Tracking-Based Semi-Supervised Learning

Alex Teichman, Sebastian Thrun
Stanford University
Department of Computer Science
{teichman,thrun}@stanford.edu

***Abstract*—In this paper, we consider a semi-supervised approach to the problem of track classification in dense 3D range data. This problem involves the classification of objects that have been segmented and tracked without the use of a class-specific tracker.**

We propose a method based on the EM algorithm: iteratively 1) train a classifier, and 2) extract useful training examples from unlabeled data by exploiting tracking information. We evaluate our method on a large multiclass problem in dense LIDAR data collected from natural street scenes. When given only three hand-labeled training tracks of each object class, the final accuracy of the semi-supervised algorithm is comparable to that of the fully-supervised equivalent which uses two orders of magnitude more.

Finally, we show that a simple algorithmic speedup based on incrementally updating a boosting classifier can reduce learning time by a factor of three.

I. Introduction

Currently, object recognition capabilities are a fundamental limitation of many practical robotic systems. To learn about new objects from non-robotics-experts, we need new learning methods which do not require large, hand-labeled training sets. We aim at making progress towards this goal.

In this paper, we consider a decomposition of the object recognition problem into segmentation, tracking, and track classification components. In particular, we consider *model-free* segmentation and tracking, *i.e.* that which works without a class-specific tracking or segmentation model. A solution to the track classification problem would reduce object recognition to a segmentation and tracking problem. There are many ways to think about the object recognition problem, but, as we will show in this paper, this decomposition enables us to greatly reduce the burden of the training process via semi-supervised learning.

In general, model-free segmentation and tracking is hard to come by. However, depth information such as that provided by a LIDAR sensor or stereo camera with projected texture can provide useful depth segmentations in some contexts. We focus our experiments on the autonomous driving scenario, where depth segmentations in LIDAR data are frequently correct because the objects of interest actively avoid collision with the environment. We expect the results in this paper to be useful in any situation in which model-free segmentation and tracking are available, and perhaps can be built upon to develop methods for those cases in which they are not.

The approach we take is based on the distinction between track classification, *i.e.* classifying a sequence of segmented LIDAR point clouds of an object recorded over time, and frame classification, *i.e.* classifying a single segmented LIDAR point cloud of an object at a single point in time. The process iterates between two steps: first, a frame classifier is trained on a given training set, and second, track classification is run on an unlabeled set to find new training examples. Because the track classifier uses the frame classifier as a component, improvements in the frame classifier result in better track predictions on the next iteration.

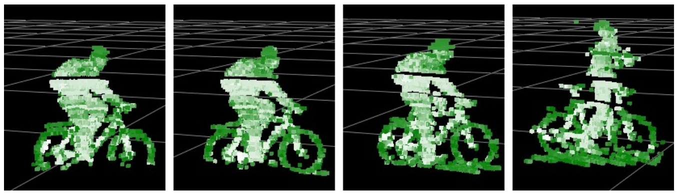


Fig. 1: A machine learning classifier which correctly recognizes several frames of a track of a bicyclist can infer that the remaining frames also are of a bicyclist. This enables the addition of new, useful training examples that include changes in pose (as above), occlusion level, and viewing distance.

A naïve implementation of this which ignores the tracking information makes little progress, as the training examples that get added - the ones the frame classifier is most confident about - are by definition not those that would be most useful in the training process. Using tracking information, a machine learning algorithm can automatically find new, useful training examples that improve classifier accuracy. See Figure 1 for an example.

This approach may seem inapplicable to rigid object classes such as cars, but, perhaps surprisingly, turns out to be effective here as well. This is likely due in part to occlusion: for example, two different vehicles might look very similar when only the front of them is seen, allowing the learning algorithm to propagate a label from one to the other. Our experiments bear out the hypothesis that the proposed method works for rigid object classes such as cars.

Tracking errors, such as when an object is temporarily segmented together with some other object or when the tracker jumps from one object to another, are a strong potential cause of failure in tracking-based semi-supervised learning. In practice, they are often difficult to avoid. However, we show empirically that, while inconsistent tracks certainly exist in our unlabeled dataset, the proposed method is able to learn effectively.

We use a variant of boosting in our experiments that makes it easy to work with multiple high-dimensional descriptor spaces and allows for intuitive incremental updating of the classifier. However, we expect the general message of tracking-based semi-supervised learning to apply to many different classifier types whenever model-free segmentation and tracking is available.

The primary contribution of this paper is to show that tracking-based semi-supervised learning is an effective method of training object recognition systems with a very small amount of hand-labeled data. In a multiclass track classification experiment on real-word data collected from unstructured, unstaged environments, we show that three hand-labeled training examples of each class can be used to train a classifier that performs comparably to the fully-supervised equivalent. Additionally, we develop an extension to the basic algorithm which uses incremental training of a boosting classifier to increase the speed of the learning process by a factor of three.

Related work is deferred until after the discussion of our approach.

