aUToTrack: A Lightweight Object Detection and Tracking System for the SAE AutoDrive Challenge

Keenan Burnett, Sepehr Samavi, Steven L. Waslander, Timothy D. Barfoot, Angela P. Schoellig
Institute for Aerospace Studies
University of Toronto
Toronto, Canada
{keenan.burnett, sepehr}@robotics.utias.utoronto.ca

***Abstract*—The University of Toronto is one of eight teams competing in the SAE AutoDrive Challenge – a competition to develop a self-driving car by 2020. After placing first at the Year 1 challenge [1], we are headed to MCity in June 2019 for the second challenge. There, we will interact with pedestrians, cyclists, and cars. For safe operation, it is critical to have an accurate estimate of the position of all objects surrounding the vehicle. The contributions of this work are twofold: First, we present a new object detection and tracking dataset (UofTPed50), which uses GPS to ground truth the position and velocity of a pedestrian. To our knowledge, a dataset of this type for pedestrians has not been shown in the literature before. Second, we present a lightweight object detection and tracking system (aUToTrack) that uses vision, LIDAR, and GPS/IMU positioning to achieve state-of-the-art performance on the KITTI Object Tracking benchmark. We show that aUToTrack accurately estimates the position and velocity of pedestrians, in real-time, using CPUs only. aUToTrack has been tested in closed-loop experiments on a real self-driving car (seen in Figure 1), and we demonstrate its performance on our dataset.**

***Keywords*-Vision for Autonomous Vehicles, Real-Time Perception, Object Recognition and Detection**

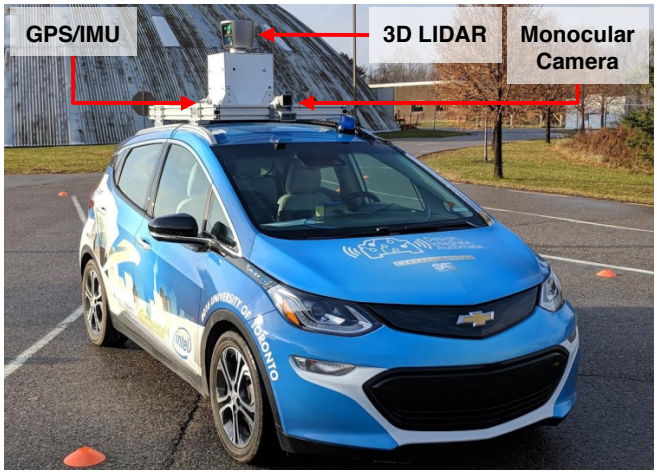

Figure 1. Our self-driving car *Zeus* at the University of Toronto Institute for Aerospace Studies (UTIAS). https://youtu.be/FLCgcgzNo80

I. Introduction

Standard object detection and tracking algorithms used for video understanding use 2D bounding boxes to identify objects of interest. For autonomous driving, 2D bounding boxes are insufficient. A 3D position and velocity estimate is required in order to localize objects in a map and anticipate their motion. Existing object detection benchmarks compare detector outputs against hand-generated labels either in 2D image coordinates or in a 3D sensor frame [2]. In both cases the "ground truth" has been generated by a human with some semi-automatic tools. For this reason, such datasets are subject to human biases in the labelling process.

Current leading methods on these benchmarks do so by replicating the labels observed in the training set [3]. Unfortunately, this does not necessarily mean that these approaches are accurately localizing objects in 3D space, which is the critical problem that we address here.

Existing datasets [2] lack a means for benchmarking 3D Object Detection and Tracking for pedestrians. To address this shortcoming, we introduce UofTPed50, a new dataset that we are making publicly available in June 2019. UofTPed50 includes vision, LIDAR, and GPS/IMU data collected on our self-driving car in 50 scenarios involving interactions with a pedestrian. We use a separate GPS system attached to the pedestrian to obtain ground truth positioning. By using GPS ground truth instead of hand labels, we can rigorously assess the the localization accuracy of our system. To our knowledge, benchmarking pedestrian localization in this manner has not been shown in the literature before.

As a secondary contribution, we describe our approach to Object Detection and Tracking (aUToTrack), and demonstrate its performance on UofTPed50 and KITTI. aUToTrack consists of an off-the-shelf vision-based 2D object detector paired with a LIDAR clustering algorithm to extract a depth for each object. GPS/IMU data is then used to localize objects in a metric reference frame. Given these 3D measurements, we use greedy data association and an Extended Kalman Filter (EKF) to track the position and velocity of each object. Figure 2 illustrates the aUToTrack pipeline. We demonstrate state-of-the-art performance on the KITTI Object Tracking benchmark while using data association and filtering techniques that are faster and much simpler than many competing approaches [4], [5], [6], [7]. We also demonstrate that we can accurately estimate the position and velocity of pedestrians using our UofTPed50 dataset while running our entire pipeline in less than 75 ms on CPUs only.

Estimating the velocity of objects is a difficult problem in self-driving [8] which we tackle in this work. Systems like ours that are lightweight, and capable of running on CPUs are uncommon in the literature but essential in practice. Although aUToTrack was designed for the SAE AutoDrive Challenge, it has utility for many robotic systems deployed in human-centered domains.

