Towards 3D Object Recognition via Classification of Arbitrary Object Tracks

Alex Teichman, Jesse Levinson, Sebastian Thrun
Stanford Artificial Intelligence Laboratory
{teichman, jessel, thrun}@cs.stanford.edu

***Abstract*— Object recognition is a critical next step for autonomous robots, but a solution to the problem has remained elusive. Prior 3D-sensor-based work largely classifies individual point cloud segments or uses class-specific trackers. In this paper, we take the approach of classifying the tracks of all visible objects. Our new track classification method, based on a mathematically principled method of combining log odds estimators, is fast enough for real time use, is non-specific to object class, and performs well (98.5% accuracy) on the task of classifying correctly-tracked, well-segmented objects into car, pedestrian, bicyclist, and background classes.**

We evaluate the classifier's performance using the Stanford Track Collection, a new dataset of about 1.3 million labeled point clouds in about 14,000 tracks recorded from an autonomous vehicle research platform. This dataset, which we make publicly available, contains tracks extracted from about one hour of 360-degree, 10Hz depth information recorded both while driving on busy campus streets and parked at busy intersections.

I. Introduction

Object recognition in dynamic environments is one of the primary unsolved challenges facing the robotics community. A solution would include segmentation, tracking, and classification components, and would allow for the addition of new object classes without the need for an expert to specify new models. Joint solutions are currently impractical; however, if the classification of well-segmented, correctly-tracked objects were solved, there is a clear path towards a joint solution using EM-like methods or multi-hypothesis trackers.

The recent introduction of dense 3D sensors makes now an ideal time to explore a large dataset approach to track classification. These sensors enable data-driven segmentation and tracking of *arbitrary* objects – that is, all objects in the environment regardless of object class. This is especially effective in the driving context, where objects often actively work to stay well-segmented from each other. Tracking and segmentation greatly facilitate the process of labeling large amounts of data, as labeling an object which has been correctly segmented and tracked can provide hundreds or thousands of training examples in a single keypress.

Our particular task of track classification is inspired by the autonomous driving problem, in which it is useful to track *all* objects in the environment and additionally recognize some object classes that must be treated specially. We address a multiclass problem in which tracks must be classified as car, pedestrian, bicyclist, or background (any other object class).

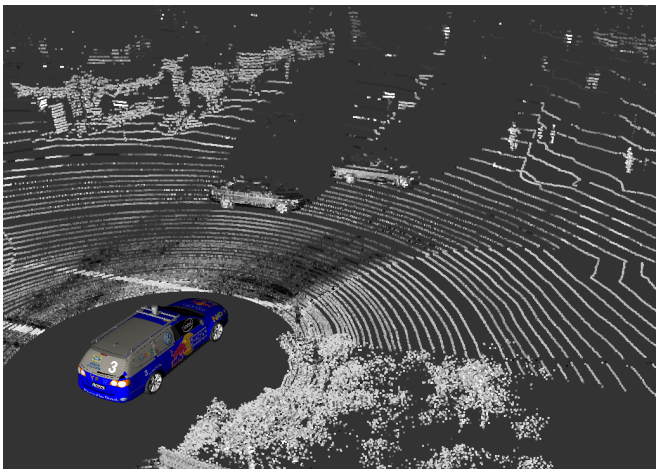

Fig. 1: Example scan of two other cars and a bicyclist at an intersection. Points are colored by return intensity.

None of the algorithms described here are specifically adapted to any particular dense 3D sensor, and the addition of new object classes requires only a labeled dataset of tracks that include the new object class. In this sense, our algorithms apply well beyond our particular sensors and classification task.

The contributions of this paper are three-fold. First, we describe a novel approach to the classification of arbitrary object tracks which performs well on a large, real-world data set. This algorithm makes progress towards the overall goal of object recognition in dynamic 3D environments. Second, we adapt the ideas of hybrid generative / discriminative models [1] to alleviate the overly strong conditional independence assumptions made when combining log odds estimates over time. Third, we provide a large dataset of labeled and tracked 3D point clouds to the research community.

A. Related Work

Current object recognition methods generally fall into two categories. First, multi-class object recognition in static scenes continues to be an active area of research, *e.g.* in [2], [3], [4], [5], [6]. One path to object recognition in dynamic scenes is to solve static scene object recognition so well that tracking detections is sufficient, as in [7], [8]. However, this approach is currently not possible for general object recognition, and the temporal nature of our task offers much

Fig. 2: Junior, the autonomous vehicle research platform. In this work, we use a Velodyne dense LIDAR sensor and an Applanix inertial measurement unit to track objects while moving.

Fig. 3: Example segments from tracks of objects. Columns contain the three foreground object classes (car, pedestrian, bicyclist) and background objects (everything else). *All* objects in the environment are tracked, and the task is to classify tracks with the appropriated label. Points are colored by laser return intensity.

useful information.

