Autonomous Driving: Mapping and Behavior Planning for Crosswalks

by

Edward Chao

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Computer Engineering

Waterloo, Ontario, Canada, 2019

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

As autonomous driving integrates with every day traffic, early adopters are initially skeptical and designers are overly cautious. With safety as the top priority, current systems are sometimes too slow at executing maneuvers. Scenarios such as switching into a crowded lane or waiting for a left turn can result in the autonomous system to wait much longer than a human driver would. This behavior can be frustrating for passengers and confusing for other drivers around. Acceptable driving style also depends on other context like location and culture. A driver may be more forceful in a densely populated city compared to a calmer driver from the suburbs.

This thesis explores the unsignalized pedestrian crosswalk scenario and methods that balance safety, assertiveness, caution, and obstruction of traffic flow when interacting with pedestrians. A configurable driving policy for the Autonomoose[1] system is introduced with results. The work adopts the lanelet [5] mapping format and introduces a method of mapping and representing the crosswalk regulation. The main contribution of the work is a tunable algorithmic approach for progressing through unsignalized crosswalks that exemplifies both conservative and assertive driving behavior.

The algorithm described in this work is one of possibly infinitely many methods for handling unsignalized crosswalks. Reinforcement learning based solutions and other hand crafted algorithms can benefit from using the work proposed as a point of comparison. General concepts proposed in the algorithm may inspire more robust algorithms in future development.

[1] Self driving research platform at the University of Waterloo https://www.autonomoose.net/

Acknowledgements

This thesis would not have been possible without the collaboration of the entire Autonomoose team. Members of both WISE[2] and WAVE[3] have contributed significantly toward a successful public drive.

Firstly, I would like to thank Professor Krzysztof Czarnecki my supervisor for granting me the opportunity to work with the team and guiding my learning. I would also like to acknowledge the team members that I have worked with directly. Sean Sedwards, Changjian Li, and Atrisha Sarkar for bouncing ideas off of and inspiring new ones. Divit Sharma for his impressive assistance with building the map server. Matthew Pitropov, Carlos Wang, Kevin Lee, and Danson Garcia for their expertise with in-vehicle testing. Frederic Bouchard for developing and adapting the rule engine. Marko Ilievski, Ashish Gaurav, and Aravind Balakrishnan for building the foundation of the behavior planner. Michal Antkiewicz, Rodrigo Queiroz and all the coop students for all their work in simulation and assistance. Nikolas Stewart for his excellent job as project manager, keeping the team moving together toward a common goal.

[2]Waterloo Intelligent Systems Engineering Lab
[3]Waterloo Autonomous Vehicles Engineering Lab

Dedication

This is dedicated to cold iron for being a light in the darkest of times.

Table of Contents

List of Figures

Glossary

autonomoose Autonomous driving research platform at the University of Waterloo, Ontario, Canada 7, 8

CAN Controller Area Network 16

DSBEE Pedestrian distance-speed combination before entry edge of roadway ix, 47, 51

ego The autonomous vehicle that the stack operates ix, 7, 11–14, 19, 21, 23, 24, 27, 29, 36–44, 54

GNSS Global Navigation Satellite System 61, 62

LIN Local Interconnect Network 16

node The atomic element of a lanelet. A minimal node defines a latitude and longitude 7, 10, 61

online Something that is executed on-the-fly, processing mostly immediately available information; not to be confused with the internet 6

osm OpenStreetMap is a collaborative project to create a free editable map of the world 7, 38, 61

PDAEE Pedestrian distance after entry edge of roadway 46

PDFLE Pedestrian distance from leaving edge of roadway 47

way An ordered list of nodes 7, 19

xml eXtensible Markup Language designed to store and transport data 7

Abbreviations

GPS Global Positioning System 54, 61, 68

GPX GPS Exchange Format 10, 65

JOSM Java OpenStreetMap Editor viii, 9, 10, 20, 38, 64, 65, 67

LIC Lane Indicator Counters 62, 63, 65–67

LIDAR Light Detection and Ranging 4, 5, 68

MTO Ministry of Transportation of Ontario 2, 17, 23, 41, 42, 44

RADAR Radio Detection and Ranging 5

ROS Robot Operating System 10, 36, 63

RTK Real Time Kinematics 62

SAE Society of Automotive Engineers 1

SLAM Simultaneous Localization and Mapping 6, 54

WGS 84 World Geodetic System 61

Chapter 1

Introduction

Autonomous robotics have proved to be successful in smaller scoped, specialized, and niche applications. Robots are conventionally designed to perform a limited set of tasks extremely well, repeatedly over an extended period of time. One of the challenges of extending the capabilities of a specialized robot to handle more complex tasks like driving a car is the vast expansion of scope that it would have to consider and overcome. The possible scenarios an autonomous vehicle can encounter is limited by the imagination. Attempts at creating intelligent robots throughout history have failed to match the performance and intelligence of humans. Recent advances in machine learning [1] and continued innovation in computer hardware have opened up the floodgates of autonomous driving research. Particularly, machine learning demonstrations have proved to be successful in areas such as perception and decision making. These demonstrations impart confidence in researchers and inspire funding toward further research efforts.

Safety and ethics have always been discussed around the topic of robotics and interaction with humans. It is especially important with autonomous cars due to the close proximity of passenger and machine. Since so many people depend on reliable transportation on a daily basis, the impact of safety scales proportionately. As autonomous vehicle technology matures, legislators are now examining more seriously the rules and regulations around it and how it will impact our everyday lives. [4, 18, 47]. Society of Automotive Engineers (SAE) have developed the 6 level system of automation for categorizing autonomous driving systems. The J3016 standard levels of automation first introduced in 2016 is constantly under revision as the autonomous driving industry evolves. As it currently stands, the 6 levels of automation are summarized in figure 1.1. In level 3, the person sitting in front of the wheel must take control when the system requests. The autonomous system described in this thesis most closely resembles level 3 autonomy.

