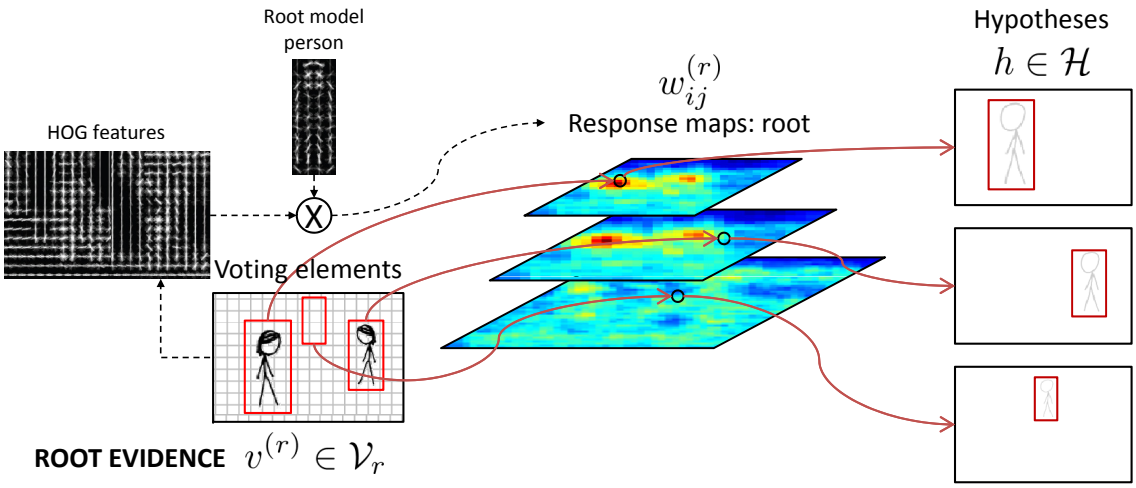


Figure 10. Deformable parts model as a base detector for the active detection framework. Edge weights come from the response maps (only the root filter is shown)

Another commonly used detection approach involves the use of sliding (or scanning) windows, which could be considered as an exhaustive search strategy over all possible locations within an image. It relies on evaluating a confidence function

over many rectangular subareas of the image. For each subarea, we can think of the detector as solving a classification problem, based on some classifiers trained on a set of labeled objects. A seminal work along this line is the deformable parts model (DPM) of (Felzenszwalb et al., 2010). It offers a robust solution for detecting objects with high intra-class variations (e.g., person with different poses or shapes), by applying a set of filters (i.e., "root" filter that describes the overall shape of the object, plus a fixed number of "part" filters) for each object category.

One problem associated with the sliding-window approach is to deal with considerably overlapping windows. For detection of multiple object instances, the usual way to counteract this problem is by non-maximum suppression or mode-seeking. However, such approaches are oblivious of any internal structure in the image: it tends to eliminate true objects that are heavily overlapping. In the main paper, we haven shown how sliding window detectors can be adapted into our framework, such that the issue could be avoided. Here we provide a slightly more detailed explanation:

In Figure 10, we demonstrate how we can build such a bipartite graph step-by-step. To discriminate part and root filters, we use $v^{(r)}$ and $\mathcal{V}_r$ to represent a single root voting element, and the corresponding evidence space. For simplicity we just show the root filters, but part filters could be modeled in the same way, simply by substituting the filters and hypotheses to the ones for object parts. As illustrated, the voting elements we use in the experiments, are the patches which are contained in the red window (bounding boxes in the left figure of Figure 10). The strength of each vote is in direct relation with its filter response. Particularly, all voting elements within the associated window of an hypothesis h equally share the response value of that window.