Algorithm 1 Tracking-based semi-supervised learning

```
$\tau$ is a confidence threshold chosen by hand
$\mathbb{S}$ is a small set of seed tracks, labeled by hand
$\mathbb{U}$ is a large set of unlabeled tracks
$\mathbb{B}$ is a large set of background tracks

$\mathbb{W} := \mathbb{S} \cup \mathbb{B}$
repeat
  Train frame classifier $\mathbb{C}$ on $\mathbb{W}$
  $\mathbb{W} := \mathbb{S} \cup \mathbb{B}$
  for $u \in \mathbb{U}$ do
    Classify track $u$ using $\mathbb{C}$
    $c := \text{confidence}(u)$
    $l := \text{classification}(u)$
    if $c \geq \tau$ and $l \neq$ "background" then
      Add $u$ to $\mathbb{W}$ with label $l$
    end if
  end for
until converged
```

II. Algorithm details

A. Tracking-based semi-supervised learning

The distinction between *track classification*, in which a prediction is made for an object tracked over time, and *frame classification*, in which a prediction is made for an object at a single point in time, is essential to our approach. Track classifications are generated by combining the outputs of the frame classifier across an entire track. A confident classification of a track allows many individual frames, some which the frame classifier may currently get wrong, to be added to the training set.

Tracking-based semi-supervised learning is initialized with a small set of hand-labeled seed tracks and a large set of background tracks. Fortunately, labeled background objects are often freely available - in our case, by collecting tracks of objects in areas known to have no pedestrians, bicyclists, or cars in them and automatically labeling them all as background.

The semi-supervised learning procedure, which bears a strong resemblance to hard-EM, is outlined in Algorithm 1. Hard decisions, *i.e.* assigning a label rather than a probability distribution, are made about each unlabeled track at each iteration of the algorithm, but they are remade each time. Only non-background classes are inducted into the working set $\mathbb{W}$ because, in the initial stages, nearly everything in the unlabeled set will be confidently classified as background. Initially, the classifier will recognize only those tracks that look extremely similar to the seed tracks. As the algorithm proceeds, its knowledge of what each class looks like will slowly spread to more distant (in terms of the descriptor space) examples. The procedure converges when the number of inducted tracks levels off.

B. Frame classification

We now consider a particular implementation of Algorithm 1. We note that the general method is likely effective for many specific implementations. This section is a brief summary of the method; more details can be found in [22].

The frame classifier $\mathbb{C}$ of Algorithm 1 is implemented by a boosting classifier. Boosting is a procedure for combining the predictions of many relatively inaccurate *weak classifiers* into a single higher-accuracy prediction. In our case, the weak classifiers encode simple rules about the various frame descriptors, such as "a very wide object is unlikely to be a pedestrian" or "an object that looks like this one from the top is likely to be a car".

We use a variant of GentleBoost [7] and JointBoost [23] for frame classification. As in JointBoost, the weak classifiers are shared across the three 1-vs-all classification problems; unlike JointBoost, since we do not consider problems with extremely large numbers of classes, we force all weak classifiers make predictions for all 1-vs-all classification problems. To simplify the notation, the following discussion will consider only single class problems. The extension to multiclass is straightforward. In the following, each training example is a pair (y_m, z_m) where $y_m \in \{-1, 1\}$ is the class label and z_m is a set of descriptors.

The set z_m includes a total of 29 descriptors which encode various aspects of object appearance: oriented bounding box size, 4 different parameterizations of spin images [8], and 24 different parameterizations of the histogram of oriented gradients [5] descriptor computed on virtual orthographic camera intensity images [22].

Boosting provides a stagewise additive solution to the optimization problem

$$\underset{H}{\text{minimize}} \quad \frac{1}{M} \sum_m \exp\left(-y_m H(z_m)\right). \tag{1}$$

The strong classifier H is defined as a sum of the weak

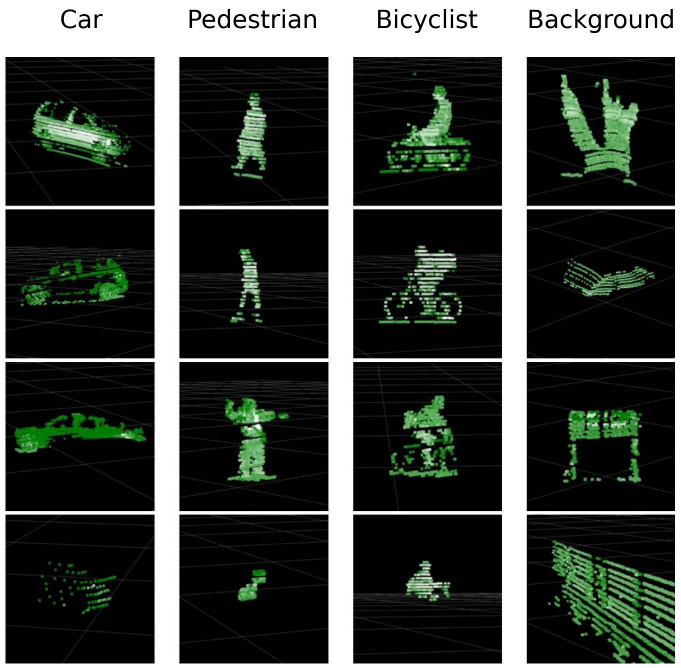


Fig. 2: Example segmented objects from the dataset.

classifiers h_k for all $k \in \{1, 2, \ldots, K\}$, *i.e.*

$$H(z) = \sum_k h_k(z).$$

Our weak classifiers take the form

$$h_k(z) = \begin{cases} a_k & \text{if } ||f_k(z) - x_k||_2 < \theta_k \\ 0 & \text{otherwise} \end{cases},$$

where $f_k(z)$ chooses a particular descriptor in the set z, and x_k is a descriptor in the same space as $f_k(z)$.

