Interactive Object Detection

Angela Yao[1] Juergen Gall[1,2] Christian Leistner[1] Luc Van Gool[1,3]
[1]ETH Zurich, Switzerland [2]MPI for Intelligent Systems, Germany [3]KU Leuven, Belgium
{yaoa, leistner}@vision.ee.ethz.ch jgall@tue.mpg.de vangool@esat.kuleuven.be

Abstract

In recent years, the rise of digital image and video data available has led to an increasing demand for image annotation. In this paper, we propose an interactive object annotation method that incrementally trains an object detector while the user provides annotations. In the design of the system, we have focused on minimizing human annotation time rather than pure algorithm learning performance. To this end, we optimize the detector based on a realistic annotation cost model based on a user study. Since our system gives live feedback to the user by detecting objects on the fly and predicts the potential annotation costs of unseen images, data can be efficiently annotated by a single user without excessive waiting time. In contrast to popular tracking-based methods for video annotation, our method is suitable for both still images and video. We have evaluated our interactive annotation approach on three datasets, ranging from surveillance, television, to cell microscopy.

1. Introduction

The demand for annotated image and video data is rapidly growing. Within the computer vision community, successful object detection algorithms continue to be heavily reliant on large amounts of annotated data for training and evaluation. Outside of the community, statistical analysis from either images or video occur in a variety of fields. Biologists routinely count cells from microscopy images, while urban planners may sit through hours of video footage tabulating the number of pedestrians using a crosswalk.

Object annotation in image and video can be laborious, tedious, and expensive. One option is to outsource, typically through a crowdsourcing platform such as Mechanical Turk [2, 9, 13, 16, 21, 22]. Crowdsourcing can be attractive, with its low costs, but does require careful planning for an effectively designed task [19], as well as a strategy for quality control of the gathered annotations [24]. Unfortunately, it cannot be used for annotating confidential data, e.g., from an industrial environment, or data requiring expert knowledge, e.g., medical images.

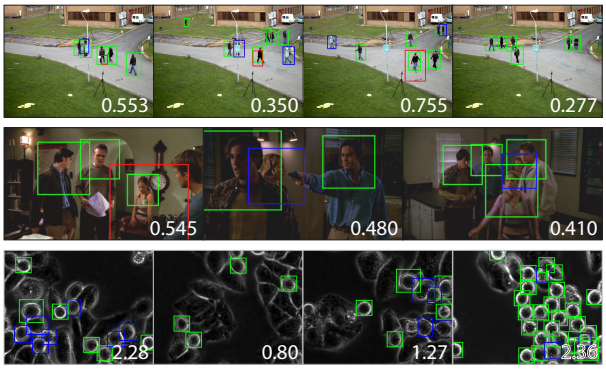


Figure 1. Examples from our interactive object detection framework for annotation. Green, red, and blue bounding boxes denote true positives, false positives, and false negatives respectively according to the annotation task. The numbers denote the predicted annotation cost, i.e., the cost to correct all detection errors in an image. The predicted annotation cost allows the user to select images for correcting detections and updating the object detector. Images best viewed in colour.

Another alternative for simplifying annotation is active learning. Active learning aims to minimize human effort in annotation by requesting labels for only a portion of the training data, e.g., data with high label uncertainty [12]) or data which can minimize some expected error measure [15]. The usual objective is, given a fixed (human) annotation budget, to train a classifier with maximum accuracy [8, 9, 21]. This paradigm is well-suited for learning from unlabelled data that is readily available, e.g., from Flickr or YouTube, but not for annotation tasks that require a *complete* set of labels for some fixed data collection, e.g., in medical imaging. Another major limiting factor which prevents active learning algorithms from being used in practice is that the majority of algorithms are still "learning"-centered [18]. For example, the queried data is determined only by potential gain for the algorithm, assuming that all queries are equally expensive, even though this has been shown not to be the case [20]. In addition, iterations between queries may be expensive or time consuming, making it unrealistic or impractical for interaction with a user.

Our goal is to provide all the labels of some fixed data collection with as little human effort as possible. In particular, we focus on a bounding box level of annotation, i.e.,

the drawing of bounding boxes around all instances of some object class. We present here an interactive object detection framework for annotation that minimizes annotation costs for humans. At the heart of the framework is an incremental learning approach which continually updates an object detector and detection thresholds as a user interactively corrects annotations proposed by the system. The learning approach is paired with an active learning element which predicts the most "difficult" images, i.e., those with the highest annotation cost. These examples are then corrected by the user for subsequent updates. The framework is illustrated in Fig. 1. Our contributions include:

- an efficient incremental learning approach for interactively annotating objects in images or video
- a user study measuring the real human annotation costs for correcting object hypotheses
- a model that estimates the optimal object detection threshold for minimizing annotation cost
- a model that predicts images with the highest annotation cost for active learning

Finally, we were strongly motivated to design an "annotator"-centered algorithm, in contrast to the many existing "learning"-centered active learning algorithms. All optimizations and evaluations were made in terms of human annotation cost. Using datasets drawn from real-world annotation tasks such as surveillance, television, and cell microscopy, we show in our experiments that incremental learning with actively selected examples results in a lower overall annotation cost than standard incremental learning.