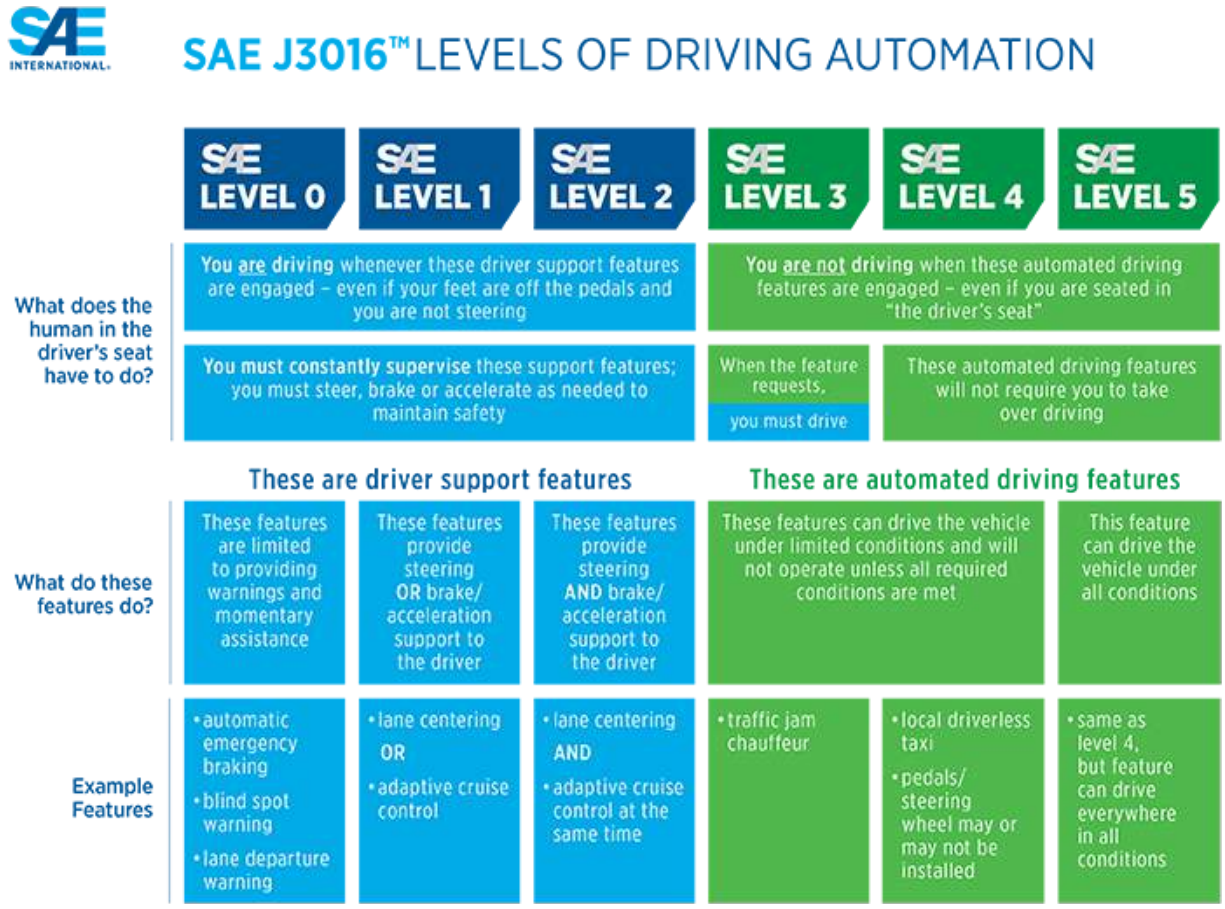


Figure 1.1: 6 levels of automation defined by SAE

With safety as the top priority, current systems are sometimes too slow at executing maneuvers. When attempting to switch lanes for example, an autonomous vehicle may exhibit hesitation that confuses other drivers and slows traffic. It can also cause the failure of following predetermined routes causing delay for passengers. Interactions with pedestrians at busy crosswalks can cause extended waiting and traffic congestion. On the contrary, other situations call for a more conservative driving style. Autonomous driving systems may need to be more conservative with passengers that are inexperienced to gain their trust. Drivers can also be expected to be more cautious in school zones for example.

Following chapters touches on a couple of essential subsystems associated with maneuvering around crosswalks, pedestrians, and general guidance created by the Ministry of Transportation of Ontario (MTO). Specifically, this thesis focuses on the mapping and

behavior planning subsystems under the context of interaction with pedestrians at unsignalized crosswalks. Following other vehicles when approaching a crosswalk and waiting behind them are not included in the scope. In section 2, a general overview of an autonomous driving system colloquially referred to as the *stack* is presented. A more detailed introduction to mapping explains how the road environment is represented in section 2.1.2. One of the contributions in this section is a method of generating small scale lanelet maps. Closely integrated with mapping is behavior planning and is introduced in section **??**. The stack as a whole engages pedestrians and crosswalks in section 3. this section contains the main contribution; the tunable driving policy for navigating through crosswalks. Simulation results based around Unreal Engine and real world data based testing is discussed in section 4.

Chapter 2

Background

2.1 Autonomous Stack

The autonomous stack refers to the architectural design of the system. Like in other large hardware/software system, there are multiple viable options and selection of one is usually associated with making trade offs. As more research effort continues in autonomous driving, a common pattern and general consensus on what defines a conventional stack has emerged [32, 28, 47]. Fundamentally, the stack must perceive its surroundings, understand where it is located, and safely command the mechanical system to execute missions safely and comfortably[1]. An example of a mission could be to take the passengers from A to B without breaking any rules of the road, while minimizing distance travelled. The following sections introduce the main subsystems of a stack as a precursor for handling crosswalks.

2.1.1 Perception and Tracking

The goal of perception and tracking is to gain an understanding of the immediate domain in which the system operates. It attempts to identify and classify features of the environment that are important for successfully executing missions. Some examples of features include pedestrian location and velocity, cyclists, and other miscellaneous obstacles.

Methods of perception are varied with each having their own benefits and disadvantages. A Light Detection and Ranging (LIDAR) sensor for example has the ability to directly sense

[1] comfort is subjective

the location including distance of an obstacle in 3 dimensions. It also has the ability to operate in any lighting condition. A limitation of LIDAR is the ability to perceive color. Because of this, standard visible light spectrum cameras are complementary to LIDAR. Other sensors include Radio Detection and Ranging (RADAR) and ultrasonic. Figure 2.1 below shows the coverage of the sensor suite including 8 cameras, 12 ultrasonic sensors for detecting hard and soft obstacles, and a forward facing radar that enables Tesla's Autopilot.