Geometrically, this means that a weak classifier makes a prediction a_k about a descriptor $f_k(z)$ if the descriptor falls within a ball of radius θ_k centered at the point x_k. The response value a_k is positive if the weak classifier predicts the descriptor to be of class 1, and negative otherwise; $|a_k|$ is the confidence of the prediction.

Classifier training proceeds as normal in boosting. Learning of weak classifiers ends when the objective function (1) value crosses a threshold.

C. Track classification

Track classifications are computed by applying a normalized discrete Bayes filter to the frame classification outputs. We desire an estimate of $\mathscr{L}(z_{1:T}) = \log \frac{\mathbb{P}(Y=1|z_{1:T})}{\mathbb{P}(Y=-1|z_{1:T})}$, the log odds given all the information known about a track. Expanding this term out using Bayes' rule, a conditional independence assumption, and another application of Bayes' rule, we have

$$\begin{aligned}\mathscr{L}(z_{1:T}) &= \log \frac{\mathbb{P}(Y=1|z_{1:T})}{\mathbb{P}(Y=-1|z_{1:T})} \\ &= \mathscr{L}_0 + \log \frac{\mathbb{P}(z_{1:T}|Y=1)}{\mathbb{P}(z_{1:T}|Y=-1)} \\ &= \mathscr{L}_0 + \sum_{t=1}^{T} \log \frac{\mathbb{P}(z_t|Y=1)}{\mathbb{P}(z_t|Y=-1)} \\ &= \mathscr{L}_0 + \sum_{t=1}^{T} \left(\log \frac{\mathbb{P}(Y=1|z_t)}{\mathbb{P}(Y=-1|z_t)} - \mathscr{L}_0 \right) \\ &\approx \widetilde{\mathscr{L}_0} + \sum_{t=1}^{T} \left(H^{\text{F}}(z_t) - \widetilde{\mathscr{L}_0} \right) \qquad (2)\end{aligned}$$

where $\mathscr{L}_0$ is the log prior odds $\log \frac{\mathbb{P}(Y=1)}{\mathbb{P}(Y=-1)}$, $\widetilde{\mathscr{L}_0}$ is its empirical estimate from the training set, and H^{F} denotes the frame classifier. Boosting classifiers naturally output an estimate of $\frac{1}{2} \log \frac{\mathbb{P}(Y=1|z_{1:T})}{\mathbb{P}(Y=-1|z_{1:T})}$ due to the exponential loss function, so we assume here that the frame classifier outputs are adjusted appropriately.

However, in (2), $\widetilde{\mathscr{L}_0}$ gets overwhelmed as track size increases. This is undesirable in practice and reflects the convenient but incorrect assumption that consecutive frames in a track are conditionally independent given their label. As in [22], we instead use the normalized discrete Bayes filter

$$\widetilde{\mathscr{L}_0} + \frac{1}{T} \sum_{t=1}^{T} \left(H^{\text{F}}(z_t) - \widetilde{\mathscr{L}_0} \right)$$

to compute track classifications.

Number of tracks

Set	Car	Pedestrian	Bicyclist	Background	All
Training	904	205	187	6585	7881
Testing	847	112	140	4936	6035
Total	1751	317	327	11521	13916

Number of frames

Set	Car	Pedestrian	Bicyclist	Background	All
Training	92255	32281	31165	532760	688461
Testing	59173	22203	25410	530917	637703
Total	151428	54484	56575	1063677	1326164

TABLE I: Breakdown of the dataset by class. Tracks were collected from busy streets and intersections. In the experiments, the fully-supervised baseline was given all the labels in the training set; the semi-supervised method of this paper was given three hand-labeled training examples of each class, approximately 3000 automatically-labeled background objects, and the remainder of the training set without labels.

III. Experimental results

A. Dataset

We evaluate our method using the Stanford Track Collection [21], a dataset of about 13,000 tracks (a total of about 1.3 million frames) extracted from natural, unstaged street scenes with a dense LIDAR system mounted on a car. Data was recorded while driving and while parked at busy intersections with many people, bicyclists, and cars. See Figure 2 for some

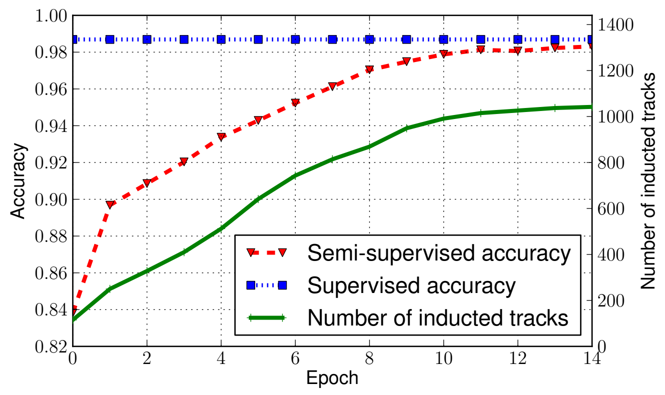


Fig. 3: Test set performance of the semi-supervised method approaches fully-supervised performance. Each epoch is one iteration of the main loop in Algorithm 1. We report results on the natural distribution of object classes found in the real world; classifying everything as background results in 81.8% correct. See Figure 4 for per-class performance details.

examples of what objects look like in such data. Motion of the car was offset using a high accuracy GPS/IMU system.