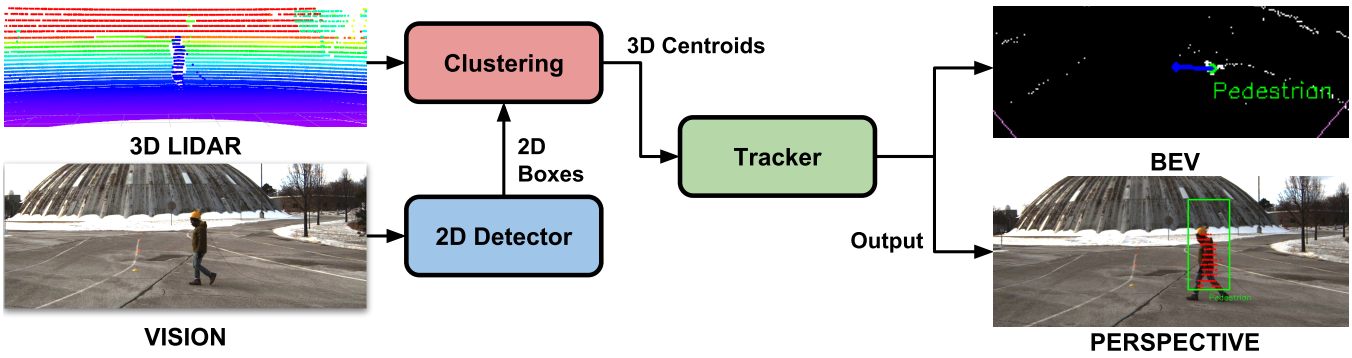

Figure 2. Our pipeline for 3D object detection and tracking. For UofTPed50 experiments, SqueezeDet is our 2D Detector. Clustering consists of Euclidean clustering over the LIDAR points that fall within the bounding boxes when projected onto the image plane. Our tracker consists of gated nearest neighbor data association and an Extended Kalman Filter for tracking position and velocity in 3D space.

II. Related Work

A. Object Detection

Object detectors can be classified into 2D and 3D detectors. Among 2D detectors, two-stage detectors have historically achieved the top accuracy on public benchmarks [9]. Single-stage detectors such as YOLO [10] and SSD [11] tend to be more computationally efficient, but typically do not achieve the best performance. A recent work that focuses on achieving a high framerate on limited hardware is SqueezeDet [12]. A work that focused on achieving the best possible accuracy is Recurrent Rolling Convolutions (RRC).

3D detectors estimate the centroid and volume of objects using a 3D bounding box. Recent works in this area include Frustum PointNets [13] and AVOD [14]. Frustum PointNets uses a 2D detector to cut out a "frustum" from an incoming pointcloud. They then use a specialized fully-connected network to cluster points within the frustum and regress a 3D bounding box. AVOD fuses vision- and LIDAR-based features together in both the region proposal and a bounding box regression stage of their network.

For the purposes of self-driving, the question becomes: is it better to use a 2D or a 3D detector? 2D object detection is perceived to be a more mature field, with many approaches exceeding 90% mean Average Precision (mAP) at acceptable framerates [2]. On the other hand, 3D detections are more useful for self-driving since they incorporate an accurate centroid estimate which can be used directly to safely interact with traffic participants. However, most of these detectors are computationally very expensive. AVOD, which is one of the faster 3D detectors, requires a TITAN X GPU to achieve an inference time of 100ms. For this reason, the decision to use a fast single-stage detector, in our case SqueezeDet, becomes clear. Compared with other approaches in literature, we are able to estimate accurate 3D centroids for pedestrians using CPUs only. This is uncommon in literature and enables our system to be used in a real self-driving car.

B. Object Tracking

Recent works in the area of 2D object tracking include [4], [5], [6], [15]. In [4], an aggregated local flow descriptor is used to associate targets and detections over a temporal window. In [5], geometry, object shape, and pose costs are used to augment data association. In [6], the authors formulate multi-object tracking using Markov Decision Processes.

The above approaches score high on KITTI, however each of their runtimes exceeds 0.2s, preventing their use in a self-driving car. Our approach is simpler and we are able to achieve competitive performance with lower runtime.

In [15], the authors describe a fast 2D object tracker that consists of a Kalman Filter and the Hungarian algorithm. For 2D tracking, they achieve competitive results on the MOT benchmark [16], while being much faster than competitors.

In 3D object detection and tracking, recent works include [17], [18]. In [17], a model-free object detection scheme based on convexity segmentation is used with a Kalman filter and constant velocity motion model. In [18], a model-free 3D object detection and tracking method based on motion cues is used. Both approaches use the same dataset, which has a sensor vehicle equipped with a LIDAR and a GPS/IMU as well as a separate GPS sensor used to ground truth the target vehicle. These approaches both have several drawbacks. First, LIDAR-based approaches are not able to classify different objects easily. In addition, [18] struggles to detect pedestrians, partly because of their motion-based detections. Neither approach presents results for pedestrian detection. The dataset presented in this work has more variation in target distance, velocities, and trajectories and is organized for a rigorous evaluation of an object detection and tracking system.