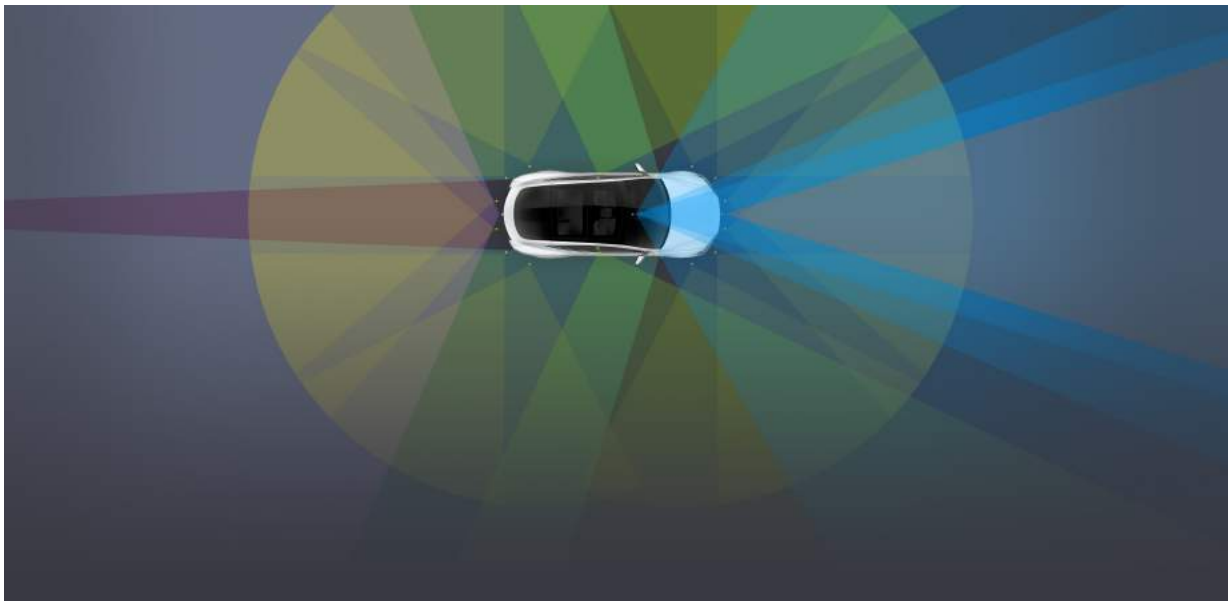


Figure 2.1: Sensor coverage on a Tesla Model 3 [27]

Naturally, instantaneous detection is insufficient for the task of driving a car. It is critical to observe how features progress throughout time in order to make proactive decisions. The tracking module processes detections and produces a pseudo-continuous temporal understanding of how features advance through time. For example in a scene of 2 pedestrians walking, the tracking module repeatedly takes instantaneous detections of the pedestrians and labels each one by a unique identifier. Their identifiers are locked on to each pedestrian throughout the sequence. The tracker may also provide secondary information such as velocity or acceleration and prediction of future trajectory.

2.1.2 Mapping and Localization

The purpose of mapping is to consolidate relevant environment features into a format that is readily consumable by the rest of the system. Where perception captures dynamic ele-

ments, mapping aims to capture static elements that are unlikely to change over time. A map may contain relevant details such as the location and category of road markings, location of curbs, location of traffic signs, and other regulatory elements [12]. Fundamentally, mapping represents the world as a high definition abstraction that only carries relevant information. Depending on system architecture, the level of detail can vary. Typically, global mapping aims to store information about traditionally static[2] road features. As a result, the map is generally understood to be unchanging or at the very most, infrequently. Localization compares the immediate surrounding with features in the map to probabilistically identify where the vehicle is. This form of localization depends on having a preexisting map on which it can compare measurements.

Another form of mapping and localization is commonly known as Simultaneous Localization and Mapping (SLAM) [8]. As the name suggests, this type does not necessarily depend on a prepared map. It attempts to build a map of the environment as it explores it for the first time. This for example is analogous to a person finding his way through a building that he has never been in before. The layout of the building is remembered so that he knows when he has returned to an area already seen before. The level of planning is local i.e. go straight as to not walk into a wall as opposed to a higher level planning i.e. go upstairs to get to the lunchroom. Since SLAM is an online process, it consequently maps dynamic features of the environment. For example, SLAM on an autonomous vehicle would include location of traffic cones, temporary barriers, and other unexpected elements that may affect behavior planning.

The distinction made here is to clarify that the mapping discussed in later chapters refer to the former global mapping rather than SLAM.

Lanelet Maps

Originally developed by Bender et al.[5] and further refined [40], the lanelet structure defines an efficient human readable representation of the static environment. By design, the format naturally allows for both geometrically and topologically accurate maps. Its simplicity makes extensions easy and intuitive. Details can be customized for extraneous scenarios. In chapter 3, the original lanelet format is extended to include crosswalks. Processes developed for generating small-scale lanelet maps is discussed in further detail in appendix A.

Lanelets have been proven in the autonomous driving domain, and the format was the cornerstone of the Bertha drive [51] with over 100 kilometers, intersections with and

[2] colloquially fixed throughout time

without traffic lights, roundabouts, narrow sections, through 23 small villages and major cities.

Nomenclature

Since the lanelet format was inspired partly by the osm format [22], there are some commonalities between the two. The atomic element of the lanelet format is a node. A node is a single point that specifies a position in the world described by a latitude and longitude coordinate. Additional properties can be tagged on a node to augment interpretation. Elevation or altitude is an example of an additional property that is often tagged on nodes. A node with elevation represents a position in 3 dimensions and therefore can be referenced by multiple coordinate frames. More informed decisions can be made by knowing elevation. For example, rising elevation along a route can be identified as a hill climb and driving behavior can accommodate for the grade. An accurate acquisition of elevation is difficult because GPS measurement is more suited for position along the earth's surface.

The lanelet map is represented in xml file format, and specifically follows the osm data format. An example of a node is shown in figure 2.2.

```
<node id='-1123941' lat='43.5014638934' lon='-80.5366403762'>
    <tag k='elevation' v='308.722194092' />
</node>
```

Figure 2.2: Node representation including elevation

Two ordered nodes define a line segment, which is referred to as the basic way. Ways are line segments composed of two or more ordered nodes. The ordering of nodes is a topological property and does not have any geometrical constraints. Because of this, ways can cross itself; however, self crossing ways are uncommon for representing road structure. Like nodes, additional properties can be tagged to ways if necessary. The maps used by the autonomoose system do not rely on additional properties on ways. Figure 2.3 is an example of a way in xml format. Since ways are defined by individual nodes and are hence piece wise linear, curvature is undefined. A method for finding the closest distance of a point to a way is introduced in the original lanelets publication. The approach interpolates tangential vectors along a way to achieve a pseudo-tangent. The magnitude of this pseudo tangent vector approximates the distance of a point to the way. It approximates the piece wise linear way as a smooth continuous line and computes the shortest distance of a point to that line. This can be used to determine for example the distance of the ego vehicle from

the edge of the road and hence facilitate lane centering. For the purpose of lane centering, autonomoose maps are augmented with explicitly defined center ways for all drive lanes.

```
<way id='-1145270'>
    <nd ref='-1123941' />
    <nd ref='-1132261' />
    <nd ref='-1132263' />
    <nd ref='-1132265' />
</way>
```

Figure 2.3: Way representation with reference to 4 nodes

The minimal lanelet is defined by two ways, conventionally named 'left' and 'right'. Lanelets can represent a multitude of different road elements. The most primitive is its use in representing drive lanes, although there is no restriction on how a lanelet can be used. As long as a road element can be represented by 2 polyline boundaries, it can be expressed as a lanelet. Examples of non drive lane lanelets include: bus stop areas, bike lanes, and road dividers. The semantic identity of a lanelet depends on the properties that are assigned to it. Lanelets can be as short as 1 meter or as long as many kilometers. The length of a lanelet depends on the properties that apply to it. Properties applied to a lanelet are constant and so affect ego's behavior throughout its entire length. An example of a property is the speed limit on drive lanes. When the speed limit changes, the lanelet must end and another lanelet must begin to reflect the change in property.