The dataset is split approximately in half into geographically separate training and testing sets. Details on dataset size can be found in Table I. The baseline supervised method is given the training set with all labels; semi-supervised methods are given three hand-labeled training examples of each class ($\mathbb{S}$ from Algorithm 1), approximately 3000 automatically-labeled background objects ($\mathbb{B}$), and the remainder of the training set without labels ($\mathbb{U}$). Segmentation and tracking problems are out of the scope of this paper, so quantitative test results consider only *consistent* tracks; otherwise errors due to bad segmentation or tracking would confound the errors made by the track classifier.

Importantly, the set $\mathbb{U}$ contains *all* tracks that were extracted from the environment, including those that have tracking and segmentation errors. This means the semi-supervised methods discussed in this paper must be robust to some inconsistent tracks, as would be required in the practical application of a system built on them. Inconsistent tracks are typically due to tracking errors and transient segmentation errors. Such inconsistent tracks make up about 11% of the cars, pedestrians, and bicyclists in the dataset.

Track extraction details can be found in [22]; they are not critical to the results presented here. The extraction process is essentially depth segmentation followed by Kalman filter tracking of the extracted segments.

In all experiments, the induction confidence threshold τ was set to 5 and the objective function threshold value for training the frame classifier was set to 0.02, with a minimum of 1000 weak classifiers.

B. Comparison with fully-supervised baseline

Figure 3 shows the results of running the semi-supervised learning method of Algorithm 1 with three hand-labeled

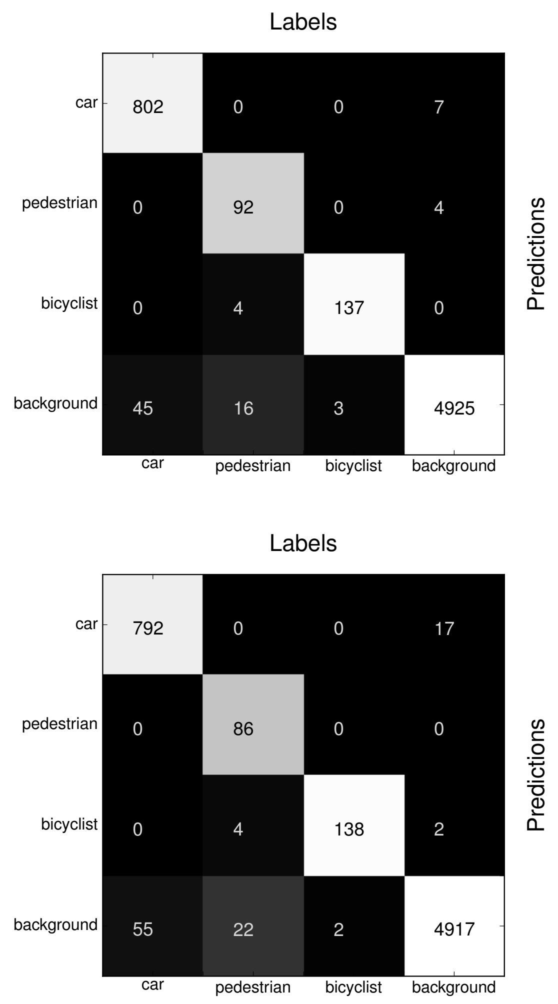
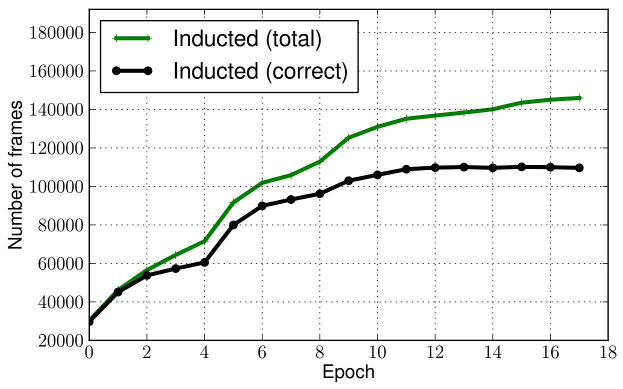


Fig. 4: The confusion matrix produced by the fully-supervised baseline (top) is closely matched by that of the semi-supervised method (bottom). Quantities in each cell are numbers of test set tracks. Final accuracies of the methods are 98.7% and 98.3%, respectively.

training examples of each class; this produces final test-set accuracy comparable to that of the fully-supervised equivalent. Confusion matrices for the two cases are shown in Figure 4. To ensure a fair comparison, the baseline classifier was allowed the same number of weak classifiers as the maximum amount used by the semi-supervised method.

