Collaborative Track Analysis, Data Cleansing, and Labeling

George Kamberov[1], Gerda Kamberova[2], Matt Burlick[1], Lazaros Karydas[1], and Bart Luczynski[1]

[1] Department of Computer Science
Stevens Institute of Technology
Hoboken, NJ 07030, USA
gkambero, mburlick,lkarydas,bluczyns@stevens.edu
[2] Department of Computer Science
Hofstra University,
Hempstead, NY 11549, USA
Gerda.L.Kamberova@hofstra.edu

Abstract. Tracking output is a very attractive source of labeled data sets that, in turn, could be used to train other systems for tracking, detection, recognition and categorization. In this context, long tracking sequences are of particular importance because they provide richer information, multiple views, wider range of appearances. This paper addresses two obstacles to the use of tracking data for training: noise in the tracking data and the unreliability and slow pace of hand labeling. The paper introduces a criterion for detecting inconsistencies (noise) in large data collections and a method for selecting typical representatives of consistent collections. Those are used to build a pipeline which cleanses the tracking data and employs instantaneous (shotgun) labeling of vast numbers of images. The shotgun labeled data is shown to compare favorably with hand labeled data when used in classification tasks. The framework is collaborative – it involves a human-in-the loop but it is designed to minimize the burden on the human.

1 Introduction

In this paper: (i) We propose a framework for detection and cleansing of tracking output noise due to data association errors and to spurious track births or premature expirations. The framework is based on clustering with respect to a space-time distance and the notions of initial and terminal appearances of a tracked object; (ii) We describe a procedure for labeling the tracking output with minimal human intervention. The procedure uses information-theoretic approach to detect inconsistencies in the tracker output and to extract an iconic representation for each consistent track; and (iii) We apply the framework and the labeling procedure in the design and construction of a pipeline for massive (shotgun) labeling of objects from video-based trackers. The pipeline has three principal stages. The first stage, described in Section 3.1, is designed to correct as much as possible association errors. We use a track segmentation method, based on a space (features)-time metric. The second stage, described in Section 3.2, is designed to address the tracker noise caused by the births of spurious targets produced

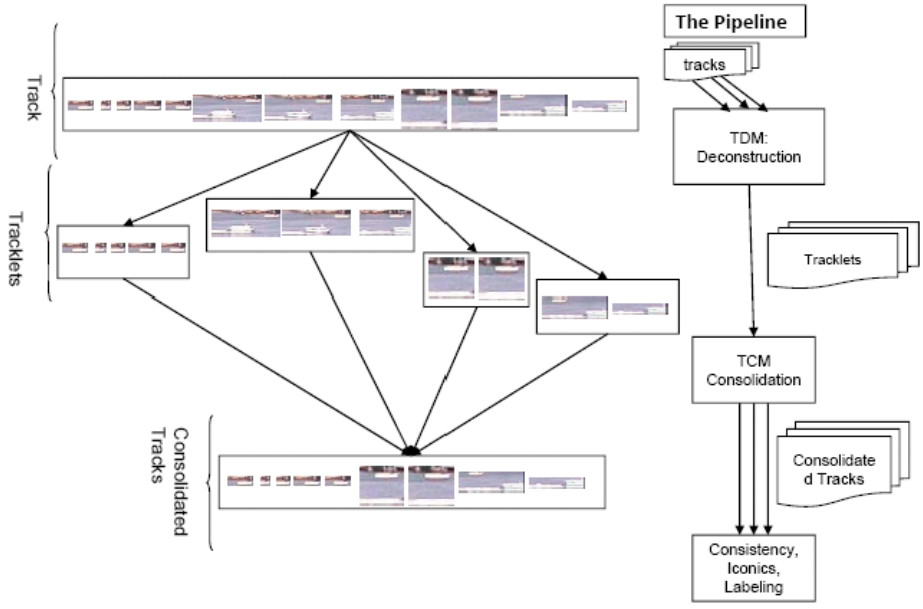

Fig. 1. (Left) Shotgun Pipeline data flow: from raw tracks to consolidated consistent tracks – note that several frames were deleted because there was a miss-match in the final and initial appearances of the constituent tracklets. (Right) A diagram of the Shotgun Pipeline architecture.

when the trackers ”lose their targets” and generate new tracks midstream. This leads to explosion in the number of tracks and thus can have a very detrimental effect on the performance of human-labelers. In the final stage, we test the consistency of the consolidated tracks and then select automatically iconic representatives from the consistent tracks. The iconic representatives are labeled manually and their labels are propagated automatically throughout the consolidated tracks.See Figure 1 for a pipeline overview.