Figure 2.4 illustrates a drive lane lanelet, which is a longitudinal section of a drive lane. Driving direction is implied by the ordering of the nodes in the ways that are referenced by a lanelet. Since two lanelets with opposite driving direction can share their left way, the ordering of nodes in the shared left way is not used. This often appears when there exists two drive lanes in opposite direction on a two lane road. Right side driving roads in North America can reliably depend on the right side way node ordering for driving direction.

Connections effectively build the road network by referencing drivable transitions from one lanelet to another. Connections of a lanelet to another are represented as left, right, next, and previous members of the given lanelet. These connections define purely topological relations and do not represent legal transitions. For example, 2 adjacent lanelets of opposite direction can both refer to one another with left connection members. This does not mean a vehicle can legally lane change from one to the other, but rather to indicate that there is a lanelet present which is to its left. Lanelets may have multiple next and previous connections for representing many paths a vehicle can take. At a four way

```
<relation id='-1145345'>
    <member type='way' ref='-1145270' role='left' />
    <member type='way' ref='-1145255' role='right' />
    <member type='way' ref='-1145290' role='center' />
    <member type='relation' ref='-1145344' role='left' />
    <member type='relation' ref='-1145357' role='next' />
    <member type='relation' ref='-1145358' role='next' />
    <tag k='lane_type' v='driving' />
    <tag k='left_marking' v='none' />
    <tag k='name' v='-61099' />
    <tag k='right_marking' v='none' />
    <tag k='speed_limit' v='50' />
    <tag k='type' v='lanelet' />
</relation>
```

Figure 2.4: Complete lanelet representation

stop sign intersection for example, a lanelet entering the intersection can have three next lanelets: one for going straight, one for left turn, and one for right turn. Figure 2.5 shows an example of such a lanelet in the JOSM editor. The main lanelet is highlighted in red and the connecting lanelets are highlighted in purple. In this case, there are two next lanelets, one for straight and one for right turn.

Some road rules can be embedded in lanelets through property tags, but more elaborate rules require additional elements. Regulatory elements are different from lanelets in that they do not follow the conventional referencing of a left and right way. These elements are applied to lanelets through the use of member tags in the affected lanelet. A single lanelet can refer to multiple regulatory elements. An example of a road rule that is expressed by a regulatory element is a stop sign. The regulatory element is defined by a type as well as additional entities that determine where or when it is relevant. A node and a way that indicates the position of the stop sign as well as the position of the stop line, respectively, are used to represent a single stop regulation. Figure 2.6 is an example of a stop sign regulatory definition.

Additional regulatory elements such as pedestrian crosswalks, traffic lights, and speed bumps can be added to the lanelet map definition simply by referring to them in the lanelet on which it affects. Section 3 goes over the pedestrian crossings regulatory element in detail and how it is represented using the lanelet format.

Figure 2.5: Lanelet with connections shown in JOSM

Map Server ROS Node

The map server Robot Operating System (ROS) node is the main component responsible for providing the road network to other modules within the autonomous stack. The responsibilities of the map server also include mission planning, and route generation. Additionally the map server provides a service for querying the current lanelet that the vehicle is located in.

As mentioned earlier, in section 2.1.3 mission planning is associated with higher level route planning more similar than others to human level planning. Consequently, the mission plan is directly input from the user. The mission is specified in GPS Exchange Format (GPX) format and contains an ordered list of nodes. These nodes represent goal points and the order in how they should be reached. The nodes in the mission plan must be within the boundary of a lanelet polygon in order to localize it within the map.

The complete lanelet map is another input to the map server and is parsed into semantic elements. Next, left, right, and previous connection references in the map connect lanelets into a graph that can be searched using common graph search algorithms. Dijkstra's algorithm [15] is implemented to find the shortest path from one goal point to the next. Edges in the graph get their length directly from the length of the lanelet. This length can

```
<relation id='-1145343' visible='true'>
    <member type='node' ref='-1130597' role='position' />
    <member type='way' ref='-1145323' role='stop_line' />
    <tag k='regulation' v='stop' />
    <tag k='type' v='regulatory_element' />
</relation>
```

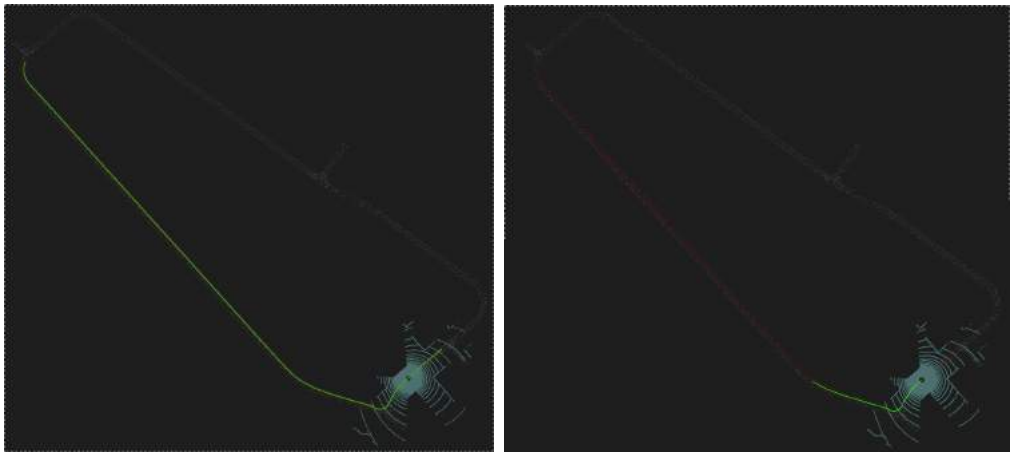
Figure 2.6: Stop sign xml representation

be augmented with additional information such as speed limit, or typical traffic congestion. Graph search algorithms are frequently improving, with recent proposals include ALT* [20], and REACH [21]. For the size and scope of the graphs that are used in this system, Dijkstra's algorithm is sufficient.