C. Comparison with non-tracking method

To confirm that the tracking information is essential to this approach, we tested the obvious alternative of running a semi-supervised method the same as that in Algorithm 1, but which inducts frames rather than entire tracks. Induction plots for the two methods are shown in Figure 5. Without using tracking information, the semi-supervised learning algorithm can only add new training examples which the classifier is already confident about. As it adds training examples closer to the boundaries between classes, errors are introduced and



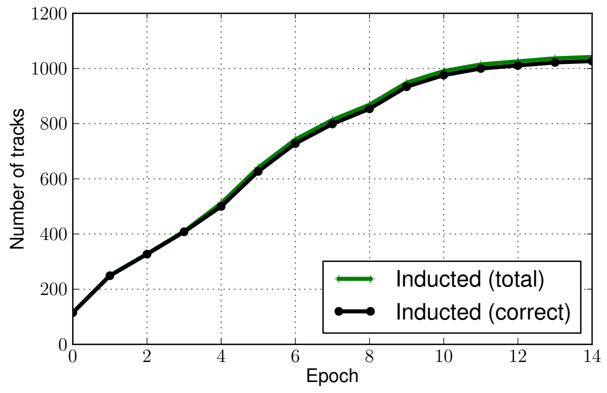


Fig. 5: Ignoring tracking information (top) results in the learning algorithm adding training examples into its training set with incorrect labels. Using tracking information (bottom), these errors are largely avoided.

the accuracy of the inducted frames drops. The final test-set accuracy of the non-tracking method is 93.5%, compared to 98.3% when using tracking information.

D. Accuracy as a function of amount of unlabeled data

To analyze the effect of the amount of unlabeled data on the final semi-supervised performance, we varied the amount of unlabeled data provided, ran the algorithm to completion, and recorded the final accuracy. Results are shown in Figure 6. In this experiment, we provided about 4000 additional unlabeled tracks to the semi-supervised method beyond those used in Section III-B and show that the semi-supervised method, with three hand-labeled tracks of each object class, can equal or outperform the fully-supervised one.

Two major sources of error must be considered in an analysis of this sort. First, individual log files used for unlabeled data can have widely disproportionate numbers of objects from each class. Second, it is not uncommon for there to exist several separate tracks that correspond to the same real-world object that become fully occluded and then reappear. Randomizing the order of tracks presented could incorrectly make it appear that additional data fails to increase performance when it is actually because this additional data is

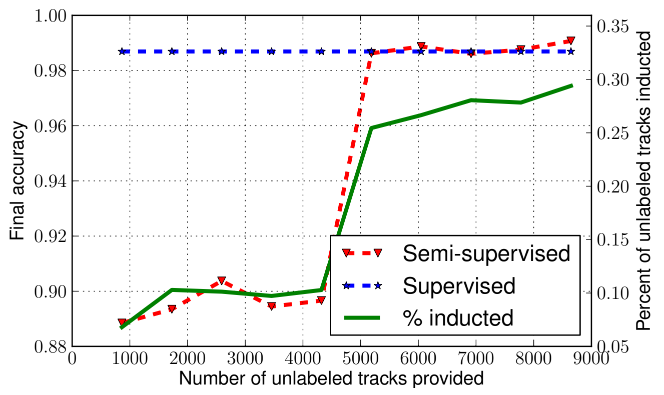


Fig. 6: Tracking-based semi-supervised learning requires a certain quantity of unlabeled data without which only little progress can be made. Intuitively, the learning algorithm can't bridge the gap in descriptor space between two very different examples of the same class without a sufficient quantity of objects connecting the two.

nearly identical to data already seen. To address these sources of error, the order of the unlabeled tracks was not randomized and tracks of each class were added in proportion to their representation in the full unlabeled set. As before, the baseline classifier was allowed the same number of weak classifiers as the maximum amount used by the semi-supervised method.

E. Speedup with incremental training

The semi-supervised learning run of Figure 3 took approximately 25 hours to complete on a modern 12-core desktop. About 7% to 1% (depending on how far in the learning process the algorithm has gotten) of the time taken for each epoch is spent searching for new training examples in the unlabeled data; the rest is spent training the classifier.

To improve this, we consider incremental training of a single boosting classifier rather than retraining from scratch each epoch. When given a new training set, we wish to first relearn the response values of all existing weak classifiers, then continue to learn new weak classifiers until the objective function reaches the stopping threshold. We refer to this method as CSSL-RR, for continuous semi-supervised learning with response relearning. This procedure is described in Algorithm 2.

The naïve alternative - to resume training of a previous boosting classifier without relearning the response values of the weak classifiers - fails because many new weak classifiers must be added to compensate for the mistakes that old weak classifiers have made. In this case, the number of weak classifiers rises quickly, reaching 12,636 by epoch four, compared to only 2,477 in the version of Algorithm 2, and this required additional training makes the naïve version too slow to be of use. See Figure 7 for a comparison of the three methods. The final accuracy of the accelerated version was unchanged.

Algorithm 2 Incremental training of a boosting classifier

```
Recompute the log prior odds $\widetilde{\mathscr{L}}_0 \approx \log \frac{\mathbb{P}(Y=1)}{\mathbb{P}(Y=-1)}$
Reset the training example weights to be $\exp(-y_m \mathscr{L}_0)$
for $k \in \{1 \dots K\}$ do
    Compute $M' = \{m : ||f_k(z_m) - x_k||_2 < \theta_k\}$
    $a_k := (\sum_{m \in M'} w_m y_m) / (\sum_{m \in M'} w_m)$
    for $m \in M'$ do
        $w_m := w_m \exp(-y_m a_k)$
    end for
end for
Learn new weak classifiers until converged
```