A recent work similar to ours is [7]. In this work, the authors employ a DNN trained to detect objects and estimate depth from a single image. For tracking, they use a Poisson Multi-Bernoulli Mixture filter. They achieve competitive results on KITTI using only images. However, when LIDAR is readily available, approaches such as ours perform better.

III. 2D Object Detector

We use squeezeDet [12] as our 2D detector for real-time experiments on UofTPed50 due to its high accuracy on the KITTI dataset for 2D Object Detection. SqueezeDet achieves top-25 accuracy for pedestrians, and top-75 accuracy for cars. However, it should be noted that among its competitors, it is one of the fastest, achieving a framerate of 57.2 FPS on a TITAN X GPU (1248x384 input images). SqueezeDet is also fully-convolutional, which makes it an attractive option for run-time optimization. Using Intel's OpenVINO Deep Learning Acceleration Tools [19], we are able to run SqueezeDet at 22.2 FPS using only 12 CPU cores (1248x384 input images).

For experiments on the KITTI Tracking benchmark, we use Recurrent Rolling Convolutions (RRC) [3] due to the fact that it is top ranked among published works on the 2D Object Detection benchmark. RRC is substantially slower than SqueezeDet, requiring over 1500 ms to process a single frame on a GTX1080Ti, therefore precluding its usage in real-time experiments. However, the increase in recall on the training set is substantial: from 65% with SqueezeDet to 94% with RRC.

IV. Clustering

In order to achieve real-time performance, we restrict our attention to points in front of the vehicle up to 40m away, and 15m to each side. Projected on to the ground, this results in a Bird's Eye View (BEV) pointcloud shape of 40m x 30m. We subsequently segment and extract the ground plane, so that only vertical objects remain. The remaining points are transformed from the LIDAR frame into the camera frame. A corresponding set of image plane locations, are obtained by projecting LIDAR points onto the image plane using an ideal perspective projection model and multiplying by the intrinsic camera matrix.

Using the projected coordinates, we retrieve the LIDAR points that lie inside each 2D bounding box. We then use Euclidean clustering on the corresponding 3D LIDAR points [20]. Several heuristics are used to choose the best cluster from the clustering process. These heuristics include: comparing the detected distance to the expected size of the object, filtering based on cluster height, and counting the number of points per cluster. Once the best cluster has been chosen, the position of the object is computed as the position of the centroid of points in the cluster. Although this method for computing an object's centroid works well for pedestrians, it does not perform as well for cars. This is because the clustered points only consist of the points reflected from the face of the object facing the LIDAR, which can be far from the true center for large objects such as cars.

V. Tracker Setup

For each object, we keep a record of the state, $\hat{\mathbf{x}}$, covariance, $\hat{\mathbf{P}}$, *class, confidence level*, and *counters* for track management. The state is defined in Equation (1), where (x, y, z) is the position, $(\dot{x}, \dot{y})$ is the velocity within the ground plane, and (w, h) are the bounding box width and height in the image plane. The position and velocity are tracked in a static map frame external to the vehicle. The class, which is output by SqueezeDet can be one of: $(car, pedestrian, cyclist, other)$. For track management, we count the number of frames an objects has gone without being observed, and a boolean for "trial" objects.

A. Constant Velocity Motion and Measurement Models

The following equations describe the constant velocity motion model that is used for all detected objects:

$$\mathbf{x} = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & w & h \end{bmatrix}^T \tag{1}$$

$$\mathbf{y} = \begin{bmatrix} x & y & z & w & h \end{bmatrix}^T \tag{2}$$

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \boldsymbol{\omega} \tag{3}$$

$$\mathbf{y}_k = \mathbf{C}\mathbf{x}_k + \boldsymbol{n} \tag{4}$$

$$\boldsymbol{\omega} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}) \tag{5}$$

$$\boldsymbol{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}) \tag{6}$$

$$\mathbf{A} = \begin{bmatrix} \mathbf{1}_2 & \mathbf{0}_{2\times 1} & T * \mathbf{1}_2 & \mathbf{0}_2 \\ \mathbf{0}_{5\times 2} & \mathbf{1}_5 & & \end{bmatrix} \tag{7}$$

$$\mathbf{C} = \begin{bmatrix} \mathbf{1}_3 & \mathbf{0}_{3\times 4} \\ \mathbf{0}_{2\times 5} & \mathbf{1}_2 \end{bmatrix} \tag{8}$$

In this setup, $\mathbf{x}$ is the state of the object, $\mathbf{y}$ is the measurement, $\boldsymbol{\omega}$ is the system noise, $\boldsymbol{n}$ is the measurement noise, $\mathbf{A}$ is the state matrix, and $\mathbf{C}$ is the observation matrix. We assume that the object moves with constant velocity between each clock cycle.