The global path is constructed by concatenating the center line points of lanelets found by Dijkstra's algorithm. The global path contains points from the very first lanelet to the very last lanelet. A semi global path contains only the next points within a configurable distance from ego's position. The purpose of the semi-global path is to prevent the path input to the motion/local planner from crossing itself. For example, if the global path resembles a figure 8 loop, the path crosses itself in the middle. The semi-global path would only make available enough points for motion planning but limited to prevent the path from crossing itself. Figure 2.7 shows the difference between global path and semi global path.

2.1.3 Mission Planning, Routing, and Navigation

Mission planning generally operates at the level of human communication [39]. Mission planning makes decisions at the highest level for example getting off at the next ramp on a freeway or making a left turn at the next intersection. This means it is most suitable for direct interaction with the passengers. Depending on system architecture mission planning can be further divided into finer decisions at the lane level. Routing and navigation for example makes more local decisions such as "merge into the left lane in the next 300 meters".



(a) (b)

Figure 2.7: Comparison between global path (a) and semi global path (b) on a Colby Drive lanelet map. Path is depicted as a green line.

2.1.4 Behavior Planning

Behavior planning receives information from the subsystems above and makes decisions to execute the mission generated by the mission planner [39]. In the context of the ego vehicle's local decision making, behavior planner outputs relatively higher level maneuvers that are further interpreted to produce vehicle control. A loose analogy to manual driving are decisions that a person would make when cruising down a freeway. Decisions such as slow down to maintain distance from a leading vehicle or overtake leading vehicle because it is too slow. The number of available maneuvers and their function are a design specification but may include for example: stop, continue in current lane, left lane change, right lane change, and follow leading vehicle. Behavior planning is one of the main subsystems related to handling pedestrians at crosswalks.

Maneuvers instruct the local planner through a number of different ways. Behavior planning consumes the abstract environment around ego represented by the map and dynamic objects provided by the tracking system. These features are summarized and converted into a format containing a set of predicates further abstracting the environment. Predicates are then processed through a set of rules in a system known as the rule engine and returns a maneuver depending on the rules that are satisfied by the predicates it receives. Decisions provided by the rule engine are further packaged with auxiliary information to finally produce one of many maneuvers available by the behavior planner.

Complete documentation of the rule engine for behavior planning is available at [7].

Maneuvers

1. *Track Speed*: This maneuver is the most common as it is what instructs the motion planner to drive toward the goal. This behavior provides a target speed to track, generally the speed limit of the road. Since this behavior is not concerned with other road users or the state of the road, it is least constrained and is of lowest priority.

2. *Decelerate to stop*: This maneuver is intended for purposes of non urgent stopping. An example of when it might be used is gradual stopping for a stop sign. This behavior is packaged with a stopping point where ego vehicle's speed should be 0 once it reaches the point.

3. *Stop*: This maneuver is intended to keep the vehicle stationary once it has stopped moving. The implicit stopping point packaged with this maneuver is the current location of the vehicle.

4. *Yield*: This maneuver is intended for scenarios where the ego vehicle does not have the right of way. An example of where this might occur is at stop sign intersections. This maneuver is packaged with a list of dynamic objects that ego should yield to. Packaging dynamic objects help during debugging to explain why the system decided to stop.

5. *Parked vehicle avoid*: This maneuver is intended to overtake vehicles that are parked by the side of the road blocking the ego vehicle's path. The message is associated with a list of vehicles that are marked as parked.

6. *Lead vehicle follow*: This maneuver is intended for following a vehicle that is directly in front of ego. It differs from track speed in that it must also consider the distance from the lead vehicle as well as the speed limit.

7. *Emergency stop*: This maneuver is designed to represent edge cases where the state of predicates are foreign in the rule engine's predicate domain. An example of a predicate state that may trigger this maneuver is when a pedestrian is not localized. Consequently, the rule engine does not know where the pedestrian is so decisions cannot be safely made.

Rules

The rule engine is a system based around a collection of rules that evaluate its input predicates independently and collectively vote for a final output maneuver. Its architecture is based off the lambda architecture [36]. Approximately 150 rules map abstract states to maneuvers together forming expectations, constraints, and overall decision making. Below are 2 examples that demonstrate how rules operate on predicates provided by behavior planning.

Figure 2.8 is an example of one of the rules designed for managing 4-way stop sign intersections. The rule is defined by an identifier, a target maneuver, a description, and the qualifying state of predicates that triggers the rule. One of the predicates included indicates approaching an intersection. The approaching predicate is further discussed with respect to crosswalks in section 3.2.1 and 3.2.1. Another predicate under the travel group indicates the affecting regulation is a stop sign. The rule translates to: under the condition that ego is approaching an intersection and is affected by a stop sign, the target maneuver voted for by this rule is decelerate to halt. If the decelerate to halt maneuver is selected resulting from this rule's vote, the maneuver is packaged with additional details. The decelerate to halt maneuver is packaged with a stopping point as mentioned in section 2.1.4.

```
System Id : RIS3
Decision  : decelerate-to-halt
Constraint: Stop Line
Goal: Slow down to stop at the stop line of the
intersection.

ALL {
  ego {
    location: {
      approaching: 'intersection'
    }
  }

  travel: {
    regulation: 'stop'
  }
}
```

Figure 2.8: Simplest rule handling stop sign regulation at an intersection

Figure 2.9 is one of many rules regarding model integrity. This rule excludes the possibility of more than 1 dynamic object (vehicle/pedestrian) marked as the leading object. It is certainly possible in the real world to see 2 vehicles driving side by side in the same lane if it is wide enough however this rule explicitly limits the system and does not handle that scenario. Anything that exceeds the expectations of the rule engine may be a source of a bug or unexpected behavior.

```
System Id : RMI1
Decision  : emergency-stop
Constraint: None
Goal: Make sure there is never more than one
leader.

SOME-OF {
  many-vehicles {
    isLeading: true
  }

  many-pedestrians {
    isLeading: true
  }

  ALL {
    a-vehicle {
      isLeading: true
    }

    a-pedestrian {
      isLeading: true
    }
  }

}
```

Figure 2.9: An example model integrity rule

With around 150 rules voting for output maneuvers, the rule engine decides on the final output by look-up in a precedence table where the maneuvers are ordered based on urgency. Maneuvers of higher urgency are chosen over ones of lower urgency regardless of the number of votes. A single vote for a more urgent maneuver such as emergency stop is selected over many votes for a less urgent one like track speed. If RMI1 in figure 2.9 and RIS3 in figure 2.8 are both triggered, emergency stop is ultimately the output maneuver because emergency stop has higher precedence over decelerate to halt. The ordering of maneuvers is listed below from highest precedence first to least:

1. Emergency stop
2. Stop
3. Yield
4. Decelerate to halt
5. Overtake vehicle
6. Follow leader
7. Track speed

2.1.5 Motion / Local Planning

Motion planning refers to the lowest level software control in the conventional autonomous stack [39]. We do not consider the vehicle's internal control systems interconnected by the LIN or CAN bus to be apart of the autonomous stack since they preexist the autonomous vehicle era and is present regardless of whether or not the vehicle is autonomous.