2. Related Work

Given the expense of human annotation, much research effort has been dedicated towards leveraging the use of unlabelled data, for instance, using semi-supervised learning [4, 11], or weakly-labelled data, which is faster to annotate [3]. Additionally, there has been extensive research in learning methods that are able to share knowledge, e.g., transfer learning [10, 14], or active learning, by querying for labels intelligently [9, 20, 21, 22].

These learning paradigms all have a common goal of reducing the number of annotations needed. On the other end of the spectrum, crowdsourcing has become the method of choice for increasing the number of annotations, particularly for building large corpora [2, 13, 16]. Reliable usage of crowdsourced labels has prompted several studies on best practices [19, 23, 24] and is an active field of research.

A few works have focused on making the annotation process itself more efficient for the human annotator. In [9], the authors propose a method for categorizing images using binary queries rather than asking annotators to select the category from some predefined list. In [20], the authors predict a tradeoff between the effort required for manually segmenting an image versus the information gain from using the segmented image for training, thereby introducing some human weighted considerations in the query order. Finally, [21] trains object detectors from Flickr images and uses hash-based queries to efficiently select query images. As noted previously, however, this paradigm is not suitable for annotating complete sets of images.

The video annotation system of [22] is conceptually the most similar work to ours. Annotations are derived from tracking results and active learning is used to intelligently query the human annotator for corrections on the tracks. The cost of annotation, i.e., the number of "clicks", has been minimized, though the system requires a full pass over a track before each correction query. Depending on the tracker's speed, interaction with a human annotator can become cumbersome. Our proposed system, however, is based purely on detections, making it more versatile. In contrast to [22], it is possible to (i) annotate both video and non-continuous image collections; (ii) control the pool size of the active learning component. The user thus does not need to wait for a completed track before getting feedback.

3. Interactive Annotation

The goal of interactive annotation is to label a fixed set of images with minimal annotation cost (Sec. 4) for the user. In our setup shown in Fig. 5, the user corrects object hypotheses that are generated on the fly. Corrected hypotheses are then used to update the detector (Sec. 3.1) and to determine an optimal detection threshold (Sec. 3.2.1). Since the order of processing the images can impact the detector performance, we introduce an active learning approach to reduce the overall annotation cost (Sec. 3.2.2).

3.1. Incremental Learning

Offline Hough Forests Our object detector of choice is the Hough forest [7], which detects objects using the generalized Hough transform and randomized decision trees [1]. In theory, any incrementally learned object detector is applicable in our framework. We have chosen Hough forests because they are extremely fast and efficient for both training and detection, making them particularly suitable for interactive applications. We first review the offline Hough forest but refer the reader to [7] for a more thorough treatment.

During training, a set of patches $\mathcal{P}$ is randomly sampled from the bounding-box annotated training data. Each patch P_i is affiliated with a class label c_i and a displacement vector d_i of the patch's center to the training example's center (if the patch is sampled from within a bounding box). Each tree $\mathcal{T}$ in the forest $\mathcal{T}$ is constructed recursively from the root node downwards. For each node, a set of binary tests $\{t\}$ are randomly generated which could split the training data $\mathcal{P}$ into two subsets $\mathcal{P}_L(t)$ and $\mathcal{P}_R(t)$. Each binary test selects an image feature and compares the feature values at

two pixel locations in a patch. Based on a threshold associated with the test, the patch is added either to the left or right split. The optimal binary test $t^\star$ maximizes the gain

$$\Delta H(t) = H(\mathcal{P}) - \sum_{S \in \{L,R\}} \frac{|\mathcal{P}_S(t)|}{|\mathcal{P}|} \cdot H(\mathcal{P}_S(t)). \quad (1)$$

Depending on the measure H used, a node can either be a classification or regression node. For classification, entropy

$$H(\mathcal{P}) = - \sum_{c \in \{neg,pos\}} p(c|\mathcal{P}) \log p(c|\mathcal{P}) \quad (2)$$

is used, where $p(c|\mathcal{P})$ is given by the percentage of samples with class label c in the set $\mathcal{P}$. For regression, the sum-of-squared-differences is used as an objective function:

$$H(\mathcal{P}) = \frac{1}{|\{d_i : c_i = pos\}|} \sum_{d_i : c_i = pos} \|d_i - \bar{d}\|_2^2, \quad (3)$$

where $\bar{d}$ is the mean of the displacement vectors d_i of the positive examples ($c_i = pos$) in the set $\mathcal{P}$. After a good split has been found, the binary test $t^\star$ is stored at the node and the sets $\mathcal{P}_L(t^\star)$ and $\mathcal{P}_R(t^\star)$ are passed to the left and right child node, respectively. Trees grow until some stopping criterion is met. We use three criteria: maximum tree depth, a strictly positive gain, i.e., $\Delta H(t^\star) > 0$, and a minimum number of samples for the sets $\mathcal{P}_L(t^\star)$ and $\mathcal{P}_R(t^\star)$. At the leaf nodes, the patches $\mathcal{P}$ arriving at leaf L are removed and only $p(c|L) = p(c|\mathcal{P})$ and the positive offsets $D_L = \{d_i : c_i = pos\}$ are stored.