B. Linear Kalman Filter

The following equations describe the linear Kalman filter used to track the state and covariance of each object:

$$\check{\mathbf{x}}_k = \mathbf{A}\hat{\mathbf{x}}_{k-1} \tag{9}$$

$$\check{\mathbf{P}}_k = \mathbf{A}\hat{\mathbf{P}}_{k-1}\mathbf{A}^T + \mathbf{Q} \tag{10}$$

$$\mathbf{K}_k = \check{\mathbf{P}}_k\mathbf{C}^T(\mathbf{C}\check{\mathbf{P}}_k\mathbf{C}^T + \mathbf{R})^{-1} \tag{11}$$

$$\hat{\mathbf{P}}_k = (\mathbf{1} - \mathbf{K}_k\mathbf{C})\check{\mathbf{P}}_k \tag{12}$$

$$\hat{\mathbf{x}}_k = \check{\mathbf{x}}_k + \mathbf{K}_k(\mathbf{y}_k - \mathbf{C}\check{\mathbf{x}}_k) \tag{13}$$

In this setup, Equations (9)-(10) are the prediction, Equation (11) is the Kalman gain, and Equations (12)-(13) are the corrector equations.

C. Tracking in World Frame

We require the position and velocity of objects in a static map frame denoted o. Raw measurements are given in a sensor frame attached to the vehicle, denoted $c2$. We

augment some of our vectors in order to track in the static frame as follows:

$$\mathbf{x}' = \begin{bmatrix} x & y & z & 1 & \dot{x} & \dot{y} & w & h \end{bmatrix}^T \tag{14}$$

$$\mathbf{C}' = \begin{bmatrix} \mathbf{1}_3 & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times2} \\ \mathbf{0}_{2\times3} & \mathbf{0}_{2\times3} & \mathbf{1}_2 \end{bmatrix} \tag{15}$$

$$\mathbf{T}'_{c_2o} = \begin{bmatrix} \mathbf{T}_{c_2o} & \mathbf{0}_{4\times4} \\ \mathbf{0}_{4\times4} & \mathbf{1}_4 \end{bmatrix} \tag{16}$$

$$\mathbf{G}_k = \mathbf{C}'\mathbf{T}'_{c_2o} \tag{17}$$

$$\mathbf{y} = \mathbf{C}'\mathbf{T}'_{c_2o}(\mathbf{x}'_k + \boldsymbol{n}') \tag{18}$$

$$\mathbf{R}' = \mathbf{R} \tag{19}$$

Keeping our prediction equations the same, our update equations become:

$$\mathbf{K}_k = \check{\mathbf{P}}'_k\mathbf{G}_k^T(\mathbf{G}_k\check{\mathbf{P}}'_k\mathbf{G}_k^T + \mathbf{R})^{-1} \tag{20}$$

$$\hat{\mathbf{P}}'_k = (\mathbf{1} - \mathbf{K}_k\mathbf{G}_k)\check{\mathbf{P}}'_k \tag{21}$$

$$\hat{\mathbf{x}}'_k = \check{\mathbf{x}}'_k + \mathbf{K}_k(\mathbf{y}_k - \mathbf{G}_k\check{\mathbf{x}}'_k) \tag{22}$$

We then remove augmented components of $\hat{\mathbf{P}}'_k$, $\hat{\mathbf{x}}'_k$ at the end before publishing our answer to other components of the system.

VI. Data Association

We use static gates in order to associate new detections to existing tracks. The gates are constructed using the maximum possible inter-frame motion. For example, assuming a maximum speed of 6 m/s for pedestrians and a maximum speed of 20 m/s for vehicles and a 0.1s time step, we have a gating region of 0.6 m and 2.0 m respectively. In order to account for the highly variable speeds of cars, we use the tracked velocity of each car to calculate the radius of its gating region.

We use a greedy approach to associate measurements to tracked objects. For each tracked object, we evaluate the object's distance to the observations that are within its gate and find the nearest neighbour to the object among the gated observations. We associate the measurement with the tracked object and remove the measurement from the list. We repeat the process for every object that has detections in its gate.

VII. Managing The List of Tracked Objects

We employ a strategy of greedy track creation and lazy deletion for managing tracks. In greedy track creation, every observation becomes a new track. However, every track must go through a "trial period". While objects are in their trial period, they are removed from the list of objects being tracked if they miss a single frame.

Once objects are promoted from their trial period, we count the number of consecutive frames that an object has been unobserved for. In order for a non-trial track to be removed from the list, there must be no associated measurements for Y (Y=5) consecutive frames. This constitutes a lazy deletion process.

Table I summarizes the algorithm's performance. We note that the majority of the run-time can be attributed to the 2D detector.

Table I
Runtime of Each Component in our Pipeline.

Component	Run Time	Hardware
SqueezeDet	20ms	GTX1080Ti
SqueezeDet	45 ms	Intel Xeon E5-2699R (12 cores)
Clustering	20 ms	Intel Xeon E5-2699R (1 core)
Tracker	5 ms	Intel Xeon E5-2699R (1 core)
Total Run Time:	70 ms	Intel Xeon E5-2699R Only

VIII. UofTPed50

As the primary contribution of this work, we are releasing a new dataset, named UofTPed50, for benchmarking 3D object detection and tracking of pedestrians. Our focus is on providing an accurate position and velocity benchmark for a pedestrian in the form of GPS ground truth. Currently, a comprehensive benchmark of this type is not publicly available. UofTPed50 consists of 50 sequences of varying distance, trajectory shape, pedestrian appearance, and sensor vehicle velocities. Each sequence contains one pedestrian. The scenarios are broken into five groups:

1) A total of 36 sequences tracking straight lateral trajectories with respect to the stationary car at seven distances performed by three pedestrians
2) A total of 3 sequences tracking straight lateral trajectories with respect to a dynamic car performed by two pedestrians
3) A total of 4 sequences tracking straight longitudinal trajectories with respect to the stationary car performed by two pedestrians
4) A total of 4 sequences tracking straight longitudinal trajectories with respect to a dynamic car performed by two pedestrians
5) A total of 6 sequences tracking complex trajectories (i.e., curves and Zig-Zags) with respect to the stationary car performed by two pedestrians

We collected UofTPed50 data on our self-driving car, Zeus, illustrated in Figure 1. Zeus is a 2017 Chevrolet Bolt Electric Vehicle. Sensor data was collected from a Velodyne HDL-64 LIDAR, a 5 MP Blackfly BFS-U3-51S5C-C monocular camera, and a NovAtel PwrPak7 GPS/IMU with TerraStar corrections (<10cm reported position error). Position data for the pedestrian was collected by attaching the tethered antenna of a NovAtel PwrPak7 GPS, also with TerraStar corrections, to the pedestrian's head as illustrated in Figure 3. Ground truth velocity for the pedestrian was generated by smoothing the finite difference between GPS waypoints. To synchronize data between the car and the

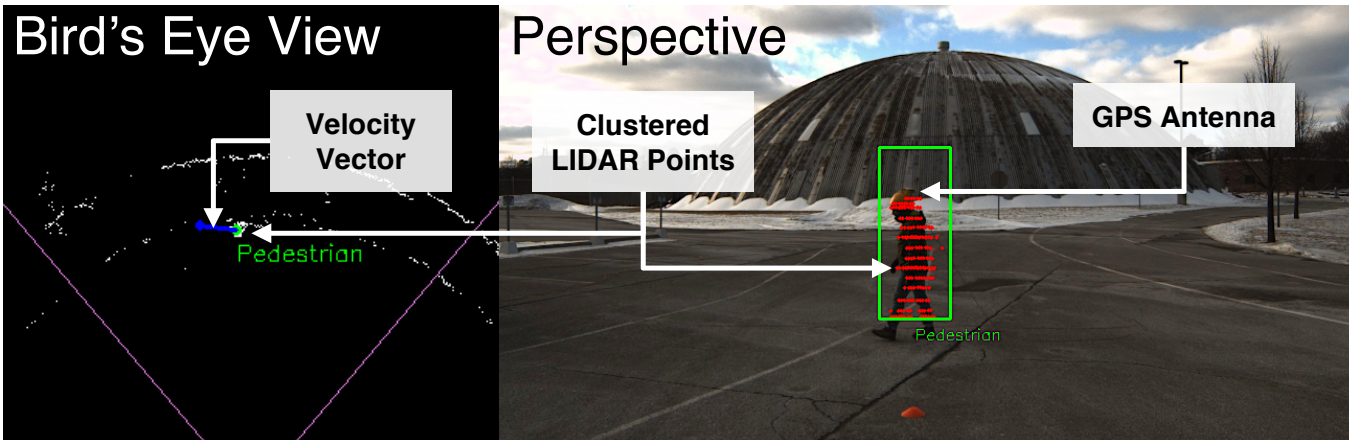

Figure 3. Example of a pedestrian track on the UofTPed50 dataset. On the left, a bird's eye view with LIDAR points projected to the ground. On the right, a perspective view from the camera.

pedestrian, we use UTC timestamps. A minor issue observed during the data collection process is that the two GPS units have a constant offset. To correct this, we estimate the centroid of the pedestrian at the beginning of each sequence using LIDAR, and use it to estimate the offset between the two GPS frames. We intend to rectify this issue before making our dataset publicly available.

There are some key differences between UofTPed50 and the KITTI dataset. First, on KITTI object tracking can only be benchmarked on the 2D image plane. Although there are 3D labels for pedestrians, the multi-object tracking benchmark does not include this information. In UofTPed50, we collect the 3D global position of the ground truth from GPS, which is more consistent in its error. Second, as illustrated in Figure 4, the KITTI dataset has a narrow distribution of pedestrian distances with nearly 25% of all labelled pedestrians being roughly 10m from the car. Despite having a smaller range of distances, UofTPed50 has a more uniform distribution. Finally, most pedestrian sequences in the KITTI dataset contain pedestrians tracking a constant heading. In UofTPed50, we collect sequences with the pedestrian tracking complex curved and Zig-Zag trajectories.

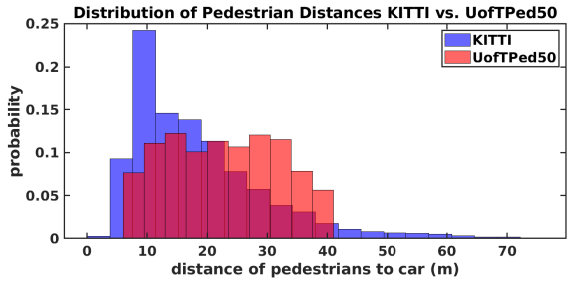

Figure 4. Distribution of pedestrian distances in the KITTI dataset and in the UofTPed50 dataset.