Second, object recognition via the tracking of *specific* object classes has been well-studied, *e.g.* in [9], [10], [11], [12], [13], but these methods require the explicit specification of new tracker models for each additional object class. In general, object recognition in dynamic scenes using class-specific trackers will (presumably) obtain better performance for the classes they are designed for, but at the cost of additional expert time to specify new models if they are needed. For specific systems, *e.g.* those that are only concerned with recognizing humans for a human-robot interaction task, this is an entirely appropriate tradeoff; however, in this paper, we propose a classification method that can increase the number of classes without necessitating onerous work by an object recognition expert.

There is only limited prior work on methods that classify tracks of arbitrary objects. The work of [14] considers the task of learning exemplar models for arbitrary object tracks in an unsupervised way. In contrast, the method in this paper is fully supervised, but can handle the large numbers of diverse distractor objects that are seen in operation of the autonomous vehicle on typical streets. The most similar work to ours, [15], fuses 2D laser and camera imagery to track pedestrians and cars at ranges less than 15 meters. It is clear that there remains a large gap between the state of the art and what would be required for a practical, high-reliability object recognition implementation that works at long range. Our work helps close this gap by demonstrating high track classification performance on a very large dataset set at longer ranges than have previously been considered.

This paper is structured as follows. In Section II, we discuss the Stanford Track Collection and the track extraction process. In Section III we describe the log odds estimator (boosting) and our method of combining boosting outputs. In Section IV we review the descriptors used by the boosting classifiers. Finally, results and conclusions are presented in Sections V and VI.

II. The Stanford Track Collection

The purpose of this dataset is to provide a large-scale testing ground for track classification in busy street scenes. The dataset includes over 1.3 million labeled segmented objects across roughly 14,000 tracks, with a maximum range of about 70 meters and mean range of about 30 meters. Table I details the breakdown by class, and Figure 4 shows the distribution of range to objects. The large percentage of tracks that do not fall into the car, pedestrian, or bicyclist classes (83%) is a natural consequence of collecting the dataset directly from an autonomous vehicle research platform while driving on normal streets; the class distribution thus reflects that of a vehicle's typical environment. The dataset is available at [16].

In the following sections, we will make use of several terms: a *scan* is the set of depth measurements recorded during one 360-degree revolution of the dense 3D sensor; a *track* is a temporal series of 3D point clouds of the same object, derived from segmentation of the scans; and a *segment* is a single point cloud from a track.

A. Details

To ensure valid results, the training set and test set are drawn from different locations at different times. Pooling all tracks, then drawing them at random into the test and training sets would be flawed; it is common for several distinct tracks

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 segments

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 Stanford Track Collection by class. Tracks were collected from busy streets and intersections.

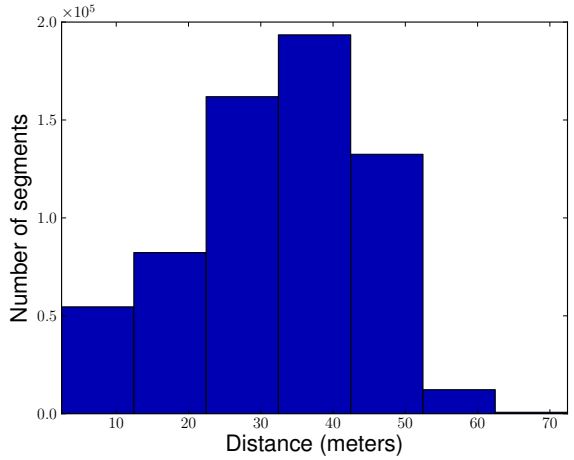

Fig. 4: Histogram of distance to tracked objects in the test set.

to exist for the same object, as occlusions can cause a tracked object to be lost and then picked up as a new track.

The details of the labeling in this dataset are motivated by the autonomous driving task. For example, because it is relatively common for two pedestrians to be segmented together, this occurrence is labeled as a pedestrian in the dataset. Similarly, rollerbladers, skateboarders, pedestrians walking bicycles, pedestrians pulling suitcases, and pedestrians on crutches are all labeled as pedestrians in this dataset. The car class contains sedans, vans, trucks, cars with trailers, etc., but does not include tractor trailers or large buses. There are many reasonable choices of where to draw class lines, and do not believe any particular choice is essential to this work provided it is consistently applied.

To avoid confounding failures of the track classification method with failures of the tracking and segmentation components of a full object recognition system, we invalidate *foreground* tracks (those of cars, pedestrians, and bicyclists) in which more than 10% of the segments are incorrect due to tracking or under-segmentation errors. The over-segmentation of an object, *i.e.* when an object is segmented into more than one part, is *not* considered a segmentation error, as this is a common case that must be dealt with in practice. Additionally, with background tracks, what constitutes a tracking or segmentation error is frequently ambiguous. As this case is also a common one that a track classifier must handle correctly, tracking and segmentation errors of *background* objects are *not* considered inconsistent. About 350 inconsistent tracks exist in the dataset; this constitutes about 2% of the entire dataset or about 13% of foreground tracks.