For detection, patches extracted from the test image are passed through all trees in the forest. Depending on the reached leaf L, votes are cast for objects centers according to the stored displacements D_L with a probability proportional to $p(c|L)$. Detections are determined by local maxima in the Hough space.

Incremental Hough Forests Hough forests are commonly trained offline, using all training samples at the outset. Offline training, however, is not suitable for an annotation task where training and annotation iterates. Therefore, we train the Hough forest incrementally, similar to the on-line learning of [17], with the exception that we additionally store the patches at the leaves after training. Having an image (referred to as a training example from here onwards) initially annotated by the user, positive patches are sampled from the bounding boxes and negative patches from the background. Initial trees T_{init} are trained with patches $\mathcal{P}_{init}$ as in the offline case. Before the user annotates the next image, T_{init} are applied to the image to generate object hypotheses. The user then only needs to correct wrong hypotheses, i.e., missed detections (FN) and false positives (FP). Positive patches are then sampled from the true positives and corrected false negatives while negative patches are sampled from corrected false positives. These newly sampled patches $\mathcal{P}_{incr}$ are used to continue training the trees. $\mathcal{P}_{incr}$ are first passed through T_{init} and arrive at leaves $\{L\}_{init}$. As new patches arrive at a leaf L, an optimal binary test $t^\star$ that satisfies the splitting criteria is determined. Finally, the class probability $p(c|L)$ and the list of displacement vectors D^L are updated. Since the number of stored patches would grow to infinity in this setup, we limit the maximum number of patches stored at each node. When this is exceeded, a subset of the patches is randomly selected and the others are removed.

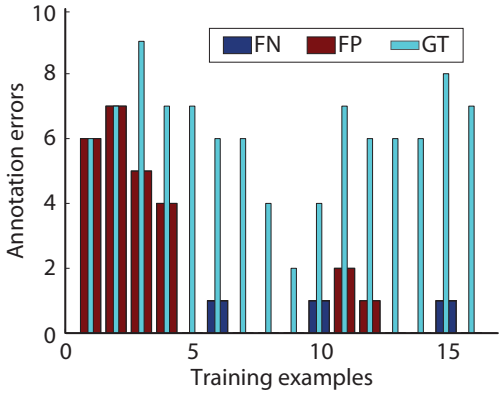


Figure 2. Incremental learning reduces annotation cost since only wrong detections (missed detections (FN) or false positives (FP)) need to be annotated. After 4 training examples, most objects (GT) are already detected.

The difference in annotation cost between offline and incremental learning is shown in Fig. 2. Assume that the number of training examples is fixed to 16. Offline training would require the annotation of all existing objects (ground truth (GT)). Incremental learning requires only the annotation of the objects that are not similar to previously annotated objects, which reduces the annotation cost for the user.

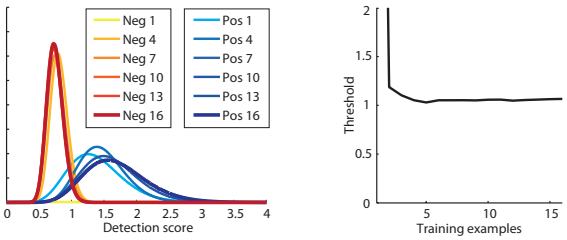
3.2. Annotation Cost

Object detectors like Hough forests provide a score for each object hypothesis which can then be thresholded when deciding to accept or reject hypotheses. By defining an annotation cost model when annotating images, one can find the best threshold $\tau^\star$ that minimizes annotation cost.

In this work, we model the annotation cost by

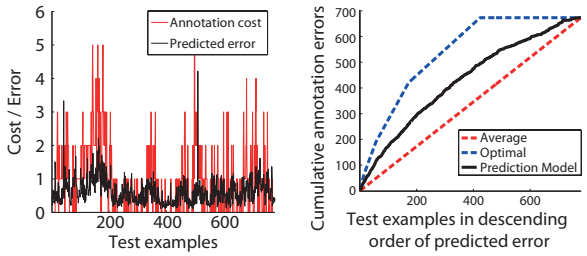
$$f(\tau) = K + f_{FP}(n_{FP}(\tau)) + f_{FN}(n_{FN}(\tau)), \quad (4)$$

where $n_{FP}(\tau)$ and $n_{FN}(\tau)$ are the number of false positives and false negatives for a given threshold τ, and K is a constant cost factor. The differential functions f_{FP} and f_{FN} measure the cost for correcting a FP or a FN. Usually, an equal cost is assigned to FPs and FNs, corresponding to $f_{FP}(x)=f_{FN}(x)=x$. Since Eq. (4) can be evaluated only when ground truth is available, we propose a method that aims to minimize $f(\tau)$ during incremental learning.



(a) Distribution of detection scores (b) Threshold convergence

Figure 3. (a) Detection scores of positive (blue) and negative (red) examples can be modelled by Gamma distributions. Distribution parameters are updated after each incremental training step $(1, \ldots, 16)$. (b) Based on the distributions, the threshold for the detection scores can be estimated.