We use tracking data from a three-hour surveillance video of vessels in a busy estuary, captured by a distant on-shore camera, to test the the framework and the pipeline for data cleansing and analysis and target labeling in a cluttered environment The experimental results are reported in Section 4.

1.1 Related Work

There has been a considerable interest in detecting and cleaning mislabeled data for training in the AI and data mining communities, [2], [1], [6], [8], and for face detection and recognition, [17]. These approaches boil down to taking an already labeled data set and applying cross validation of some categorization/recogntion method based on pre-selected set of classifiers or by using case based learning. Semi-supervised learning (SSL) can be used to minimize the tedious hand-labeling task [3, 4, 11, 10, 19] by starting with a small kernel of reliably hand labeled data and a large collection of unlabeled

data. The initial labeled data is used to learn a classifier; the classifier is applied to label part of the unlabeled data; then a new classifier is learned from all the currently labeled data. The process is iterative. The success of an SSL approach depends very much on the selection of: initial training sets, features, and suitable similarity functions and kernels. In contract, we present a method to detect the data categories and to label large sets of data with minimal human intervention. To select the categories represented in the data and assign labels, we develop an automatic method for extraction of a small number of most informative exemplars – called *iconics*. The exemplar-based analysis of data is a staple of active learning (AL). In a typical AL framework, the machine learning algorithm uses a small initial set of labeled data and a dialogue with an expert to select (and label) the data from which it learns and to set by hand appropriate tuning parameters [13, 7, 14, 5, 15, 18, 12]. Our method does not have tuning parameters and does not rely on an initial correctly labeled set.

2 Track Consistency Criterion and Iconics Selection

2.1 Background

For noise detection, track segmentation and selection of iconic representatives, we exploit mutual and disjoint information together with a space-time metric built from the feature metric used by the tracker. The principal idea is to search for noise by measuring when two feature vectors in a sequence of tracking data are more different than alike.

We use the term random variable to denote, both, scalar or vector-valued random variables. Since we work only with discrete variables and finite sample spaces, the sample space could be enumerated, the problem re-mapped, and a uni-variate notation could be used as appropriate. We follow standard conventions: we use P to denote a probability measure; random variables are denoted by capital letters and their values (samples), by lower case letters. For a discrete random variable $X$with a sample space $\mathcal{X}$, the point mass function is denoted by p_X, $p_X(x) = P[X = x], \;\; x \in \mathcal{X}$, and the entropy, $H(X)$ is given by

$$H(X) = -\sum_{x \in \mathcal{X}} p_X(x) \log\, p_X(x).$$

Since we deal only with discrete random variables, without danger of misunderstanding, we use the term "random variable" to mean a "discrete random variable" and "probability distribution" to refer to the point mass function. The joint probability distribution of two random variables, $X$and Y, with sample spaces $\mathcal{X}$and $\mathcal{Y}$, respectively, is defined by, $p_{X,Y}(x, y) = P[X = x, Y = y], \;\; \text{for} \;\; x \in \mathcal{X}, \; y \in \mathcal{Y}$. When there is no confusion, we will omit the subscripts, and thus use $p(x, y)$ instead of $p_{X,Y}(x, y)$.