Finally, we require that all tracks have at least ten segments (*i.e.* the objects is seen for at least one second) and that at least one of these segments has greater than 75 points.

B. Track Extraction Process

Our research vehicle, shown in Figure 2, senses the environment with a Velodyne HDL-64E S2, a dense 64-beam scanning LIDAR that provides 360-degree coverage at 10 Hz, generating just over 1 million 3D points with associated infrared remittance values per second. A tightly coupled GPS/IMU pose system, the Applanix LV-420, provides inertial updates and global position estimates at 200 Hz, enabling tracking while the vehicle is moving. The LIDAR is calibrated using the algorithm of [17].

Scans are segmented using a connected components algorithm on an obstacle map. The 2D obstacle grid is built using standard efficient methods, then clustered with a flood fill algorithm. Clusters that are too large or too small to be of interest are dropped.

Tracking of cluster centroids is accomplished using linear Kalman filters. The data association problem is resolved by distance from measurement centroid to the expected position of each filter. Any cluster not matched to a filter spawns a new filter. Any filter not matched to a cluster has its prediction step run but not its update step, increasing the uncertainty in position; the filter is removed after position uncertainty exceeds a threshold.

We note that the specific segmentation and tracking algorithms discussed here are not central to our overall approach, and more sophisticated algorithms could be readily substituted in different contexts.

III. Classification Methods

We first describe the boosting methods used to classify individual segments, and then we describe how the outputs of the boosting classifiers are combined into a final log odds estimate for a track. We make some references to the descriptors used in these sections, but defer those details to Section IV.

A. Boosting Framework

Boosting is a method of combining the predictions of many weak classifiers into a single, higher-accuracy strong classifier. Intuitively, the weak classifiers encode simple rules: for example, "if the object has a large bounding box, then it is probably not a pedestrian," or "an object that has a side profile that looks like this one is probably a bicyclist."

We use two boosting classifiers in our system. First, the *segment classifier* makes a prediction when given a set of

segment descriptors, *i.e.* a set of vectors which describe the appearance of an object at a point in time. Second, the *holistic classifier* makes a prediction when given a set of *holistic descriptors* of a track, *i.e.* a set of vectors that describe speed, acceleration, and other properties of the track as a whole.

Taking the optimization perspective on boosting, in a two-class problem we desire a solution to

$$\underset{H}{\text{minimize}} \quad \mathbb{E}\left[\exp\left(-\frac{1}{2}Y\,H(Z)\right)\right], \tag{1}$$

where the expectation is over the joint distribution, Y is the class label, Z is a set of descriptors, and H is the strong classifier. The $1/2$ term causes the optimal solution to this problem to be $H(z) = \log \frac{\mathbb{P}(Y=1|z)}{\mathbb{P}(Y=-1|z)}$; see [18] for details. In this paper, we assume all boosting classifiers use this form and thus output log odds estimates rather than the usual $1/2$ log odds.

In practice, we approximate the expectation with a sum, resulting in

$$\underset{H}{\text{minimize}} \quad \sum_{m=1}^{M} \exp\left(-\frac{1}{2}y_m H(z_m)\right),$$

with training examples (y_m, z_m), where $y_m \in \{-1,+1\}$ is the class label and z_m is the descriptor set for training example m.

Extending to the multiclass boosting formulation, we use a 1-vs-all scheme, resulting in the optimization problem

$$\underset{H}{\text{minimize}} \quad \sum_{c=1}^{C}\sum_{m=1}^{M} \exp\left(-\frac{1}{2}y_m^c H(z_m, c)\right).$$

Here, the strong classifier H makes a prediction for each class c. The integer class label y_m is one of $\{1,2,\ldots,C\}$. If $y_m = c$, $y_m^c = +1$; otherwise $y_m^c = -1$. Background objects have $y_m^c = -1$ for all c.

We use a variant of GentleBoost [18] and JointBoost [19] to handle the multi-class, multi-descriptor, high-dimensional nature of the problem. 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 to make predictions for all 1-vs-all classification problems.

As is usual in boosting, the strong classifier H is defined as a sum of the weak classifiers h_k for all $k \in \{1,2,\ldots,K-1\}$, *i.e.*

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

Our weak classifiers take the form

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

where θ_k is in $\mathbb{R}$, f_k chooses a particular descriptor in the set z, and x_k is a descriptor in the same space as $f_k(z)$. The response value a_k^c is positive if the weak classifier predicts the descriptor to be of class c, and negative otherwise; $|a_k^c|$ is the confidence of the prediction. Geometrically, this means that a weak classifier makes a prediction a_k^c about a descriptor $f_k(z)$ if the descriptor falls within a hypersphere of radius θ_k centered at the point x_k.