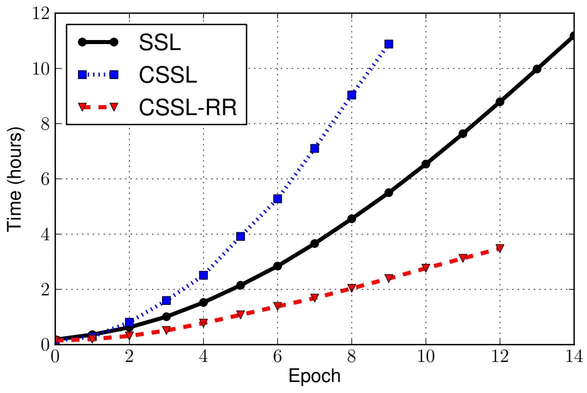


Fig. 7: The naïve method of incrementally training the classifier (CSSL) is substantially slower than retraining from scratch each iteration (SSL). Many new weak classifiers must be learned to correct the mistakes of previous weak classifiers. Using the procedure of Algorithm 2, *i.e.* relearning the weak classifier response values before resuming the training procedure, however, provides a substantial increase in speed (CSSL-RR).

F. Qualitative analysis

To obtain a qualitative understanding of how our method performs, we drove the sensor platform on previously-unseen streets, extracted all tracks in the environment, classified those tracks using the semi-supervised learning algorithm described in Section III-E, and projected the classification of those tracks into aligned camera images of the area. Some examples of the output are shown in Figure 8. We note that this is an example of the full object recognition problem, whereas this paper primarily addresses only the track classification subproblem. Errors in tracking and segmentation result in object recognition errors that are not reflected in the track classification results.

A video of these results can be seen at [20].

IV. Related Work

Machine learning methods can be broken down into two broad categories: inductive methods, which learn a prediction rule, and transductive methods, which make a prediction for each example in a given dataset but can not (easily) produce a prediction for an example outside the given dataset. Many semi-supervised methods fall into the transductive category. As we are interested in robotics applications in which predictions on never-before-seen examples must be made in a timely manner, we are most focused on inductive semi-supervised methods.

Many methods of reducing the burden of labeling have been explored as labeling is often a major task in the building of intelligent systems. Here, we briefly survey the broad categories of these methods and mention a few specific examples that are closely related to the work of this paper. Zhu [26] provides a good survey of semi-supervised methods.

In transfer learning [15], data from one distribution is used to improve the performance of a system on a different distribution. In a recent example from the robotics literature, Lai and Fox [11, 10] show that it is possible to use 3D models from the Google 3D Warehouse to improve object recognition in depth data.

Co-training [2] assumes that two conditionally independent views of the same underlying data are available. Alternating training and classification on the two different views allows the learning algorithm to leverage large amounts of unlabeled data. Perhaps surprisingly, co-training in the robotics literature is scarce. In computer vision, Christoudias et al. [3] proposed a variant of co-training that can improve multi-view object recognition and audio-visual gesture recognition.

Self-supervised methods exploit the presence of a reliable automatic labeling source inherent in the system to train a classifier on a different data modality. For example, Dahlkamp et al. [4] trained a vision-based road detection system for autonomous driving on labels provided by a laser range finder, significantly extending the range of road detection while requiring no extra human supervision. Lookingbill et al. [13] employed reverse optical flow and close range sensors to automatically learn about the visual appearance of distant obstacles. More recently, Wurm et al. [25] used a measure of vibration as a source of self-supervision to train a laser-based vegetation detector.

Unsupervised clustering methods can ease the labeling task by grouping unlabeled examples. Triebel and Siegwart [24] locate similar objects in dense 3D point clouds. Luber et al. [14] track moving objects in line scanner data and perform unsupervised clustering on the tracks.

Graph-based semi-supervised learning methods [27] construct a graph of labeled and unlabeled training examples where edges encode the similarity of examples, typically distance between examples in the descriptor space. These methods have several appealing interpretations and are easy to implement. Basic implementations, however, are inherently transductive and scale cubically with the number of unlabeled examples, though some progress has been made in improving this, *e.g.* in the work of Liu et al. [12].

Other methods of generating useful label propagation between distant regions of the descriptor space also exist. For example, Socher and Fei-Fei [16] use information from newspaper articles to inform image classification.