Next, we summarize the relevant definitions from information theory. Let $X$and Y be two random variables with finite sample spaces $\mathcal{X}$and $\mathcal{Y}$, respectively, and $m = |\mathcal{X}|$ and $n = |\mathcal{Y}|$. We enumerate the elements of the sample spaces,

$$\mathcal{X} = \{x_1, \ldots, x_m\}, \quad \mathcal{Y} = \{y_1, \ldots, y_n\}.$$

The joint Shannon entropy, $H(X,Y)$, the mutual information, $I(X,Y)$, and the disjoint information, $D(X,Y)$, are defined as follows

$$H(X,Y) = -\sum_{i=1}^{m}\sum_{j=1}^{n} p(x_i,y_j)\log\ p(x_i,y_j) \tag{1}$$

$$I(X,Y) = H(X) + H(Y) - H(X,Y), \tag{2}$$
$$D(X,Y) = H(X,Y) - I(X,Y). \tag{3}$$

2.2 Detection of Inconsistent Instances in Data Collections

Given a feature vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, for example an image or a histogram, it is interpreted as a sample of size n of a random variable X, and one can infer the probability distribution p_X from the sample. In the case of two feature vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, by inferring the probability distributions of the corresponding random variables X and Y, one can answer the question whether the collections $\boldsymbol{x}$ and $\boldsymbol{y}$ are more different than they are alike by comparing the mutual and disjoint information of X and Y. When there is no danger of confusion, we say, for short, "the mutual and disjoint information of the feature vectors" and write $D(\boldsymbol{x},\boldsymbol{y})$ and $I(\boldsymbol{x},\boldsymbol{y})$, instead of "the mutual and disjoint information, $D(X,Y)$ and $I(X,Y)$, of the corresponding random variables X and Y.

More generally, we say that the collection $\mathcal{S}$ of feature vectors is *possibly inconsistent*, if there exist feature vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ in $\mathcal{S}$ for which the disjoint information is sufficiently higher than the mutual, i.e.,

$$D(\boldsymbol{x},\boldsymbol{y}) - I(\boldsymbol{x},\boldsymbol{y}) > \alpha,$$

for some threshold $\alpha \geq 0$.

To detect possible class noise (misclassifications) in the collection $\mathcal{S}$, one can attempt to search for appropriate threshold levels. Instead, we exploit a metric d over all possible feature vectors,

$$d : \mathcal{X} \to [0,\infty)$$

and consider the collection inconsistent if a most distant pair of features with respect to the metric is more different than alike in terms of the disjoint/mutual information.

Definition 1: We will say that the collection of feature vectors, $\mathcal{S}$, is potentially inconsistent (contains class noise or misclassifications), if a most distant pair of feature vectors $\widehat{\boldsymbol{x}}$ and $\widehat{\boldsymbol{y}}$ defined by

$$(\widehat{\boldsymbol{x}},\widehat{\boldsymbol{y}}) = \underset{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{S}\times\mathcal{S}}{\arg\max}\ d(\boldsymbol{x},\boldsymbol{y}) \tag{4}$$

satisfy

$$D(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}}) > I(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}}). \tag{5}$$

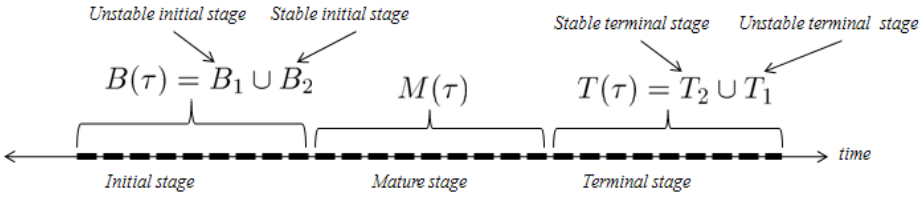

Fig. 2. The evolution of the appearance of an object in a track has three stages: initial, $B(\tau)$, mature, $M(\tau)$, and terminal, $T(\tau)$. Furthermore, the initial and terminal stages are partitioned into into unstable and stable clusters of feature vectors (appearances), respectively.

