Pedestrian Motion Classification for Autonomous Vehicles (6.867 Final Project)

Michael Everett & Björn Lütjens

(a) Ground robot in Stata

(b) Golf carts (MIT MoD fleet)

Fig. 1: Many types of autonomous vehicles operate among pedestrians and must account for pedestrian motion in the vehicle motion planning algorithms.

***Abstract*— TODO**

I. Introduction

Autonomous vehicles use onboard sensors to perceive their surroundings, and use the data to make decisions about where it is safe to drive. Today's research vehicles typically rely on Lidar and cameras as the main sensors, as these provide information about where/what obstacles are, and can help the vehicle maintain safe position in its lane and with respect to obstacles. In addition to static obstacles, vehicles interact with pedestrians in a variety of environments: people in crosswalks, jaywalkers, kids running across neighborhood streets. A campus shuttle could even drive on sidewalks among pedestrians, so the vehicle would be interacting with pedestrians almost all the time. These regular interactions mean the vehicle must be able to predict pedestrians' next moves in order to maintain safety.

Pedestrian motion prediction is difficult because people often behave very unexpectedly. Humans use many strategies to predict pedestrian intent while driving, such as body language, eye contact, and other non-verbal cues between human driver and human pedestrian, but autonomous vehicles can't easily communicate or read these signals. The information from sensors is typically fused: Lidar is used to measure the pedestrian position, and the camera is used to label the particular obstacle as a pedestrian. Therefore the only measurements collected are position over time (from which velocity can also be computed).

This project uses a dataset collected on MIT's campus over several months by three golf cart shuttles providing Mobility on Demand service to students, while simultaneously collecting pedestrian trajectory data to optimize vehicle routing strategy. The dataset contains about 65,000 pedestrian trajectories as well as the vehicles' trajectories, all in a global frame across the MIT campus.

M. E. **related works**

The objective of this project is to develop a classifier that can determine whether a person will step in front of a vehicle, based on a few seconds of their trajectory. We use a portion of our data set to train different classifiers, optimize hyperparameters with a separate portion, and evaluate performance with yet another portion. This classifier could be a useful component of an autonomous vehicle, or part of an active safety feature on a human-driven vehicle that could take over in case the driver does not see a pedestrian in time.

The main contributions of this work are (i) a pedestrian trajectory dataset with 65,000 trajectories over three months, (ii) an SVM classifier for predicting when pedestrians will step in front of a vehicle, (iii) a classifier using Deep RNNs for that same objective, and (iv) a pedestrian motion predictor using Deep RNNs.

II. Dataset

An important realization we made during this project is the challenge of working with a real dataset. Understanding the raw data became a significant portion of the project, so this chapter provides a thorough explanation of our findings and methods to process the data.

A. Raw Data

Our data comes from golf carts equipped with Lidar and cameras [1], [2]. Fortunately, it is relatively straightforward to visualize trajectory data, as opposed to some high-dimensional datasets that exist for other applications.

The raw dataset's fields are shown in Table I, where (easting,northing) are the latitude/longitude global coordinates, and (x,y) are the coordinates in our global campus map. Veh id indicates which of the three vehicles corresponds to that data point, or in the pedestrian case, which vehicle sensed that pedestrian. Ped id is a unique id given to each pedestrian seen.

Type	Fields						
Vechicle	time	easting	northing	x	y	veh id	
Pedestrian	time	easting	northing	x	y	veh id	ped id

TABLE I: Raw data fields

There are some noise-related issues with the raw data, as it was collected on a research vehicle under development. One issue is that the vehicle's (x,y) position sometimes jumps, because the vehicle's localization system does not use GPS and is imperfect. Other minor issues include pedestrian trajectories that incorrectly split/merge or are too short to be useful for this project.

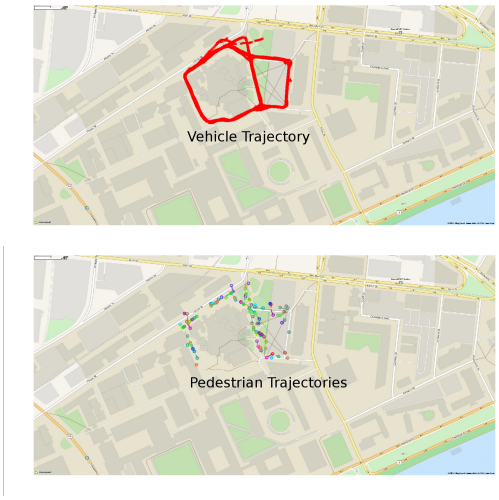

Fig. 2: The raw trajectories from one day of data collection visualized on a campus map. All trajectories are recorded in the global map coordinate frame.

In addition to addressing noise, we also pre-process the data by converting pedestrian trajectories into the vehicle's local frame. Our objective is to classify whether pedestrians cross in front of the vehicle, so the pedestrian trajectories must be converted into the vehicle's local frame. That is, our dataset is initially unlabeled, and we must generate the ground truth label that we wish to learn to classify later.

It might be possible to somehow feed both the vehicle trajectory and pedestrian trajectory into a classifier, and have it learn the transformation, but it seemed much more straightforward for us to rotate the data ahead of time. Using global coordinates could potentially allow the classifier to learn a global understanding of the map (e.g. where sidewalks/intersections are), but we chose to focus on a more structured problem for this project. Since global coordinates should have an impact on pedestrian motion, we could in the future try a hybrid approach where we feed local coordinates along with some context features (e.g. distance to curb, traffic light state) as in [3].