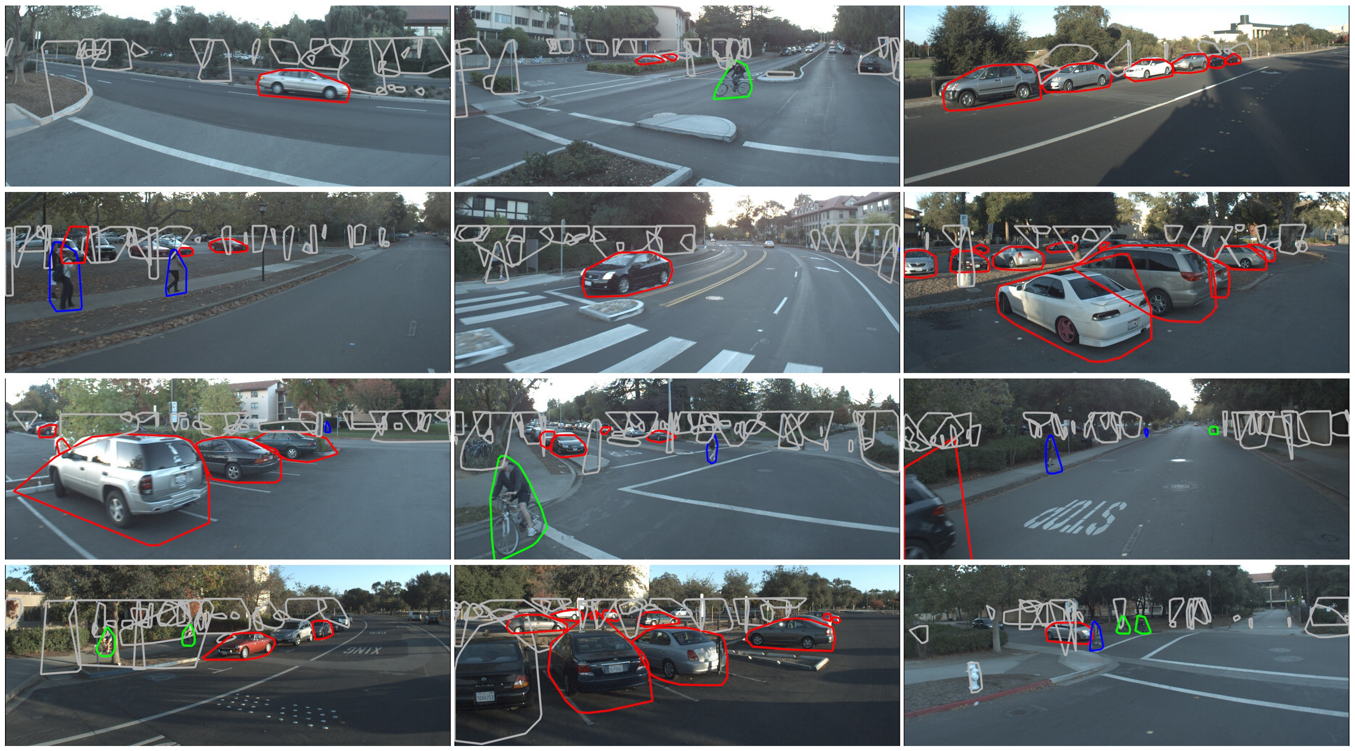


Fig. 8: Track classifications in LIDAR data projected into aligned camera data for visualization. Best viewed in color. Cars are outlined in red, pedestrians in blue, bicyclists in green, and background in gray. Objects without outlines were not tracked. Tracking and segmentation errors (beyond the scope of this paper) are the major cause of failures. A video of these results can be seen at [20].

Ali et al. [1] demonstrate a system that could be considered the pure-vision analogue of our method, known as FlowBoost. Starting with a sparse labeling (every 64th frame, for example) tracking information and a boosting classifier are iteratively applied to fill in the gaps. This method avoids the difficulty of model-free tracking in video data, but at the cost of being unable to process completely-unlabeled video sequences.

It is worth mentioning the related topic of discriminative tracking, *e.g.* [9, 18, 19], in which semi-supervised methods are sometimes used to address the model-free tracking problem. In contrast, we assume model-free segmentation and tracking is given, and a semi-supervised method is used to address the track classification problem. As one example of a discriminative tracker, Kalal et al. [9] use a Lucas-Kanade tracker to provide their discriminative tracker with image patches that are known to be either the tracked object or background.

Independent of the level of supervision, object recognition using depth sensors has become a promising area of research. Douillard et al. [6] use semi-supervised training of a CRF to address semantic mapping of urban environments - that is, point-wise classification into classes such as car, people, foliage, grass, wall, etc. In contrast, the recent work of Spinello et al. [17] considers track classification, as we do here, though they use a fully-supervised approach. Additionally, they track detections of objects made by classifiers, whereas we track all objects provided by depth segmentation and classify the resulting tracks.

V. Conclusion

In this paper, we have shown that tracking-based semi-supervised learning given only three hand-labeled training examples of each class can perform comparably to equivalent fully-supervised methods. We also show that a simple algorithmic speedup based on incremental training of boosting classifiers can increase the efficiency of learning by a factor of about three.

Given the relatively high reliability of track classification and ease of training, the primary performance bottleneck in object recognition systems such as the one discussed in Section III-F is tracking and segmentation. This is encouraging. Using the methods of this paper, the cost of scaling up an object recognition system (in terms of trained human time) could be very much lower than previously possible.

However, no method is without tradeoffs. This system and other similar ones based on track classification are entirely dependent on some relatively reliable method of segmenting potential objects. In our case, this role is filled by depth segmentations available due to range finders and the cooperation of the environment. In many other scenarios, such as object recognition in cluttered indoor environments or using exclusively cameras, model-free segmentation is not so freely available. It is an open question as to whether methods such as the one proposed in this paper can be adapted to such a scenario, but given the effectiveness of the method it seems worthy of investigation.

Unfortunately, there is no guarantee that this method will not diverge because of tracking and segmentation errors or simply because some distant or highly occluded objects are indistinguishable from background. In practice, this appears to be held in check by the large quantity of automatically labeled background data that is provided. Further difficulties may be encountered if one desired finer class distinctions, for example between sedans and trucks; evidently, the labels are able to propagate from one to the other, likely due to occlusions. In this case, we suspect that providing sufficient seed labels at the boundaries will prevent leakage of label propagation from one side to the other, though this has not yet been tested.