IX. Experiments:

In this section, we describe the experimental evaluation we have performed on UofTPed50, as well as on the KITTI Object Tracking benchmark. We have divided our evaluation into several subsections. First, we demonstrate the performance of our approach with varying target depth. Second, we show the impact relative motion can have on the accuracy of our position and velocity estimates. Third, we compare the performance of our approach on different trajectory shapes. We also discuss the performance of our approach using several qualitative examples that compare the pedestrian trajectory generated by aUToTrack against the ground truth GPS-based trajectories. We present a qualitative analysis of our velocity tracking performance on several scenarios. We describe the accuracy with which we can assess whether or not a pedestrian is static. Finally, we briefly analyze our performance on the KITTI Object Tracking benchmark.

A. *Varying Distance*

The first set of sequences in UofTPed50, the pedestrian walks laterally from one side of the vehicle to the other at evenly spaced distances. Figure 6 illustrates an example of a lateral trajectory. We use this to simulate a pedestrian crossing the road immediately in front of the vehicle. We need to ensure that position and velocity estimates remain accurate with increasing distance. Being able to accurately estimate the state of other traffic participants regardless of distance is key to the safety of an autonomous vehicle.

We use Root Mean Squared Error (RMSE) as our error metric for both position and velocity estimation. As summarized in Table 4, our position and velocity estimation error tends to increase with distance. The position error increases from 0.14m at 5m to 0.37m at 35m. This appears to be an acceptable increase in error given the range increase. Although we are able to achieve high velocity estimation accuracy up to 30m, there appears to be a steep drop

off in performance at 35m. This is potentially due to the increasing sparsity of points further from the LIDAR. We have also observed that 30m is close to the detection range of squeezeDet trained on KITTI, another potential cause of the drop in performance.

Table II
POSITION AND VELOCITY ESTIMATION ERROR VS. TARGET DISTANCE

Target Distance(m)	**5**	**10**	**15**	**20**	**25**	**30**	**35**
Position RMSE (m)	0.14	0.18	0.21	0.26	0.22	0.27	0.37
Velocity RMSE (m)	0.20	0.19	0.18	0.23	0.32	0.29	0.55

B. Varying Relative Motion

In these experiments, the pedestrian is either walking straight towards or away from the vehicle. We repeat the pedestrian trajectories with the vehicle stationary and driving forward. These trajectories are used to simulate a pedestrian walking along a sidewalk. It is important to distinguish between pedestrians moving laterally across and longitudinally along the road. Table III summarizes the results of this experiment. We note that when the vehicle is driving forward (V > 0) and the pedestrian is walking towards the vehicle, position error increases. In general, high sensor vehicle velocity introduces several complications, such as pointcloud distortion and sensor message misalignment. Thus, our expectation is that error should increase with increasing relative motion. However, it is surprising to see that velocity estimation error remains relatively constant in both cases regardless of vehicle motion. For scenarios where the pedestrian is walking away, relative velocity decreases. As such, we observe that both the position and velocity error also decrease.

Table III
POSITION AND VELOCITY ESTIMATION ERROR VS. RELATIVE MOTION

Pedestrian Motion	Walk Towards		Walk Away	
Vehicle Motion	V = 0	V > 0	V = 0	V > 0
Position RMSE (m)	0.27	0.51	0.31	0.27
Velocity RMSE (m/s)	0.21	0.19	0.26	0.23

C. Varying Trajectory Shape

In these experiments, our goal is to push our system in order to find corner cases where performance declines. In the other sequences, the pedestrian motion follows a relatively constant heading. Here, we demonstrate trajectories that have more changes in velocity and directionCurved trajectories involve the pedestrian jogging in a shape similar to a parabola. Figure 5 illustrates an example of a Zig-Zag trajectory.

Wait trajectories involve the vehicle driving forwards, stopping and then waiting for the pedestrian to cross. Table IV summarizes the results of this experiment. As expected, the straight-line trajectories (Across, Toward, Away) tend to have the lowest tracking error due to their simplicity. Interestingly, position error remains relatively low during the Curve, Zig-Zag, and Wait trajectories. This can be due to the fact that relative motion remains low. Velocity estimation error increases for the more complex trajectories, but is highest for Zig-Zag. We anticipate this is caused by lag in our current estimator. Estimator parameters were tuned to compromise between lag and smoothness.

Table IV
POSITION AND VELOCITY ESTIMATION ERROR VS. SCENARIO TYPE

Scenario	Curve	Zig-Zag	Across	Toward	Away	Wait
Position RMSE(m)	0.32	0.26	0.21	0.27	0.31	0.35
Velocity RMSE(m/s)	0.51	0.59	0.18	0.21	0.26	0.46

D. Qualitative Analysis

Figures 6, 5, 7, 8 directly compare the trajectories and velocities estimated by aUToTrack against the GPS-based ground truth in our dataset. Our first observation is that our approach is capable of replicating the reference position trajectories with high accuracy. However, one can also observe that the position and velocity estimation appears to overshoot and somewhat lag behind the ground truth. This is likely caused by estimator dynamics, and can be remedied with parameter tuning. Even though the Zig-Zag velocities shown in Figure 8 change rapidly, we are still able to track the velocity with respectable accuracy. Figure 7 shows a similar story, where the pedestrian abruptly changes velocity, challenging the tracker to keep up.