In particular, a track $\tau = \{\boldsymbol{x}_{t_1}, \ldots, \boldsymbol{x}_{t_n}\}$ is a time stamped collection (a stream) of feature vectors endowed with a natural metric

$$d(\boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_j}) = g_f(\boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_j}) + c^2|t_i - t_j|^2. \tag{6}$$

Here g_f is a metric in feature space and

$$c = \max_{t_i < t_j} \left(\frac{1}{|t_i - t_j|} g_f(\boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_j}) \right)$$

plays the role of the maximum speed of signal propagation in the stream.

We introduce a criterion for track consistency. The criterion may require a human-in-the loop to inspect no more than two frames in the track. Let $\boldsymbol{x}_t$ and $\boldsymbol{x}_{t'}$ be as far as possible in the track τ, i.e.,

$$(\boldsymbol{x}_t, \boldsymbol{x}_{t'}) = \underset{(\boldsymbol{x},\boldsymbol{y}) \in \tau \times \tau}{\arg\max} \; d(\boldsymbol{x}, \boldsymbol{y}) \tag{7}$$

A track τ is *automatically consistent*, if $I(\boldsymbol{x}_t, \boldsymbol{x}_{t'}) > D(\boldsymbol{x}_t, \boldsymbol{x}_{t'})$, where $(\boldsymbol{x}_t, \boldsymbol{x}_{t'})$, satisfies (7). If a track is not automatically consistent, then the final decision whether it is consistent or not is made by a human-in-the-loop. The human inspects only the two images (track frames) corresponding to $\boldsymbol{x}_t$ and $\boldsymbol{x}_{t'}$. Thus *inconsistent* tracks are diagnosed by a human-in-the loop (or another external decision maker) but this intervention is kept to a minimum.

With this criterion at hand, we cleanse tracking data by simply discarding inconsistent tracks.

2.3 Iconic Images and Track Consistency

To label in one shot all the images corresponding to a consistent track , we extract automatically a single *iconic image* (IIM) from each consistent track. The IIM has to be informative; and not overwhelming for the human.

To assure that the representative of a noise free collection is optimal in information theoretic sense, we propose to use as an iconic feature vector (IFV) a feature vector that minimizes the average pairwise disjoint information over all other feature vectors in the in the collection.

Fig. 3. (Left)A sample frame from the surveillance video with two tracked objects. (Right) The tracked objects' background subtraction-based representations; the histograms of these representations are the feature vectors of the objects in this frame.

Definition 2: A feature vector $\tilde{\boldsymbol{x}} \in \mathcal{S}$ is an iconic feature vector of the collection $\mathcal{S}$, if

$$\tilde{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \frac{1}{|\mathcal{S}|} \sum_{\boldsymbol{y} \in \mathcal{S}} D(\boldsymbol{x}, \boldsymbol{y}). \tag{8}$$

2.4 Initial and Terminal Appearance of a Track

We model the evolution of the appearance of an object in a track $\tau = \{\boldsymbol{x}_{t_1}, \ldots, \boldsymbol{x}_{t_n}\}$ as a three-stage process: initial stage $B(\tau) = \{\boldsymbol{x}_{t_1}, \ldots, \boldsymbol{x}_{t_\phi}\}$, a mature(stable)stage $M(\tau) = \{\boldsymbol{x}_{t_{(\phi+1)}}, \ldots, \boldsymbol{x}_{t_{(n-\phi-1)}}\}$, and a terminal stage $T(\tau) = \{\boldsymbol{x}_{t_{(n-\phi)}}, \ldots, \boldsymbol{x}_{t_n}\}$. See Figure 2. We assume that the initial stage and the terminal stage have at most ϕ frames. For tracks obtained from a background subtraction tracker, which uses running averages to estimate the background layer, ϕ equals the size of the history window. When the size of the history window is not known or is not informative, for example difference-based trackers which use single frame differences we set ϕ to a third of the length of the track.