A further challenge along these lines is the memory constraint: the boosting algorithm used here requires all training examples to be loaded into main memory before learning. To consider larger unlabeled datasets containing days or weeks worth of data, it will probably be necessary to use an online learning algorithm.

Acknowledgments

We are grateful to Mike Sokolsky for maintaining the autonomous vehicle research platform and to Christian Plagemann for helpful comments.

References

[1] Karim Ali, David Hasler, and Francois Fleuret. FlowBoost - Appearance Learning from Sparsely Annotated Video. In *Computer Vision and Pattern Recognition*, 2011.

[2] Avrim Blum and Tom Mitchell. Combining Labeled and Unlabeled Data with Co-Training. *Computational Learning Theory*, pages 92–100, 1998.

[3] C. Mario Christoudias, Raquel Urtasun, Ashish Kapoor, and Trevor Darrell. Co-training with noisy perceptual observations. *Computer Vision and Pattern Recognition*, pages 2844–2851, June 2009.

[4] Hendrik Dahlkamp, Adrian Kaehler, David Stavens, Sebastian Thrun, and Gary Bradski. Self-supervised Monocular Road Detection in Desert Terrain. In *Robotics: Science and Systems*, 2006.

[5] Navneet Dalal and Bill Triggs. Histograms of Oriented Gradients for Human Detection. In *Computer Vision and Pattern Recognition*, volume 1, pages 886–893. IEEE, 2005.

[6] B. Douillard, D. Fox, F. Ramos, and H. Durrant-Whyte. Classification and Semantic Mapping of Urban Environments. *International Journal of Robotics Research*, 2010.

[7] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Special Invited Paper. Additive Logistic Regression: A Statistical View of Boosting. *The Annals of Statistics*, 28(2):337 – 374, 2000.

[8] A.E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433–449, 1999.

[9] Zdenek Kalal, Jiri Matas, and Krystian Mikolajczyk. P-N Learning: Bootstrapping Binary Classifiers by Structural Constraints. In *Conference on Computer Vision and Pattern Recognition*, 2010.

[10] K. Lai and D. Fox. 3D Laser Scan Classification Using Web Data and Domain Adaptation. In *Robotics: Science and Systems*, 2009.

[11] K. Lai and D. Fox. Object Recognition in 3D Point Clouds Using Web Data and Domain Adaptation. *The International Journal of Robotics Research*, 29(8):1019–1037, May 2010.

[12] Wei Liu, Junfeng He, and Shih-fu Chang. Large Graph Construction for Scalable Semi-Supervised Learning. In *International Conference on Machine Learning*, 2010.

[13] A. Lookingbill, J. Rogers, D. Lieb, J. Curry, and S. Thrun. Reverse Optical Flow for Self-Supervised Adaptive Autonomous Robot Navigation. *International Journal of Computer Vision*, 74(3):287–302, January 2007.

[14] Matthias Luber, Kai O. Arras, Christian Plagemann, and Wolfram Burgard. Classifying dynamic objects: An unsupervised learning approach. In *Robotics: Science and Systems*, volume 26, 2008.

[15] Sinno Jialin Pan and Qiang Yang. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, October 2010.

[16] Richard Socher and Li Fei-Fei. Connecting Modalities : Semi-supervised Segmentation and Annotation of Images Using Unaligned Text Corpora. In *Computer Vision and Pattern Recognition*, 2010.

[17] Luciano Spinello, Rudolph Triebel, and Roland Siegwart. Multiclass Multimodal Detection and Tracking in Urban Environments. *International Journal of Robotics Research*, pages 1–35, 2010.

[18] Severin Stalder, Helmut Grabner, and Luc Van Gool. Beyond semi-supervised tracking: Tracking should be as simple as detection, but not simpler than recognition. In *International Conference on Computer Vision Workshop on On-line Learning for Computer Vision*, September 2009.

[19] Feng Tang, Shane Brennan, Qi Zhao, and Hai Tao. Co-Tracking Using Semi-Supervised Support Vector Machines. In *International Conference on Computer Vision*, October 2007.

[20] Alex Teichman and Sebastian Thrun. Video output of object recognition. 2011. URL http://cs.stanford.edu/people/teichman/rss2011.html.

[21] Alex Teichman, Jesse Levinson, and Sebastian Thrun. The Stanford Track Collection, 2011. URL http://cs.stanford.edu/people/teichman/stc.

[22] Alex Teichman, Jesse Levinson, and Sebastian Thrun. Towards 3D Object Recognition via Classification of Arbitrary Object Tracks. In *International Conference on Robotics and Automation*, 2011.

[23] A. Torralba, K.P. Murphy, and W.T. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. In *Computer Vision and Pattern Recognition*. IEEE, 2004.

[24] Rudolph Triebel and Roland Siegwart. Segmentation and Unsupervised Part-based Discovery of Repetitive Objects. In *Robotics: Science and Systems*, 2010.

[25] Kai M. Wurm, Rainer Kummerle, Cyrill Stachniss, and Wolfram Burgard. *Improving robot navigation in structured outdoor environments by identifying vegetation from laser data*. IEEE, October 2009.

[26] Xiaojin Zhu. Semi-Supervised Learning Literature Survey, 2005.

[27] Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. In *International Conference on Machine Learning*, 2003.