Overall, we are please with the performance of aUToTrack on our dataset, although there is clearly room for improvement. Future work will include improving the velocity tracking in challenging scenarios, and boosting the position accuracy at 35m.

E. KITTI Object Tracking

Table V summarizes the performance of aUToTrack on the KITTI Object Tracking test set. As of writing, we rank among the top five published works. The metrics used in

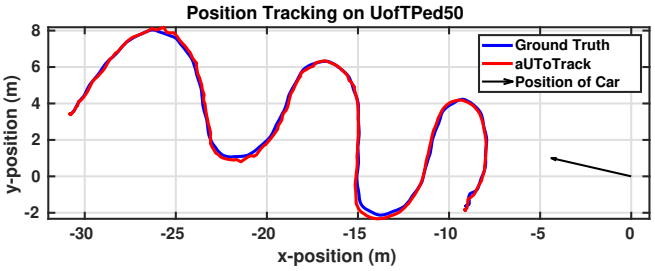

Figure 5. Zig-Zag scenario position tracking. The pedestrian starts roughly 30m away, then follows a Zig-Zag trajectory towards the stationary car.

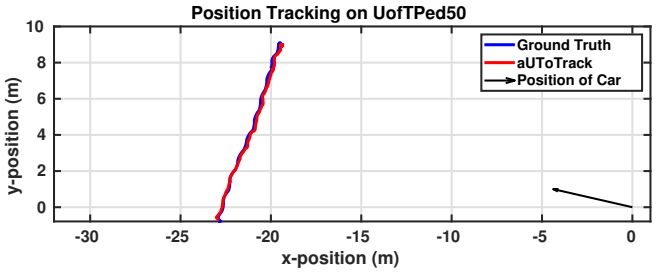

Figure 6. Straight Scenario Position Tracking. The pedestrian starts roughly 20m away on the left of the car, then follows a straight trajectory laterally.

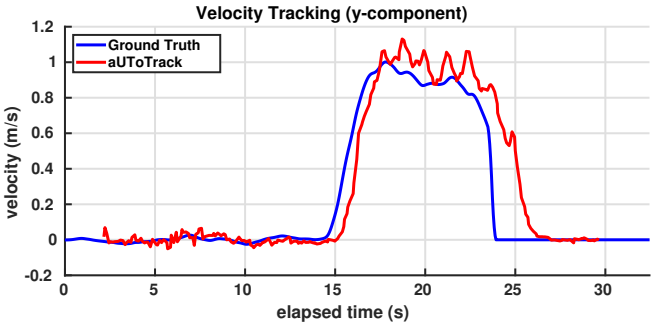

Figure 7. Wait scenario velocity estimation. For the first 15 seconds, the car approaches a stopped pedestrian and stops. The pedestrian then crosses laterally and stops.

this benchmark are defined as the Clear MOT metrics from [21]. Our approach is very simple: we use LIDAR data and 2D bounding boxes to estimate the 3D position of objects and then we employ baseline data association and tracking techniques. Despite this simplicity, we are able to achieve quite competitive performance, while tieing for highest framerate among the top five published works. It should be noted that we are only able to achieve state-of-the-art performance when using Recurrent Rolling Convolutions (RRC). Since we are working on a tracking benchmark, it seemed appropriate to use a very good detector.

We note that our MT metric ranks quite highly. Our interpretation of this result is that tracking objects in 3D is simply not as difficult as tracking objects within a 2D image plane. Since our system possesses a 3D estimate of the location and velocity of all objects, we are able to predict the motion of objects more reliably than vision-based approaches. We also note that our FRAG and IDS metrics do not do as well. This is easily attributed to the fact that our data association relies purely on the locations of objects in 3D space. We believe that using image features for data association could dramatically reduce these numbers. However, a more complex data association step would likely also increase runtime.

Table VI, compares our results on the training set when using different detectors, with and without our 3D clustering

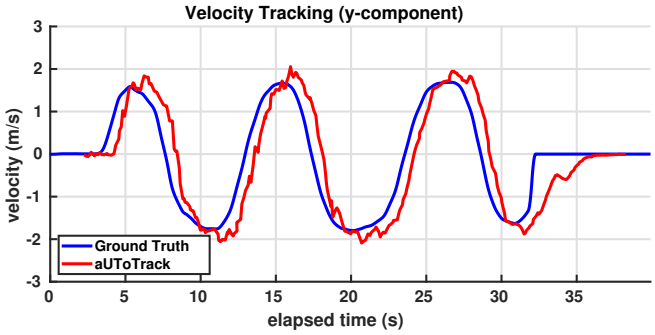

Figure 8. Zig-Zag scenario velocity estimation, corresponding to Figure **??**. The pedestrian starts roughly 30m away, then follows a Zig-Zag trajectory towards the stationary car.

information. The results show that choosing the best detector has by far the largest impact on performance on these datasets. Thus, in order to compete with the other approaches listed in Table V, a powerful detector is required. Table VI also show that our addition of 3D information via clustering has a substantial improvement on the MOTA and Precision, a modest improvement to MT, and negligible impact on the other metrics.