The strong classifier can be imagined as a set of different-sized hyperspheres living across many descriptors spaces of different dimensions. A test example will have a descriptor in each of these descriptor spaces, and the weak classifier hyperspheres which contain it all respond with their own predictions for each class.

The strong classifier is learned using the procedure of GentleBoost [18]; new weak classifiers are iteratively added by solving

$$\underset{f_K, x_K, \theta_K, \vec{a}_K}{\text{minimize}} \quad \sum_c \sum_m w_m^c \exp\left(-\frac{1}{2}y_m^c h_K(z_m, c)\right), \tag{2}$$

where $w_m^c = \exp(-\frac{1}{2}y_m^c \sum_k h_k(z_m, c))$ is the multiclass boosting weight.

This problem is solved through a combination of directed search, dynamic programming, and convex optimization. We wish to direct the learning process to focus its attention on difficult examples. Concretely, a training example is sampled from the boosting weights distribution and a descriptor space is chosen uniformly at random; this specifies an f_K and an x_K. Given f_K, x_K, and θ_K, the value of $\vec{a}_K$ can be found analytically with a single Newton step for each class. Dynamic programming efficiently solves for $\vec{a}_K$ for each possible θ, *i.e.* one θ per training example other x_K. This process is repeated for a number of choices of (f_K, x_K) pairs. The best evaluated choice of $(f_K, x_K, \theta_K, \vec{a}_K)$ in terms of the optimization objective (2) is then added as a new weak classifier.

B. The Augmented Discrete Bayes Filter

We have now described a method for producing log odds estimates given the descriptors of a segment (*e.g.* object shape) or the descriptors of a track as a whole (*e.g.* mean speed). What remains is to specify a principled and effective method of combining these into a final log odds estimate for a track.

Central to our approach is the fact that a boosting classifier outputs an estimate of the log odds. This enables their combination with a discrete Bayes filter. Other log odds estimators, such as logistic regression, could be substituted into this framework.

To simplify the notation, the following section considers only two-class models; the extension to multiclass is straightforward.

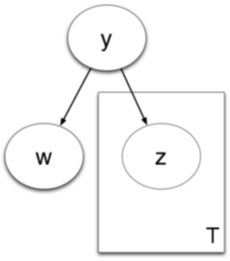

Fig. 5: The graphical model encoding the independencies in the discrete Bayes filter for our task. Each track has one label y, one set of holistic descriptors w, and one set of segment descriptors z_t for each segment in the track. T is the total number of segments in the track.

For reference, we provide the definitions of several symbols below.

- z_t – The segment descriptors for the t-th segment in a track. These include spin images, bounding box size, and virtual orthographic camera views.
- w – The holistic descriptors of a track, which include mean speed, maximum speed, maximum angular velocity, etc.
- $\mathscr{L}(x) = \log \frac{\mathbb{P}(Y=1|x)}{\mathbb{P}(Y=-1|x)}$ – The log odds given x, for any choice of x.
- $\mathscr{L}_0 = \log \frac{\mathbb{P}(Y=1)}{\mathbb{P}(Y=-1)}$ – The log prior odds.
- H^H – The holistic classifier. Returns a log odds estimate given the holistic descriptors w of a track.
- H^S – The segment classifier. Returns a log odds estimate given the segment descriptors z_t of segment t.
- $\mathscr{L}_0^S$ – The empirical estimate of the log prior odds for the segment classifier.
- $\mathscr{L}_0^H$ – The empirical estimate of the log prior odds for the holistic classifier.

As in the discrete Bayes filter, the independencies of the model are the same as those in naïve Bayes; see Figure 5.

We desire an estimate of $\mathscr{L}(w, z_{1:T})$, the log odds given all the information known about a track. Expanding this term out, we have

$$\begin{aligned}
\mathscr{L}(w, z_{1:T}) &= \log \frac{\mathbb{P}(Y=1|w, z_{1:T})}{\mathbb{P}(Y=-1|w, z_{1:T})} \\
&= \mathscr{L}_0 + \log \frac{\mathbb{P}(w, z_{1:T}|Y=1)}{\mathbb{P}(w, z_{1:T}|Y=-1)} \\
&= \mathscr{L}_0 + \log \frac{\mathbb{P}(w|Y=1)}{\mathbb{P}(w|Y=-1)} + \sum_{t=1}^{T} \log \frac{\mathbb{P}(z_t|Y=1)}{\mathbb{P}(z_t|Y=-1)} \\
&= \mathscr{L}(w) + \sum_{t=1}^{T} \left(\mathscr{L}(z_t) - \mathscr{L}_0\right) \\
&\approx H^H(w) + \sum_{t=1}^{T} \left(H^S(z_t) - \mathscr{L}_0^S\right). \qquad (3)
\end{aligned}$$

The *naïve discrete Bayes filter*, (3), is arrived at from $\mathscr{L}(w, z_{1:T})$ via Bayes' rule, the conditional independence assumptions shown in Figure 5, and then another application of Bayes' rule. The final approximation is a result of substituting in the holistic classifier H^H and the segment classifier H^S, which provide estimates of the log odds given the holistic descriptors and the segment descriptors, respectively.