The concepts of stable and unstable initial and terminal clusters are introduced to delineate the relatively unstable appearance of an object soon after its birth and just before its expiration. To determine the initial and the terminal stable and unstable appearance clusters, we use K-medoids, $K = 2$, to cluster the object feature vectors in the track. The metric used in this clustering is the feature space metric $g_f(\cdot, \cdot)$, from equation (6). By definition, the stable appearance cluster in the initial (terminal) stage is the one for which most of the objects are closer (in time-order) to the interior, mature stage $M(\tau)$ of the track. Formally, let B_1 and B_2 be are the two clusters of $B(\tau)$. The relative ordering of B_1 and B_2 is with respect to to their median time stamps, i.e.,

$$\text{med}(\{t | \boldsymbol{x}_t \in B_1\}) \leq \text{med}(\{t \,|\, \boldsymbol{x}_t \in B_2\}). \tag{9}$$

And similarly, let T_1 and T_2 be the two clusters of terminal features then T_1 and T_2 are reverse-ordered with respect to their members median time stamps. Thus,

$$\text{med}(\{t | \boldsymbol{x}_t \in T_2\}) \leq \text{med}(\{t | \boldsymbol{x}_t \in T_1\}). \tag{10}$$

Then, B_2 and T_2 represent the stable appearance initial and terminal clusters of the track.

Table 1. Binary Categorization (recognition success rates out of 10 runs) using hand-labeled training data **HL**: 285 hand-labeled images for training (95 per each of the three categories), and then Shotgun-labeled training data **SL**: 4680 Shotgun labeled features (1560 per category) by examining only 216 images. For the definition of success rate **SR** see equation (13).

Vessel	*Mean of True Positives*		*Std of True Positives*		*Success Rate* **SR**	
	HL	**SL**	**HL**	**SL**	**HL**	**SL**
Speedboat	80.87%	85.25%	11.00%	4.00%	78.75%	81.25%
Ferry	92.25%	96.50%	2.00%	2.00%	91.25%	96.25%
Cruise	84.00%	84.87	6.00%	3.00%	73.75%	81.25%

Definition 3: Let $\mu(B)$ and $\mu(T)$ represent the medoid feature centers for the clusters B_2 and T_2, respectively, the initial and the final appearance feature vectors $S(\tau)$ and $E(\tau)$ are the stable feature vectors closest to these medoids. Namely,

$$S(\tau) = \underset{\boldsymbol{x}_t \in B_2}{\operatorname{argmin}} \left(g_f(\boldsymbol{x}_t, \mu(B)) \right) \tag{11}$$

$$E(\tau) = \underset{\boldsymbol{x}_t \in T_2}{\operatorname{argmin}} \left(g_f(\boldsymbol{x}_t, \mu(T)) \right) \tag{12}$$

3 Shotgun Pipeline

We are now ready to build a pipeline that will be used to label in one sweep the images in multiple tracks. The input to the pipeline, Figure 1, are a collection of tracks $\{\tau_i\}_{i=1}^{\kappa}$ extracted from a surveillance video over some period of time. The pipeline stages are: track segmentation/deconstruction; consolidation; consistency check and iconic extraction and a single shotgun labeling of all the consistent tracks.

3.1 Track Segmentation

We segment each track into tracklets using unsupervised agglomerative clustering and the metric d defined in (6). If the task was to really understand the possible optimal segmentation of the track into consistent tracklets, then one needs to develop an adequate stopping criterion. This is not our goal. Here we are only concerned with creating few relatively long tracks that are consistent. We use gap statistic driven clustering [16] to obtain an initial segmentation of each track into tracklets each of which is more consistent than the original track. At the end of the segmentation stage we obtain, completely automatically, all the resulting tracklets $\{\tau_{i_1}, \ldots, \tau_{n_i}\}_{i=1}^{\kappa}$. We thereat each one of them as a track and extract their initial and terminal feature vectors $\{S(\tau_{i_1}), \ldots, S(\tau_{n_i})\}_{i=1}^{\kappa}$ and $\{T(\tau_{i_1}), \ldots, T(\tau_{n_i})\}_{i=1}^{\kappa}$. This is a crude segmentation, it could result into over-segmentation, and by the very nature of the gap statistic clustering the results could be slightly different for different segmentation runs. Over-segmentation can lead to inefficient labeling. Thus the next stage of the pipeline is devoted to consolidation of the possibly over-segmented tracks.