(a) (b)

Figure 4. (a) The prediction model predicts images with a high annotation cost. (b) Ranking the images by the predicted error shows that the cumulative annotation cost is between random order (average) and optimal order, indicating that the measure is useful for selecting images with high cost.

3.2.1 Threshold Estimation

As incremental learning iterates between training and testing, one obtains positive and negative hypotheses with the corresponding detection scores, denoted by $\mathcal{S}_{pos}$ and $\mathcal{S}_{neg}$. Therefore, we can estimate an optimal threshold $\tau^\star$ for Eq. (4) based on $\mathcal{S}_{pos}$ and $\mathcal{S}_{neg}$. To this end, we model the distributions of the scores conditional to a positive or negative hypothesis, denoted by $p(s|pos)$ and $p(s|neg)$. In our experiments, we have discovered that the conditional probabilities are well modelled by Gamma distributions:

$$p(s|pos) = \Gamma(k_{pos}, \theta_{pos}), \quad (5)$$

$$p(s|neg) = \Gamma(k_{neg}, \theta_{neg}). \quad (6)$$

The parameters k and θ are estimated by moment estimates from the sets $\mathcal{S}_{pos}$ and $\mathcal{S}_{neg}$. The estimated probabilities over several training iterations are shown in Fig. 3(a).

To find an optimal threshold for (4), we formulate the problem as

$$\underset{\tau}{\operatorname{argmin}} f_{FP}\left(p\left(FP|\tau\right)\right) + f_{FN}\left(p\left(FN|\tau\right)\right), \quad (7)$$

where $p\left(FP|\tau\right)$ and $p\left(FN|\tau\right)$ are the probabilities that a hypothesis is a FP or a FN for a given threshold τ. Based on (5) and (6), we obtain

$$p\left(FP|\tau\right) = p(neg) \cdot \int_{\tau}^{\infty} p(s|neg)\,\mathrm{d}s, \quad (8)$$

$$p\left(FN|\tau\right) = p(pos) \cdot \int_{0}^{\tau} p(s|pos)\,\mathrm{d}s, \quad (9)$$

where $p(pos) = \frac{|\mathcal{S}_{pos}|}{|\mathcal{S}_{neg}|+|\mathcal{S}_{pos}|}$ and $p(neg) = 1 - p(pos)$. Since (7) is differentiable, we can estimate for $\tau^\star$ very efficiently by local optimization using the average of the lowest positive score and the highest negative score as initial threshold. Although (7) can be non-convex, the local optimization is sufficient in practice; see Fig. 3(b).

Once new training examples are gathered for training, the parameters of the distributions (5) and (6) are updated. The evolution of the distributions with an increasing number of incremental training steps is shown in Fig. 3(a). Fig. 3(b) shows that the threshold starts very high but stabilizes after a few training iterations. Since the threshold estimation requires at least two training images, the annotation cost for the first two images is the same as in the offline case as shown in Fig. 2. With more iterations, the trees detect objects better and a stable threshold can be estimated. In the example shown in Fig. 2, the detection already performs well after 4 training examples.

3.2.2 Active Learning

Incremental learning processes the images according to a fixed scheme, e.g., according to the order of the images or every n^{th} image. In active learning, one can reduce the annotation cost by selecting the most useful training images. For our annotation task, we assume this to be equivalent to selecting the most "difficult" images, i.e., those with the highest annotation cost. Similar to Sec. 3.2.1, we can estimate the annotation cost of a given set of hypotheses cost with scores $\mathcal{S}$ using

$$f_{pred}(\mathcal{S}, \tau) = \sum_{s \in \mathcal{S}} f_{FP}\left(p\left(FP|s,\tau\right)\right) + f_{FN}\left(p\left(FN|s,\tau\right)\right), \quad (10)$$

where τ is the estimated threshold. The predicted annotation cost f_{pred} can be computed by

$$p\left(E|s,\tau\right) = \frac{p\left(s|E,\tau\right)p\left(E|\tau\right)}{p\left(s|neg\right)p(neg) + p\left(s|pos\right)p(pos)}, \quad (11)$$

where $E \in \{FP, FN\}$ and

$$p(s|FN,\tau) = \begin{cases} 0 & \text{if} \quad s \geq \tau, \\ \frac{p(s|pos)}{\int_0^\tau p(s|pos)\,\mathrm{d}s} & \text{if} \quad s < \tau, \end{cases} \quad (12)$$

$$p(s|FP,\tau) = \begin{cases} \frac{p(s|neg)}{\int_\tau^\infty p(s|neg)\,\mathrm{d}s} & \text{if} \quad s \geq \tau, \\ 0 & \text{if} \quad s < \tau. \end{cases} \quad (13)$$

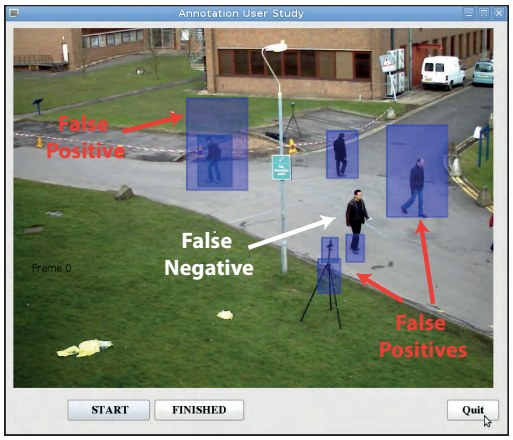


Figure 5. Interface for user study.