Local planning aims to execute maneuvers dictated by behavior planning. It converts high level command to low level command by controlling steering and acceleration. This is usually where non-holonomic constraints and kinematic models are used to generate feasible paths that can be physically executed. Passenger comfort can be incorporated at this level as constraints and requirements such as: maximum lateral acceleration, maximum or nominal longitudinal acceleration, collision avoidance, safety, and comfort to name a few.

Chapter 3

Mapping and Behavior Planning for Pedestrian Crosswalks

A crosswalk in the context of this thesis is defined as a cross sectional area of the drivable lanes that are clearly marked for pedestrians to cross from one side of the road to the other. In general, crosswalks may be supported by overhead lights which may be activated by pedestrians. These crosswalks are out of the scope of this discussion because the current system does not rely on perceiving flashing signal lights. The MTO identifies a crosswalk by pedestrian crossing signs, pavement markings, and lights. Figure 3.1 show the 4 types of pedestrian crossovers present in Ontario. Only the upper right crosswalk configuration is considered as the other three have some form of light signal.

The majority of interaction the autonomous vehicle faces at crosswalks involve pedestrians. There is always the possibility in any case of encountering other road users such as following other vehicles, cyclists, and pedestrians on scooters or skateboards. Although the current state of the rule engine is capable of vehicle following, interaction with pedestrians is the main topic with vehicle following as a minor extension. Additional variation of crosswalks sometimes do not span across the entire width of the road. The crosswalks considered in the following discussion refer to only those that span from one side of the road to the other. The specificity of type of crosswalk that is handled is because it is the only type of crosswalk that is present in the considered Operational Design Domain and on Ring Road at the University of Waterloo. Absence of any light signalling also provides pure pedestrian-driver interaction.

The expectation is that pedestrians within range of a crosswalk can choose to either cross or not cross. Pedestrians that choose to cross the road are expected to use the nearest

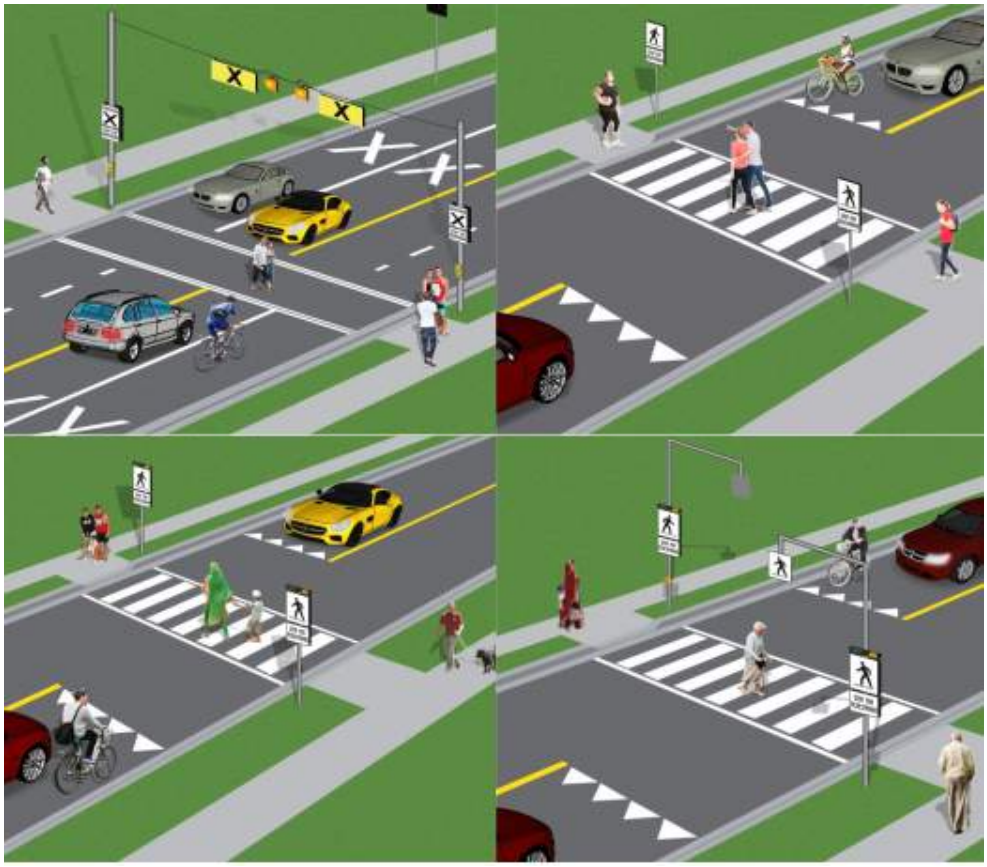


Figure 3.1: 4 Types of pedestrian crossovers in Ontario [38]

crosswalk instead of jaywalking. Jaywalking is loosely defined as crossing the road farther than approximately 1 meters from the edge of the marked boundary of a crosswalk. Some uncommon but not impossible scenarios include sprinting across the crosswalk, pedestrian stopping on the road while crossing, and pedestrian walking back and forth on the road along the crosswalk. Scenarios are further discussed in section 4.

Jaywalking is similar to using a crosswalk in that pedestrians leave one side of the road to get to the other however they are different with respect to the right of way. Pedestrians have the right of way where pedestrians can be expected with higher likelihood of crossing at crosswalks. With jaywalking on the other hand, the autonomous vehicle has the right of way and follows a different driving policy because pedestrians are expected to wait at the edge of the road. This chapter aims at handling common crosswalk scenarios while considering the possibility of unlikely scenarios. Explicit jaywalking is not included in the

scope of this thesis.

3.1 Map representation of crosswalks

Lanelets2 [40] briefly describe pedestrian crosswalks represented as lanelets elements with a left and right way/linestring without explicit example. This section introduces the crosswalk designed as a regulatory element with supporting detail.

Crosswalks are represented in the lanelet map by a minimum number of elements, including only those that are present on the road. Crosswalks contain 2 boundary edges that outline the area of the road where pedestrians can cross as well as stop lines where the vehicles should stop for pedestrians. Figure 3.2 and 3.3 shows the 4 ways that together represent a crosswalk. In the lanelet map, crosswalks are regulatory elements that are referenced by the lanelets they affect. On Ring Road, a standalone crosswalk is referenced by two lanelets, one in either direction because Ring Road is a 2 lane road. Since the crosswalk regulatory element contains two stop lines, behavior planning is responsible for identifying the relevant stop line affecting ego. To account for pedestrians that walk near the crosswalk but outside of its ground marking, crosswalk bounds are relaxed by approximately 1 meter on both sides. Figure 3.2 shows 4 different lanelets that are separated by the stop lines creating a staggered result.