Table V
RESULTS ON THE KITTI OBJECT TRACKING TEST SET

	MOTA↑	MOTP↑	MT↑	ML↓	IDS↓	FRAG↓	FPS↑
[4]	**86.6**	84.0	72.5	6.8	293	501	1.7
[5]	84.2	**85.7**	**73.2**	**2.8**	468	944	3.3
[6]	83.0	82.7	60.6	11.4	172	**365**	5.3
Ours	82.3	80.5	72.6	3.5	1025	1402	**100**
[7]	80.4	81.3	62.8	6.2	**121**	613	**100**

Table VI
ABLATION STUDIES: NETWORK COMPARISON, USE OF CLUSTERING ON KITTI OBJECT TRACKING TRAINING SET

	MOTA↑	MOTP↑	R↑	P↑	MT↑	ML↓	FPS↑
SqueezeDet +Clustering	48.8	78.2	64.9	85.6	31.4	25.7	**50**
SqueezeDet	46.0	78.7	65.1	83.5	31.0	26.1	**50**
RRC +Clustering	**84.9**	79.7	94.3	**95.0**	**87.2**	**1.9**	0.7
RRC	80.2	**80.0**	**94.4**	92.6	87.0	**1.9**	0.7

X. CONCLUSION

In this paper, we introduced the UofTPed50 dataset[1], an alternative to KITTI for benchmarking 3D Object Detection and Tracking which we will be making publicly available. The UofTPed50 dataset offers precise ground truth for the position and velocity of a pedestrian in 50 varied and challenging scenarios – something that is currently unavailable anywhere else. We also described our approach to the problem of 3D Object Detection and Tracking – aUToTrack.

[1]We plan on releasing the dataset in June 2019

We use vision and LIDAR to generate raw detections and use GPS/IMU measurements to track objects in a global metric reference frame. We use an off-the-shelf 2D object detector paired with a simple clustering algorithm to obtain 3D position measurements for each object. Given this 3D information, we then use simple data association and filtering techniques to obtain competitive tracking performance. We demonstrate state-of-the-art performance on the KITTI Object Tracking public benchmark while showing that our entire pipeline is capable of running in less than 75 ms on CPUs only. Our future work will include bolstering our velocity estimation on UofTPed50 and reducing the error in our position estimates at ranges exceeding 35 m.

REFERENCES

[1] K. Burnett, A. Schimpe, S. Samavi, M. Gridseth, C. W. Liu, Q. Li, Z. Kroeze, and A. P. Schoellig, "Building a winning self-driving car in six months," *arXiv preprint arXiv:1811.01273*, 2018.

[2] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[3] J. Ren, X. Chen, J. Liu, W. Sun, J. Pang, Q. Yan, Y.-W. Tai, and L. Xu, "Accurate single stage detector using recurrent rolling convolution," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[4] W. Choi, "Near-online multi-target tracking with aggregated local flow descriptor," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 3029–3037.

[5] S. Sharma, J. A. Ansari, J. K. Murthy, and K. M. Krishna, "Beyond pixels: Leveraging geometry and shape cues for online multi-object tracking," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 3508–3515.

[6] A. Sadeghian, A. Alahi, and S. Savarese, "Tracking the untrackable: Learning to track multiple cues with long-term dependencies," in *International Conference on Computer Vision (ICCV)*, 2017.

[7] S. Scheidegger, J. Benjaminsson, E. Rosenberg, A. Krishnan, and K. Granström, "Mono-camera 3d multi-object tracking using deep learning detections and pmbm filtering," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2018, pp. 433–440.

[8] H. Somerville. (2019). Gm's driverless car bet faces long road ahead, [Online]. Available: https : / / uk . reuters . com / article / uk - gm - selfdriving - cruise - insight - idUKKCN1MY0CQ (visited on 02/18/2019).

[9] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[10] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*, Springer, 2016, pp. 21–37.

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[12] B. Wu, F. Iandola, P. H. Jin, and K. Keutzer, "Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 129–137.

[13] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3d object detection from rgb-d data," *arXiv preprint arXiv:1711.08488*, 2017.

[14] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander, "Joint 3d proposal generation and object detection from view aggregation," *CoRR*, vol. abs/1712.02294, 2017. arXiv: 1712.02294. [Online]. Available: http://arxiv.org/abs/1712.02294.

[15] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *Image Processing (ICIP), 2016 IEEE International Conference on*, IEEE, 2016, pp. 3464–3468.

[16] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler, "Motchallenge 2015: Towards a benchmark for multi-target tracking," *arXiv preprint arXiv:1504.01942*, 2015.

[17] F. Moosmann and C. Stiller, "Joint self-localization and tracking of generic objects in 3d range data," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, IEEE, 2013, pp. 1146–1152.

[18] A. Dewan, T. Caselitz, G. D. Tipaldi, and W. Burgard, "Motion-based detection and tracking in 3d lidar scans," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2016, pp. 4508–4513.

[19] Intel. (2019). Intel distribution of openvino toolkit, [Online]. Available: https://software.intel.com/en-us/openvino-toolkit (visited on 02/18/2019).

[20] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 2011.

[21] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: The clear mot metrics," *Journal on Image and Video Processing*, vol. 2008, p. 1, 2008.