Performance of the prediction model is shown in Fig. 4. Having incrementally trained a Hough forest on 16 images of a sequence from PETS [6], we applied the Hough forest with the prediction model to the remaining images of the sequence. Fig. 4(a) shows that the predicted annotation cost is high at frames where the real annotation cost is high, e.g., around frame 150 or 500. Fig. 4(b) shows the accumulated annotation cost when the images are sorted by the predicted cost. In the active learning setup, the user receives the predicted annotation cost as feedback for images that have not been annotated and can then select the images with a high value, e.g., frame 500. In our quantitative experiments, we randomly select a set of unannotated images (i.e., our active learning pool) and rank them according to the predicted annotation cost. Only the image with the highest value is presented for correction of the hypotheses. In Sec. 5, we show that the active learning approach tends to select images that contain positive and negative samples that are less redundant to previously sampled training data, thereby reducing the annotation cost when compared to incremental learning with a fixed scheme.

4. User Study

Since $f_{FP}(x) = f_{FN}(x) = x$ is an unrealistic annotation cost, we conducted a user study to measure the annotation cost for our interface shown in Fig. 5. We presented 24 frames from the PETS 2009 [6] sequence *S2.L1, View001* with varying number of FPs (0-7) and FNs (0-9) on each frame and asked 14 participants to correct the annotations for a pedestrian class. FPs shown were real instances taken from results of the offline system. To account for absolute timing differences across participants (some were more experienced annotators), we normalized each participant's time per frame with respect to the overall time used.

In our study, we investigated a linear cost model, i.e., $f_{FP}(x) = w_{FP} \cdot x$ and $f_{FN}(x) = w_{FN} \cdot x$ (4), and estimated w_{FP} and w_{FN} by a linear regression. The regressed parameters in both absolute and relative time are shown in Table 1. The results reveal that it takes roughly twice as long to correct a FN versus a FP. When integrating this model into our threshold estimation, it implies that the estimated threshold should be lower than when FPs and FNs have equal cost. For instance, using equal costs, we obtain a threshold of 1.11 after 16 images as shown Fig. 3(b). Using the model given in Table 1, the threshold estimate is 1.06.

	w_{FP}	w_{FN}
absolute time (s)	0.78 ± 0.21	1.69 ± 0.16
relative time	0.25 ± 0.06	0.54 ± 0.05

Table 1. Human annotation costs from user study.

5. Experiments

Datasets and Evaluation. We test our annotation framework on three datasets, covering a variety of data ranging from surveillance, television to cell microscopy. The first is the PETS 2009 S2.L1 video sequence (795 frames, continuous) for which we annotate pedestrians (∼4700). The second is Buffy Stickmen V3.0 [5] (748 frames, disjoint), for which we annotate the human upper body (∼1350). Finally, we test our system on a set of microscopy images (60 frames, disjoint) to annotate cells (∼950)[1]. For all experiments, we count a detection as a true positive if the intersection-union ratio of the detection and ground truth bounding box is greater than 0.5.

PETS. For incremental learning, we evaluate the impact of several parameters on the annotation cost to find a good balance between performance and computational efficiency; see Fig. 6. In each experiment, 64 training examples were used, taken at equal intervals from the sequence.

Patches stored per node. A smaller number of patches per node limits the memory requirements and reduces the training time for incremental learning. In Fig. 6(a), the error is slightly reduced for 400-600 patches per node but increases with more than 800 patches. This shows that restricting the memory capacity for incremental learning can reduce the bias towards the first patches and overfitting. For remaining experiments, we fix the maximum number of patches per node to 600.

Votes cast per leaf. To reduce time, one can also reduce the votes per leaf that are cast during detection. The impact of this parameter (Fig. 6(b)) is very small since the number of positive and negative samples are already limited.

Patches per child node before split. [17] showed that the number of patches is a good criteria for accepting a split at a node. Fig. 6(c) shows that the minimum number of patches for each child that are required to accept a split has a strong impact on the annotation error. A very small value gen-

[1]Our images are taken from differential interference contrast microscopy. We count the number of cells arrested in mitosis (see last row in Fig. 1), which could be used for a mitotic index assay.