Alternatively, it would be simpler for behavior planning to represent a crosswalk on Ring Road with 2 separate regulatory elements, each one affecting their own respective lanelet. Using this representation, there would be a 1 to 1 relation between stop line and regulatory element. This eliminates the burden on behavior planning to identify which stop line affects ego's current lanelet. The trade off between detail in the map and additional processing in the behavior planner is between map complexity, and behavior planning complexity. A single regulatory element per crosswalk is chosen because it maintains consistency between the map and real world.

3.2 Behavior planning for Crosswalks

The main goal for navigating through crosswalks is to identify when an area is free for ego to drive through. There are many possible strategies that accomplish the task, for instance, waiting until the crosswalk is clear first before proceeding. The selected approach is based on windows of time; crossing only when ego's window of time does not overlap with any

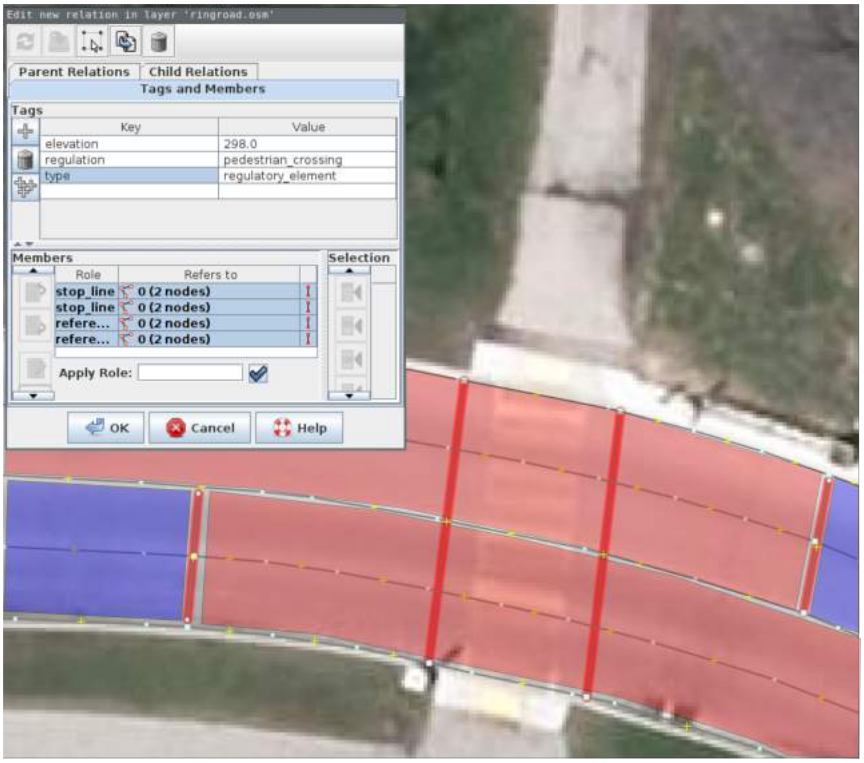


Figure 3.2: A crosswalk highlighted in JOSM including 2 stop line members and 2 reference members

pedestrians' window of time. To carry out this approach, 2 main predicates are used to abstract the environment for the rule engine to consume. The predicates are converted into windows of time and checked for conflict.

3.2.1 Environment Abstraction Predicates

Predicate design balances complexity between predicate generation from the sensed environment and predicate evaluation in the rule engine. For the purpose of pedestrian crosswalks, two main predicates are identified to reduce scenarios into an easily manageable scope.

The predicates rely on a number of special reference points. Figure 3.4 is a crosswalk annotated with these reference points. Crosswalk entrance points (yellow) are the center points between the 2 bounds of the crosswalk on either side of the road. The crosswalk

```
<relation id='-1110798'>
    <member type='way' ref='-1110642' role='reference' />
    <member type='way' ref='-1110643' role='reference' />
    <member type='way' ref='-1110645' role='stop_line' />
    <member type='way' ref='-1110644' role='stop_line' />
    <tag k='elevation' v='298.0' />
    <tag k='regulation' v='pedestrian_crossing' />
    <tag k='type' v='regulatory_element' />
</relation>
```

Figure 3.3: Lanelet crosswalk in XML representation

center point (green) is calculated from the average of the 2 crosswalk entrance points. The red points along the top and bottom edge of the crosswalk are used to calculate the exit time of crossing pedestrians. Their usage and further detail is discussed in the next section.

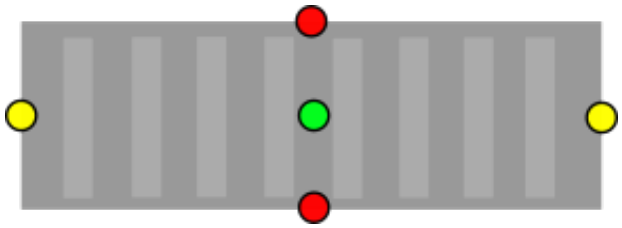


Figure 3.4: Crosswalk with reference points marked

Predicate - Time for pedestrians in the crosswalk to exit

One of the main predicates is the estimated amount of time for currently crossing pedestrians to exit the crosswalk area. In this context, the crosswalk area may refer to the entire crosswalk or only part of it that is closer to ego. Depending on parameters discussed in later sections, the crosswalk area refers to an area that is deemed relevant. The first challenge is identifying pedestrians on the crosswalk. An attempt at this based on the principle of dot products uses 2 unit vectors. One from each crosswalk entrance point is drawn toward the pedestrian. The dot product between these 2 vectors is negative when the pedestrian is on the roadway and positive when it is off the roadway. This is illustrated in figure 3.5.

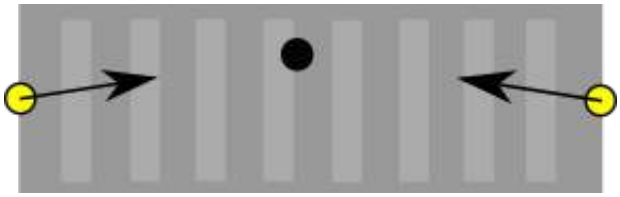


Figure 3.5: Crosswalk with unit vectors (black arrows) pointing from entrance points (yellow) toward pedestrian (black circle)

In theory, the dot product polarity successfully identifies the position of a pedestrian; however, in practice road geometry is not exactly rectangular. In reality, crosswalks are parallelograms and sometimes distorted closer toward trapezoids. This approach fails to identify pedestrians on the crosswalk shown in figure 3.6.