A readily apparent shortcoming of the naïve discrete Bayes filter is that the holistic classifier H^H becomes increasingly negligible as track length increases. If the conditional independence assumptions were strictly correct, this behavior would be desired. However, due to a variety of possible dependencies between segments that arise in practice, the individual segment classifier estimates are not truly conditionally independent given the object class. This shortcoming is addressed in the *normalized discrete Bayes filter*,

$$H^H(w) + \frac{1}{T} \sum_{t=1}^{T} \left(H^S(z_t) - \mathscr{L}_0^S\right).$$

Finally, we notice that the (again, incorrect) conditional independence assumption $W \perp Z_{1:T} \mid Y$ remains, and that we can compensate for this approximation by learning additional parameters $\alpha, \beta, \gamma \in \mathbb{R}$ that weight the relative importance of each term. This gives rise to the *augmented discrete Bayes filter* (ADBF),

$$\begin{aligned}
H^A(w, z_{1:T}) = \alpha \mathscr{L}_0^H + \beta \left(H^H(w) - \mathscr{L}_0^H\right) \\
+ \gamma \frac{1}{T} \sum_{t=1}^{T} \left(H^S(z_t) - \mathscr{L}_0^S\right). \qquad (4)
\end{aligned}$$

The weights are learned via the unconstrained convex optimization problem

$$\underset{\alpha,\beta,\gamma}{\text{minimize}} \quad \sum_m \log\left(1 + \exp\left(-y_m H^A\left(w_m, z_{1:T}^m\right)\right)\right). \qquad (5)$$

This formulation is equivalent to logistic regression on the output of the holistic and segment classifiers with an intercept term. We note that the optimization problem

$$\underset{H}{\text{minimize}} \quad \mathbb{E}\left[\log\left(1 + e^{-YH(Z)}\right)\right]$$

has the same optimal solution of $H(z) = \log \frac{\mathbb{P}(Y=1|z)}{\mathbb{P}(Y=-1|z)}$ as in (1); therefore the final augmented discrete Bayes filter is optimized to produce the best log odds estimate possible.

In summary, to obtain a log odds estimator for a track, we cannot simply throw all descriptors into a logistic regression or boosting model because the length of a track is variable. The discrete Bayes filter handles the variable length of tracks nicely, but, like naïve Bayes, makes strong conditional independence assumptions that are frequently incorrect. Like logistic regression, the ADBF learns parameters that can compensate for these mistakes to some extent.

The ADBF builds on ideas used in [1] for document classification, in which weights are learned to combine terms in a multinomial naïve Bayes model for different regions of a document. Here, we use a comparatively complex discriminative model in place of the simple generative model of multinomial naïve Bayes, and combine entirely different models that operate on different descriptor sets.

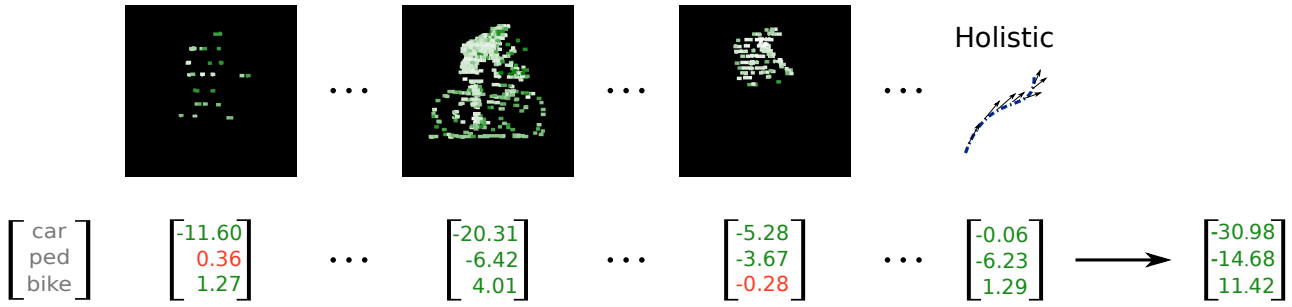

Fig. 6: Schematic drawing of the classification process. Log odds estimates from the segment classifier and from the holistic classifier are combined using the *augmented discrete Bayes filter*. In this example, a distant bicyclist is initially difficult to distinguish from a pedestrian but becomes recognizable as it draws nearer. The holistic classifier, evaluating statistics of the track such as mean speed and maximum acceleration, is fairly confident that the object is not a pedestrian, but relatively uncertain otherwise. A transient occlusion causes an incorrect segment classification, but the overall prediction is correct.