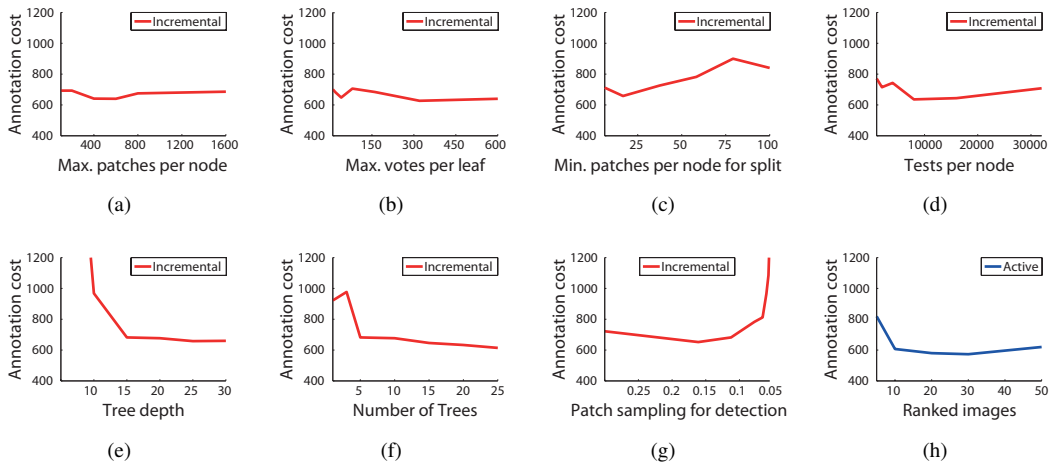


Figure 6. (a) Maximum samples per node during training. (b) Maximum votes per leaf during testing. (c) Minimum number of samples for each children that is required for accepting a split. (d) Generated tests per node during training. (e) Maximum depth of trees. (f) Number of trees. (g) Percentage of patches sampled during testing. (h) Number of ranked images for active learning.

erates many splits without having enough information and the trees grow too fast. A very large value slows down tree growth and trees are not discriminative enough at the beginning, which increases the annotation cost. For remaining experiments, we fixed this parameter to 20.

Number of tests per node. The number of randomly generated tests at each node impacts performance. A small number does not optimize the trees for the detection task but many tests increase the risk of overfitting. Fig. 6(d) shows that 8000 tests are a good trade-off.

Tree depth. The trees require a minimum depth for reasonable performance. Performance gain for trees with depth greater than 15 is small and saturates at 25 (Fig. 6(e)).

Number of trees in forest. A similar minimum value can also be observed for the number of trees (Fig. 6(f)). At least 5 trees are required to achieve a good performance; after 5, the error further decreases at increasing computational cost.

Detection patch subsampling. Computation time can be reduced by sub-sampling the image patches during detection. Using only 10% of the densely sampled patches does not result in performance loss and 5% is still acceptable though lower values become problematic (Fig. 6(g)).

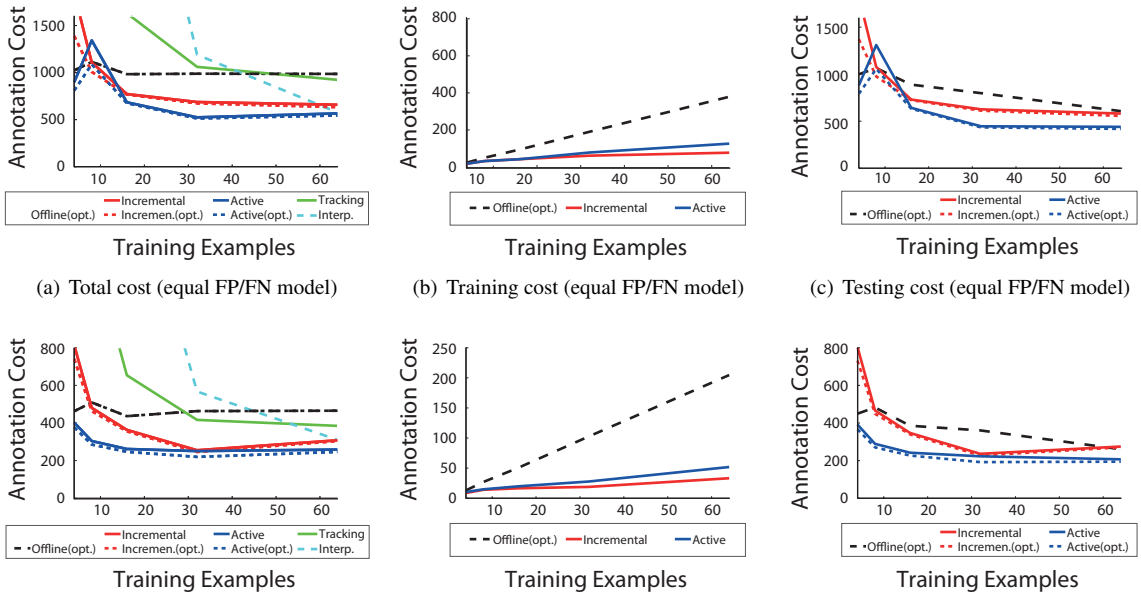
For active learning, we look at the size of the learning pool, i.e., the set of images randomly drawn from the unannotated images and ranked according to the prediction model. Fig. 6(h) shows that at least 10 images should be drawn, whereas larger numbers result only in small changes. Note that simple random selection corresponds to having only 1 image. This experiment was performed with only 16 training examples instead of 64 since the impact of this parameter is stronger for fewer training examples.