Table 2. Multi-category confusion matrices using: (Left) Training with hand-labeled data (HL) and (Right) Training with shotgun-labeled data (SL).

	Training on the HL data.			*Training on the SL data.*		
Vessel	Speedboat	Ferry	Cruise	Speedboat	Ferry	Cruise
Speedboat	34	3	4	35	0	6
Ferry	2	38	1	2	39	0
Cruise	14	1	26	10	0	31

3.2 Consolidating trajectories

The input in this stage is the collection of all tracklets. To consolidate together tracklets, we follow the network model and the linear-programming optimization introduced in [9]. The occlusion nodes in the network correspond to the terminal and initial nodes of the different tracklets. The appearance similarity component in the cost function is computed using the feature space metric $g_f(\cdot,\cdot)$ and in particular, if the terminating tracklet τ_{ia} is in occlusion with the new-born tracklet τ_{jb} the appearance component of the metric is $g_f(T(\tau_{ia}), S(\tau_{jb}))$. Here $T(\tau_{ia})$ and $S(\tau_{jb})$ are the corresponding terminal and initial appearance vectors. Furthermore even after possible consolidation we discard the unstable terminal features from the terminating tracklet and the unstable initial segments from the new tracklet.

4 Applications and Experiments

In this section we present the results obtained by applying our framework to: detect errors in tracker output, cleanse tracker output, and label cleansed tracking output so that it can be used for training categorization engines. The experiments were performed on a three hours long surveillance video of shipping traffic in an estuary. Due to the width of the river and the shipping lanes patterns, the camera is far away from the targets (the distances target-to-camera are in the range 380 meters to 1100 meters). We used a background-subtraction based tracker to process the video. Due to the low resolution and the dynamic appearance of the river surface, feature vectors using SIFT or HOGs descriptors are not applicable to this tracking scenario, since many of the tracked objects are no more than four pixels wide. Instead, we opt to use simpler features in the form of histograms. (See Figure 3).

Tracker Errors Detection The tracker detected ninety one tracks (# of feature vectors 24,738). These tracks were tested for consistency using the Consistency Stage of the Shotgun Pipeline. Thirty one tracks (accounting for over 12,364 feature vectors) were found inconsistent (objects of clearly different categories were mixed in the same track). The whole consistency test of the original tracks required that a human-in-the-loop looked a 96 images (two images per each not automatically consistent track, such tracks are automatically flagged).

Table 3. Statistics of the multi-class classification rates over all categories for the hand-labeled and shotgun-labeled experiments. By definition, the classification rate is the percent of correctly labeled data.

Training Data Set	*Mean*	*Std*	*Median*
Hand-labeled (HL)	85.00%	3.40%	85.40%
Shotgun-labeled (SL)	86.30%	2.60%	86.20%

Hand Labeling Data vs Shotgun A team of five students (two PhD students in computer vision and three undergraduate students with no prior experience) hand-labeled, independently, the tracking results. All reported difficulties assigning proper labels due to the low image resolution and clutter. Only a very small number of the targets and the feature vectors were hand-labeled unambiguously and correctly. The whole set was processed by the Shotgun Pipeline. After de-construction and consolidation (fully automatic), a human-in-the loop was asked to examine 105 consolidated tracks. Ultimately 44 tracks were found inconsistent and 94 were found to be consistent. This allowed us to label $10,900$ feature vectors at the total cost of examining 304 images.

Hand Labeling Data vs Shotgun in Recognition and Categorization To test the performance of Shotgun labeled data in recognition and categorization tasks, we created the following data sets of images of objects in the following categories: speed boat, ferry, and cruise ship: (i) **HL**: 285 hand-labeled images for training (95 per each of the three categories); (ii) **Test**: A testing set: 123 hand-labeled feature vectors (41 per each of the three categories); (iii) **SL**: 4680 Shotgun labeled features (1560 per category) by examining a total of 216 images.