C. Implementation Details

The number of candidate weak classifier centers – that is, the number of (f_K, x_K) pairs – evaluated at each round of boosting was set to 20. The segment classifier was trained until 1000 weak classifiers, and the holistic classifier until 100. To learn the weights in the ADBF, part of the training set was used for training segment and holistic classifiers and the other part was used in the optimization problem (5).

We found that, in training the segment classifier, using a random sampling of 20% of the training segments did not hurt performance, likely because tracks were sampled at about 10Hz and the variation from one segment to the next is relatively low.

Training of the segment classifier is the slowest part of the training procedure and takes on the order of a few hours in a multithreaded implementation running on an octocore desktop.

IV. Descriptors

In this section, we review the descriptors used for the segment and holistic boosting classifiers. In general, we believe that the details of the descriptors are not as important as the availability of a wide range of descriptors for the classifier to consider.

A. Holistic Descriptors

Holistic descriptors describe the track as a whole. In our system, these include (a) maximum speed, (b) mean speed, (c) maximum acceleration, (d) mean acceleration, (e) maximum angular velocity, and (f) the segment descriptors applied to the accumulated point cloud of all segments in the track. Each motion descriptor is computed relative to a velocity estimate of the track's centroid smoothed over five consecutive segments, or about half a second. We note that there is significant noise in these estimates due to occlusions and segmentation jitter.

The accumulated segment descriptors, while unintuitive for moving objects, can provide complete views of stationary objects that are badly occluded in most or all individual segments.

Fig. 7: Examples of side, top, and front views of objects using virtual orthographic camera intensity images. Pixel intensities reflect the average LIDAR return intensity. Several HOG descriptors are computed on each view. Alignment into a canonical orientation enables seeing consistent views of objects.

B. Segment Descriptors

We use a total of 29 segment descriptors which encode various aspects of object appearance: oriented bounding box size, 4 different parameterizations of spin images [20], and 24 different parameterizations of the histogram of oriented gradients [21] descriptor computed on virtual orthographic camera intensity images.

Central to the descriptor pipeline is the concept of a canonical orientation of a point cloud. In a coordinate system where z is up, the canonical orientation is a rotation of the segment around the z axis in which the long and short sides of the object are aligned with the x and y axes. We use a simple RANSAC algorithm to fit the dominant line in the XY-projection of the segment's points.

Virtual orthographic camera images of the front, side, and top views are taken relative to the canonical orientation of the segment. As such, descriptors derived from these images are invariant to rotations of the 3D object around the vertical axis. Image pixels measure average brightness of the return intensities from the sensor. Examples of the orthographic intensity images are shown in Figure 7.

The histogram of oriented gradients descriptor captures

Track classification

Method	Car	Pedestrian	Bicyclist	Overall
ADBF	98.7%	99.8%	99.9%	98.5%
Normalized DBF	98.0%	99.8%	99.8%	97.7%
Naïve DBF	98.3%	99.6%	99.8%	97.7%
Segment classifier only	98.3%	99.5%	99.8%	97.6%
Holistic classifier only	94.3%	99.2%	99.2%	93.0%
Prior only	86.0%	98.1%	97.7%	81.8%

Single segment classification

Method	Car	Pedestrian	Bicyclist	Overall
ADBF[1]	97.9%	99.6%	99.7%	97.5%
Segment classifier only	95.8%	98.3%	98.4%	93.1%
Prior only	90.7%	96.5%	96.0%	83.3%

TABLE II: Summary of accuracy results for each 1-vs-all problem and for the overall problem.

local object appearance in images with invariance to small translations in the gradient locations. A window size and offset from the edges of the projected segment are specified to capture different parts of an object, such as the front or back wheel of a bicycle; in some cases, these descriptors will be invariant to partial occlusion.

Spin images are all taken about the vertical axis, as other rotations are generally significant in the driving context; cars rarely appear upside down, for example. Spin images are "whitened", *i.e.* scaled to have zero mean and unit variance, so they will be invariant to point cloud density.

V. Results

Performance was evaluated on the Stanford Track Collection. An example of the classification process is shown in Figure 6. The prediction for each track is taken to be the class with the maximum log odds, or background if the log odds for all classes are negative. We note that the log odds approach presented in Section III-B is more naturally suited to the case in which a track can have multiple positive classifications. For example, it may be useful to have both "car" and "police car" classes, where instances of the latter are also instances of the former.

Because the dataset is drawn from the tracks recorded in actual driving sequences, a large number of background tracks populate the test set. As such, a classifier which simply predicts background for all test tracks would be 81.8% accurate.

Accuracy results are summarized in Table II; the ADBF achieves 98.5% accuracy. Precision-recall curves for the three classes and a confusion matrix are shown in Figures 8 and 9, respectively. The holistic classifier alone reaches only 93.0% accuracy; this method is insufficient on its own, but when combined properly with the segment classifier can improve performance.