For comparing the annotation cost of offline, incremental, and active learning, we selected a fixed number of N training examples (referred to as training) and applied the learned trees to the remaining images (referred to as testing). Training examples were drawn at equal intervals from the sequence for offline and incremental learning. Active learning selects the training examples automatically except for the first two, which are the same for all methods. To evaluate our threshold estimation, we compared against an "optimal" threshold, i.e., the threshold for which one can achieve the lowest annotation cost determined *post-hoc* based on ground truth data. Note that the offline learning does not estimate a threshold, therefore we report only the performance using the theoretical optimal threshold.

Annotation costs are plotted in Fig. 7. The active and incremental learning approaches have lower annotation costs than the offline approach once a minimum number of training examples have been presented ($\sim$10 frames). Since the frames in a video sequence are often similar, incremental and active learning already detect many objects in the training examples and require less annotations for training (Fig 7(b)). The active learning has a lower test annotation cost (Fig 7(c)) than the incremental, despite having a higher training cost, indicating that training with higher costing "difficult" examples pays off with a better detector.

Similar trends are observed in the study-based cost model as the equal FP/FN cost model, though the differences between the active and incremental learning are more pronounced, especially with fewer training examples. Threshold estimates are generally accurate for both the incremental and the active learning, as annotation cost are only marginally higher than when using the optimal threshold (Figs. 7(a),7(d)). Example annotations of the active learning, along with the predicted errors according to the study cost model are shown in the top row of Fig. 1.

Finally, we compare the use of our framework against linear interpolation and a tracking approach [17]



(a) Total cost (equal FP/FN model) (b) Training cost (equal FP/FN model) (c) Testing cost (equal FP/FN model)

(d) Total cost (user-based model) (e) Training cost (user-based model) (f) Testing cost (user-based model)

Figure 7. Annotation cost for PETS using (a-c) the equal FP/FN cost model ($w_{FP} = w_{FN} = 1.0$) and (d-f) the user-based model from Table 1 ($w_{FP} = 0.25$; $w_{FN} = 0.54$). (a,d) show the overall cost for annotation, (b,e) show the cost only for training and (c,f) the cost only for testing. The dashed lines are theoretical values where the threshold is optimized based on ground truth data. The solid lines show the real performance of incremental and active learning. Active learning selects images with higher costs for training but performs better on the test images, yielding an overall lower annotation cost.

(Figs. 7(a),7(d)). As expected, tracking and interpolation both have very high annotation costs when the number of training examples are low (<30 frames). The annotation costs become comparable only at higher number of training examples, lending strong support for using our framework for annotating video.

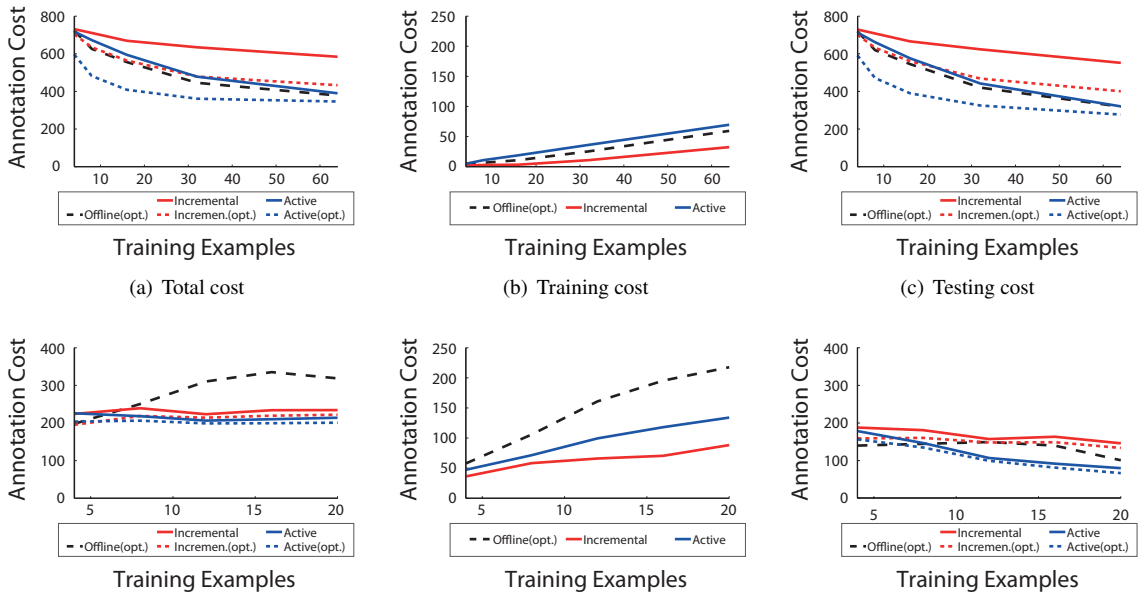
For 10 scales, our implementation requires 13sec for feature computation, 1.5sec for detection, and 2sec for training 5 trees incrementally. The median time for users to correct an image in the user study is about 12sec, indicating that the system operates in a reasonable time-frame though improvements for speedup can still be made. In particular, the features can be precomputed. The active learning in our setting requires that 10 images are processed in advance. This means that the system needs to be 15sec ahead of the user.