The **HL** and **SL** training sets were used to train three neural networks to perform binary categorization tasks (a Speed boat recognition NN, a Ferry recognition NN, and a Cruise ship recognition NN). The results using the hand labeled set **HL** and the Shotgun -labeled training set **SL** are shown in Table 1. We define the success rate, **SR**, as:

$$\mathbf{SR} = \frac{\text{True Positives} + \text{True Negatives}}{\text{True Positives} + \text{True Negatives} + \text{False Positives} + \text{False Negatives}}. \quad (13)$$

To test the performance in a multi category scenario we trained another NN. The results are shown in Tables 2 and 3.

The results indicate that we can replace massive hand-labeling with the less demanding Shotgun labeling while improving the quality of categorization. Our future work centers on the improvement of the selection of iconic images and on developing methods to exploit inconsistent video streams data as negative response training data.

References

1. D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. In *Machine Learning*, pages 37–66, 1991.

2. C. E. Brodley and M. A. Friedl. Identifying mislabeled training data. *Jour. Artificial Intelligence Res.*, 11:131–167, 1999.
3. O. Chapelle, A. Zien, and B. Schölkopf, editors. *Semi-Supervised Learning*. MIT Press., 2006.
4. H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. In *Proc. 10th European Conference on Computer Vision*, pages 234–247, 2008.
5. E. G. Hauptmann, W. hao Lin, R. Yan, J. Yang, and M. yu Chen. Extreme video retrieval: joint maximization of human and computer performance. In *ACM Multimedia*, pages 385–394. ACM Press, 2006.
6. D. He, X. Zhu, and X. Wu. Error detection and uncertainty modeling for imprecise data. In *Proc. 21st International Conference on Tools with Artificial Intelligence*, pages 792 –795, 2009.
7. A. Hoogs, J. Rittscher, G. Stein, and J. Schmiederer. Video content annotation using visual analysis and a large semantic knowledgebase. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages II: 327–334, 2003.
8. M. Jacob, A. Kuscher, M. Plauth, and C. Thiele. Automated data augmentation services using text mining, data cleansing and web crawling techniques. In *Proc. IEEE Congress on Services - Part I,*, pages 136 –143, 2008.
9. H. Jiang, S. Fels, and J. Little. A linear programming approach for multiple object tracking. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.
10. R. Lefort, R. Fablet, and J.-M. Boucher. Weakly supervised classification of objects in images using soft random forests. In *Proc. 11th European Conference on Computer Vision*, pages 185–198, 2010.
11. C. Leistner, H. Grabner, and H. Bischof. Semi-supervised boosting using visual similarity learning. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.
12. H. Lin and J. Bilmes. How to select a good training-data subset for transcription: Submodular active selection for sequences. In *Proc. 10th Annual Conference of the International Speech Communication Association*, 2009.
13. I. M. Muslea, S. Minton, and C. A. Knoblock. Active + semi-supervised learning = robust multi-view learning. In *Proc. 19th International Conference on Machine Learning*, pages 435–442, 2002.
14. H. T. Nguyen and A. Smeulders. Active learning using pre-clustering. In *Proc. the 21st International Conference on Machine Learning*, pages 623–630. ACM Press, 2004.
15. G.-J. Qi, X.-S. Hua, Y. Rui, J. Tang, and H.-J. Zhang. Two-dimensional active learning for image classification. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 1 –8, 2008.
16. R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.
17. S. Venkataraman, D. Metaxas, D. Fradkin, C. Kulikowski, and I. Muchnik. Distinguishing mislabeled data from correctly labeled data in classifier design. In *Proc. 16th IEEE Int. Conf. on Tools With Artificial Intelligence*, pages 668–672, 2004.
18. S. Vijayanarasimhan and K. Grauman. What's it going to cost you?: Predicting effort vs. informativeness for multi-label image annotations. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 2262 –2269, 2009.
19. R. Yan and M. Naphade. Semi-supervised cross feature learning for semantic concept detection in video. In *Proc. IEEE Computer Vision and Pattern Recognition*, pages 657–663, 2005.