Classifying individual segments results in 93.1% accuracy, vs 97.5% accuracy when classifying with the augmented discrete Bayes filter, an error reduction of about 65%.[2]

[1]For comparison to the other segment-wise results, the classification for the track is applied to each individual segment.

[2]While a labeled track is allowed to have up to 10% of its segments be incorrect due to segmentation or tracking errors, the total percentage of incorrectly labeled segments due to this cutoff is below 1%.

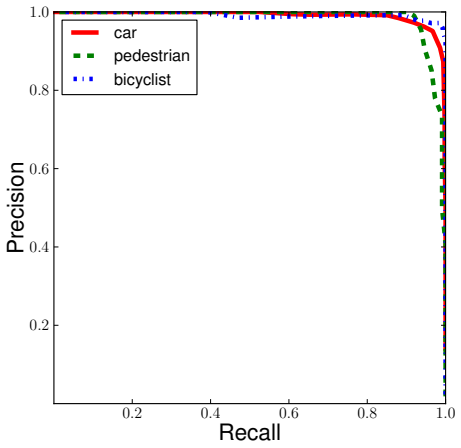

Fig. 8: Precision-recall curve for classification using the augmented discrete Bayes filter.

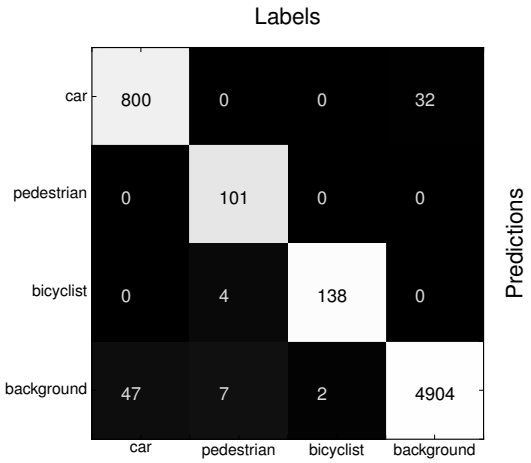

Fig. 9: Confusion matrix for classification using the augmented discrete Bayes filter. Entries in the matrix are numbers of tracks.

Perhaps the largest failure case is illustrated by Figure 10, which shows degradation of the segment classifier's accuracy as the number of observed points in the segment decreases. This result includes both long-range and badly occluded objects, both of which tend to be difficult test cases for the classifier.

Classification speed is fast enough to enable real-time use of the system when considering only objects on the road. The average time to compute a classification for a segment, including descriptor computation and amortizing holistic descriptor computation time across the entire track, is about 2 milliseconds in our multithreaded implementation on an octocore computer. Caching of shared computation in the descriptor pipeline is essential.

There are a number of reasons for the strong performance of our system. Segmentations of dense 3D data make possible descriptors that see objects from consistent viewpoints, as we

use here with the virtual orthographic camera intensity images. Segmentation also provides more consistent descriptors since background noise is removed; in contrast, consider the case of HOG descriptors of pedestrians in camera images, which can have widely varying backgrounds (and therefore widely varying descriptors) for the same pedestrian. Classifying tracks rather than individual segments allows multiple views of an objects to contribute to the final prediction and is a key component of achieving high performance.

VI. Conclusions

We have developed a new multiclass track classification method which demonstrates high accuracy on a large, real-world dataset. Our method, which relies on segmentation and tracking made possible by dense 3D sensors, provides a component of a joint solution to 3D object recognition, which includes segmentation, tracking, and classification. The addition of new object classes does not require the specification of new tracker models, but rather only supplemental labeled tracks of the new object class. Finally, our method is fast enough for real time use in plausible implementations.

In this work, we have assumed one possible decomposition of the full object recognition problem - namely, into segmentation, tracking, and track classification components. Our experiments demonstrate an effective method of track classification given correct segmentation and tracking. Initial work on the full object recognition problem indicates that the majority of classification errors are due to failures of segmentation and tracking, so that direction is one of great interest for making progress on object recognition. Additionally, better occlusion-invariant descriptors would be useful. Dense 3D sensors should enable this work, since depth information can provide occlusion labels.

VII. Acknowledgments

We are deeply grateful to Mike Sokolsky for maintaining the autonomous vehicle research platform, to Christian Plagemann, Neal Parikh, and David Stavens for helpful comments, and to Angad Singh and Jim Lin for assistance with data labeling. The authors acknowledge financial support by the DARPA NeoVision 2 program, and additional financial support by Boeing.