B. Global-to-Local Transformation

The global-to-local transformation relies on knowledge of vehicle orientation (heading angle) and smooth vehicle trajectories, neither of which we have by default. Line 1 describes the procedure for filtering, transforming, and labeling the raw dataset. M. E. **explain fig 3** M. E. **explain how to do global-to-local transform**

C. Processed Dataset

The processed dataset is visualized in Fig. 4. The vehicle (yellow taxi) has a blue rectangle representing the "cross" zone (similar to Fig. 3). Green trajectories are local pedestrian trajectories that do not cross into the blue zone, and red trajectories are ones that do enter the blue zone.

A few important observations about the processed dataset are the jaggedness and locations of trajectories. The jaggedness is due to the transformation and sensor rates: the Lidar

Algorithm 1: Algorithm for extracting local trajectories

Input: V_g, P_g: global vehicle, pedestrain trajectories (Table I)

Output: P_l: pedestrian trajectories in local vehicle frame

1: **for** each vehicle **do**

2: $I_{posjump} = \{i \in [1, len(V_g) - 1] \mid euclid\_dist(V_g(i) - V_g(i+1)) > 1.0\}$

3: $I_{timejump} = \{i \in [1, len(V_g) - 1] \mid time(V_g(i) - V_g(i+1)) > 0.5\}$

4: J = $I_{posjump} \cup I_{timejump}$

5: $T_{valid} = \{[J(i), J(i+1)] \mid time(J(i+1) - J(i)) > 5.0\}$

M. E. **finish**

6: **return** pedestrian trajectories

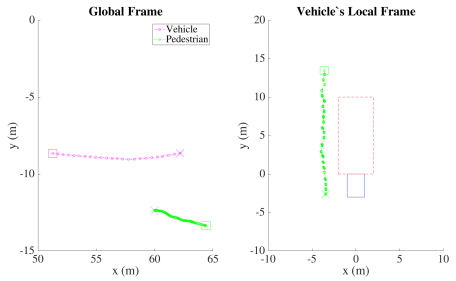

Fig. 3: On the left, the vehicle trajectory (magenta) and pedestrian trajectory (green) in the global frame. Both start from the square and move to the X. On the right is the local vehicle frame: the vehicle is the blue rectangle, and the red dashed rectangle is the "cross" zone. The green pedestrian trajectory doesn't cross into the red rectangle, so this trajectory is given binary label 0.

provides pedestrian trajectory at high rate (50 Hz), but the vehicle position is updated at a slower rate (10 Hz), so the pedestrian trajectory jumps a bit each time vehicle time step. Since the rotation is dependent on vehicle heading angle, and vehicle heading angle is computed from consecutive vehicle positions, this process is subject to some noise. It is also somewhat difficult to imagine what local trajectories *should* look like, because these trajectories are showing how the vehicle and pedestrian move relative to one another, and an observer doesn't know what the vehicle's velocity was for any particular trajectory. For example, a trajectory that arcs along the front of the vehicle could be the vehicle turning as a pedestrian walks straight, or a pedestrian walking in a curve while the vehicle is parked. All of this could have been avoided if we had access to raw sensor measurements, as these provide local coordinates by default.

The second observation is that the trajectories are in front/along the sides of the vehicle, and have a limited range of about 20m. This makes sense given our sensor: it is mounted on the vehicle's front and sees a 270° cone. Its maximum range is more than 20m, but tracking pedestrians reliably is difficult beyond this distance.

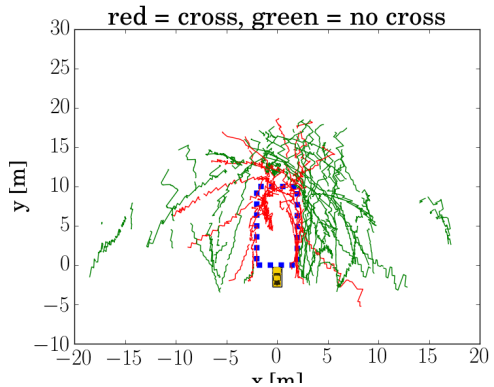

Fig. 4: A small portion of the training set for the binary classifier. The vehicle (yellow taxi) has a blue rectangle representing the "cross" zone. Green trajectories are local pedestrian trajectories that do not cross into the blue zone, and red ones do.

ACKNOWLEDGMENT

Michael mostly worked on the dataset processing and SVM optimization, and Björn did the RNN work and the rest of the SVM work. This project is not used for any other classes. Our code can be found at: https://github.com/mfe7/6.867.

REFERENCES

[1] J. Miller, A. Hasfura, S.-Y. Liu, and J. P. How, "Dynamic arrival rate estimation for campus mobility on demand network graphs," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[2] J. Miller and J. P. How, "Predictive positioning and quality of service ridesharing for campus mobility on demand systems," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[3] Anonymous, "CASNSC: A context-based approach for accurate pedestrian motion prediction at intersections," https://openreview.net/pdf?id=rJ26HSLRb, 2017, [Online; accessed 12-Dec-2017].