Buffy. We apply only the study-based cost model to the Buffy dataset (Fig. 8). This dataset is more challenging than PETS as the appearance variations are much greater. With an optimal threshold, the incremental approach performs comparably to the optimal offline, while the active learning has much lower annotation costs (Fig. 8(a)). The threshold estimation does not work as well as on PETS but the differences become smaller with an increasing number of training examples in the case of active learning. Since Buffy contains only few frames taken from various episodes, it is very difficult to learn the distributions of Eq. (5) and (6) of the full dataset from very few training examples. Nevertheless, active learning still performs comparable to offline learning with optimal threshold. Example annotations of the active learning are shown in the middle row of Fig. 1.

Cell. Detecting the mitotic cells in the microscopy images is a relatively easy task, as the cells have a very regular, round-ish shape (see examples in bottom row of Fig. 1). As such, there is little difference in annotation cost between the three learning approaches as well as between the optimal and estimated threshold for the incremental and active learning (Fig. 8(d)). It takes only a few training examples to achieve reasonable performance (∼5 images) and using more examples is not beneficial. However, using more examples also does not have a negative impact, though the cost increases for offline learning. Since it is in practice not a priori clear how many training examples are needed for minimal annotation cost, it is very important that the annotation cost does not increase with an increasing number of training examples. This behaviour is shown by incremental and active learning for all three datasets.

6. Conclusion

We have presented an annotator-centric approach for interactive object annotation in still images and video sequences. To this end, we modelled the annotation cost by the amount of time users need to correct object hypotheses. Using the model, we can successfully estimate the optimal detection threshold that minimizes the annotation cost and predict the expected annotation cost for active learning. Finally, the active learning approach has been shown to out-



(a) Total cost (b) Training cost (c) Testing cost

(d) Total cost (e) Training cost (f) Testing cost

Figure 8. Annotation cost for Buffy (a-c) and Cells (d-f) with cost model ($w_{FP} = 0.25$; $w_{FN} = 0.54$).

perform incremental and offline learning.

Acknowledgements We thank Chia Huei Tan of the Meraldi group at ETH Zurich for the cell microscopy images. Research was funded by the Swiss National Foundation NCCR project IM2, the EC projects IURO and RADHAR, and NSERC Canada.

References

[1] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[3] T. Deselaers, B. Alexe, and V. Ferrari. Localizing objects while learning their appearance. In *ECCV*, 2010.

[4] R. Fergus, Y. Weiss, and A. Torralba. Semi-supervised learning in gigantic image collections. In *NIPS*, 2009.

[5] V. Ferrari, M. Marin-Jimenez, and A. Zisserman. Progressive search space reduction for human pose estimation. In *CVPR*, 2008.

[6] J. Ferryman and A. Shahrokni. Pets2009: Dataset and challenge. In *IEEE Int. W. on Performance Evaluation of Tracking and Surveillance*, 2009.

[7] J. Gall, A. Yao, N. Razavi, L. van Gool, and V. Lempitsky. Hough forests for object detection, tracking, and action recognition. *PAMI*, 33(11):2188–2202, 2011.

[8] P. Jain and A. Kapoor. Active learning for large multi-class problems. In *CVPR*, 2009.

[9] A. Joshi, F. Porikli, and N. Papanikolopoulos. Breaking the interactive bottleneck in multi-class classification with active selection and binary feedback. In *CVPR*, 2010.

[10] C. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*, 2010.

[11] Y. Lee and K. Grauman. Learning the easy things first: Self-paced visual category discovery. In *CVPR*, 2011.

[12] D. Lewis and W. Gale. A sequential algorithm for training text classifiers. In *SIGIR*, 1994.

[13] S. Oh et al. A large-scale benchmark dataset for event recognition in surveillance video. In *CVPR*, 2011.

[14] D. Parikh and K. Grauman. Relative attributes. In *ICCV*, 2011.

[15] N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. In *ICML*, 2001.

[16] B. Russell, A. Torralba, K. Murphy, and W. Freeman. Labelme: a database and web-based tool for image annotation. *IJCV*, 77(1-3), 2008.

[17] S. Schulter, C. Leistner, P. Roth, H. Bischof, and L. van Gool. On-line hough forests. In *BMVC*, 2011.

[18] B. Settles. From theories to queries: Active learning in practice. In *Active Learning and Experimental Design W.*, pages 1–18. 2011.

[19] A. Sorokin and D. Forsyth. Utility data annotation with amazon mechanical turk. In *W. on Internet Vision*, 2008.

[20] S. Vijayanarasimhan and K. Grauman. Cost-sensitive active visual category learning. *IJCV*, 91(1), 2011.

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

[22] C. Vondrick and D. Ramanan. Video annotation and tracking with active learning. In *NIPS*, 2011.

[23] C. Vondrick, D. Ramanan, and D. Patterson. Efficient video annotation with crowdsourced markets. In *ECCV*, 2010.

[24] P. Welinder and P. Perona. Online crowdsourcing: rating annotators and obtaining cost-effective labels. In *W. on Advancing Computer Vision with Humans in the Loop*, 2010.