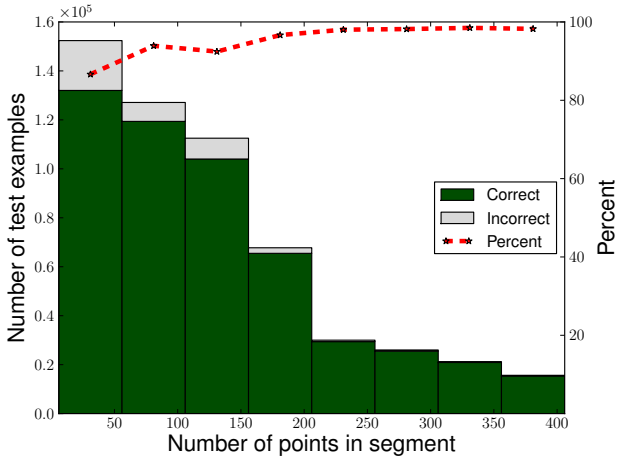

Fig. 10: Accuracy vs number of points for segment-wise classification. Objects with few points account for the majority of misclassifications.

References

[1] R. Raina, Y. Shen, A. Y. Ng, and A. McCallum, "Classification with Hybrid Generative / Discriminative Models," in *Neural Information Processing Systems*, 2004.

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

[3] R. Triebel, R. Schmidt, O. M. Mozos, and W. Burgard, "Instance-based AMN classification for improved object recognition in 2D and 3D laser range data," *International Joint Conference On Artificial Intelligence*, p. 5, 2007.

[4] B. Douillard, A. Brooks, and F. Ramos, "A 3D Laser and Vision Based Classifier," in *Conference on Intelligent Sensors, Sensor Networks and Information Processing*, vol. 1, no. c, 2009.

[5] L. Spinello, K. Arras, R. Triebel, and R. Siegwart, "A Layered Approach to People Detection in 3D Range Data," in *AAAI*, no. 2007, 2010.

[6] P. Espinace, T. Kollar, A. Soto, and N. Roy, "Indoor Scene Recognition through Object Detection," *International Conference on Robotics and Automation*, pp. 1406–1413, 2010.

[7] M. Bansal, S.-H. Jung, B. Matei, J. Eledath, and H. Sawhney, "A Real-Time Pedestrian Detection System Based on Structure and Appearance Classification," *International Conference on Robotics and Automation*, pp. 903–909, 2010.

[8] D. Gavrila and S. Munder, "Multi-cue Pedestrian Detection and Tracking from a Moving Vehicle," *International Journal of Computer Vision*, vol. 73, no. 1, pp. 41–59, June 2007.

[9] M. Andriluka, S. Roth, and B. Schiele, "People-tracking-by-detection and people-detection-by-tracking," in *Computer Vision and Pattern Recognition*. IEEE, 2008, pp. 1–8.

[10] A. Petrovskaya and S. Thrun, "Model based vehicle detection and tracking for autonomous urban driving," *Robotics: Science and Systems*, 2009.

[11] B. Wu and R. Nevatia, "Detection and Tracking of Multiple, Partially Occluded Humans by Bayesian Combination of Edgelet based Part Detectors," *International Journal of Computer Vision*, vol. 75, no. 2, pp. 247–266, Nov. 2007.

[12] D. Schulz, W. Burgard, D. Fox, and A. B. Cremers, "People Tracking with Mobile Robots Using Sample-Based Joint Probabilistic Data Association Filters," *The International Journal of Robotics Research*, vol. 22, no. 2, pp. 99–116, Feb. 2003.

[13] D. Schulz, "A Probabilistic Exemplar Approach to Combine Laser and Vision for Person Tracking," in *Robotics: Science and Systems*, 2006.

[14] M. Luber, K. O. Arras, C. Plagemann, and W. Burgard, "Classifying dynamic objects: An unsupervised approach," in *Robotics: Science and Systems*, vol. 26, no. 2, 2008.

[15] L. Spinello, R. Triebel, and R. Siegwart, "Multiclass Multimodal Detection and Tracking in Urban Environments," *International Journal of Robotics Research*, pp. 1–35, 2010.

[16] A. Teichman, J. Levinson, and S. Thrun, "The Stanford Track Collection," `http://cs.stanford.edu/people/teichman/stc`.

[17] J. Levinson and S. Thrun, "Robust Vehicle Localization in Urban Environments Using Probabilistic Maps," *International Conference on Robotics and Automation*, pp. 4372–4378, 2010.

[18] J. Friedman, T. Hastie, and R. Tibshirani, "Special Invited Paper. Additive Logistic Regression: A Statistical View of Boosting," *The Annals of Statistics*, vol. 28, no. 2, pp. 337 – 374, 2000.

[19] A. Torralba, K. Murphy, and W. Freeman, "Sharing features: efficient boosting procedures for multiclass object detection," in *Computer Vision and Pattern Recognition*, vol. 2. IEEE, 2004, pp. 762–769.

[20] A. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3D scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 433–449, 1999.

[21] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *Computer Vision and Pattern Recognition*, vol. 1. IEEE, 2005, pp. 886–893.