Figure 3.6: Exaggerated crosswalk where dot products of unit vectors are positive

Another approach for determining whether a pedestrian is on the crosswalk is with methods of computational geometry, determining whether the pedestrian lies within the area of the polygon enclosed by the crosswalk boundary ways. The even-odd rule states that a point P is within the polygon if a straight line projected in any direction from P crosses the polygon boundary an odd number of times under the condition that the line does not pass through any vertex. A pedestrian is not on the crosswalk if a line projected from P crosses the polygon boundary an even number of times. Another strategy is called "non-zero winding"; where a straight line is projected again from the pedestrian point P. In the winding strategy, the direction of each edge of the polygon is used. For each clockwise intersection of the polygon edge with the straight line projection, add 1. Subtract 1 for each counter-clockwise intersection. If the value after all intersections with the curve is 0, the point is in the polygon. Figure 3.7 illustrates both strategies. In the implemented source code, point in a polygon is computed using the boost geometry library [3] with the default "non-zero winding" strategy. A pedestrian on the edge of the crosswalk polygon is considered to be on the crosswalk.

The crosswalk is split longitudinally along the road into 2 sides: ego's side of the road

and other side of the road. Ego's side of the road is the side of crosswalk where ego would enter as it drives through the crosswalk. The boundary between the 2 sides is determined by 2 points, 1 along each crosswalk bounding way. Figure 3.8 shows a crosswalk boundary with reference points A and B. A point along this boundary is selected based on parameter theta with the affine combination in equation 3.1. The same parameter theta selects the point along the other boundary. The side of A is selected to be further away from ego. In the figure, ego would be traveling upward along the right side. These 2 resulting points dictate the position of the boundary line between ego's side of the crosswalk and the other side.

$$P = \theta A + (1 - \theta)B \tag{3.1}$$

With regards to crossing pedestrians, there are 4 categories illustrated in Figure 3.9: 2 directions and 2 sides of the crosswalk allowing for 4 total combinations. The figure illustrates the ego vehicle on the right side of the road and the crosswalk divided between ego's side of the road and the other side. Pedestrian 1 and 3 are on the side away from ego separated by the line drawn across the points acquired from above. Pedestrian 3 can be safely ignored since it is not on ego's side of the road and walking away from ego. Note that MTO suggests that vehicles should yield for pedestrians throughout the entire span of the crosswalk from curb to curb. Adjusting the parameter theta in equation 3.1 allows the ego vehicle to drive with a range of behavior. Pedestrian 4 is considered to be off the crosswalk by the time it reaches the dividing line between ego's side and other side. Pedestrians 1 and 2 are considered to be off the crosswalk when they reach the right edge of the crosswalk. Pedestrians are considered stopped when their speed is below the tracker noise threshold of 0.3 meters per second. In this case, a stopped pedestrian on ego's side of the road produces an infinite time to exit while a stopped pedestrian on the other side produces a time to exit of 0. Consequently, a pedestrian stopped on ego's side of the road causes ego to wait indefinitely and a stopped pedestrian on the other side of the road is ignored. Since the algorithm operates at 10 Hz, as soon as the pedestrian begins moving, it will be allocated a reasonable time to exit crosswalk and no longer ignored. When a pedestrian enters the crosswalk area, a time to exit crosswalk is computed.

Time is computed as the distance over velocity outlined in equation 3.2 where distance d is the shortest distance between a point to a line. The velocity v used in computing time is the component of pedestrian velocity along the vector from one crosswalk entrance to the other. This works ideally for perfectly rectangular crosswalks and works well as an approximation for crosswalks resembling more of a trapezoid.

$$t = \frac{d}{v} \tag{3.2}$$

When there are multiple pedestrians crossing simultaneously, the time to finish crossing is computed for each and the max is taken. By considering only the maximum time to finish crossing, all pedestrians that are crossing are abstracted to a single variable. When there are no pedestrians currently crossing the crosswalk, a default value of 0 seconds is set to represent the empty crosswalk.

Predicate - Time for pedestrians to enter crosswalk

The other main predicate is the estimated time for a pedestrian to enter the crosswalk. This predicate allows the rule engine to determine whether there is enough time for ego to drive through the crosswalk before any pedestrian enters.

The first step is to identify pedestrians around a crosswalk that may potentially cross. A semicircle is drawn around each crosswalk entrance for determining whether to ignore detected pedestrians or not. Figure 3.10 illustrates a crosswalk with a semicircle shown on the left side.

Pedestrians that are not in the area bounded by the semicircles are excluded from further consideration. Since the semicircle is not explicitly marked in the map, it is extrapolated from the yellow entrance reference points in figure 3.4. This is effectively achieved by the intersection of 2 sets. The polarity of the dot product between a unit vector from one crosswalk entrance to the other and another unit vector from the crosswalk entrance to the pedestrian position selects the half space off of the road. If the dot product is positive, then the pedestrian is unlikely to be on the roadway. If also the distance of the pedestrian from the closest entrance center point is within a certain value, then the pedestrian is within a circle around the closest crosswalk entrance. Consequently, the pedestrian is within the area of the semicircle. The same operation is done on both sides by swapping the roles of the control points.

Pedestrians that fall within the semicircles are further filtered based on direction of travel. Only movement is examined for this filter because it is the only reliable input available. Other studies investigating whether a pedestrian will enter the road use learned models to predict future trajectory [19, 43, 24]. Pedestrians moving away from the crosswalk suggest they will not enter. The classification of a pedestrian moving toward or away from the crosswalk is determined by its velocity component in the direction toward the center of the crosswalk. Equation 3.3 is used to compute the component shown in Figure

3.11. Vector a points from the pedestrian position to the center of the crosswalk and is shown as the solid arrow in Figure 3.11. Vector b is the pedestrian velocity vector shown as the dotted arrow. When a pedestrian's velocity is on the scale of measurement noise, i.e. less than 0.3 meters per second, it is not excluded. This is useful for identifying pedestrians waiting to cross.

$$comp_{\vec{a}}\vec{b} = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}|} \tag{3.3}$$

The list of qualifying pedestrians is then analyzed for approximating the time to enter crosswalk predicate. The operation is instantaneous extrapolation of velocity at the operating frequency of behavior planning which is around 10 Hz. Similarly for approximating time for pedestrians to finish crossing, the entering time estimate is computed by equation 3.2.

Distance to the crosswalk is expressed by the shortest distance to a circle centered at the crosswalk center with a configurable radius. Figure 3.12 illustrates an example of a circle centered at the crosswalk. The radius is another configurable parameter that facilitates the selection of driving assertiveness.

This formulation of distance accounts for pedestrian paths that slightly cut the corner when entering the crosswalk since the size of the crosswalk is effectively expanded to cover those corners. This formulation along with the slightly expanded crosswalk boundaries captures a smooth transition from off-road to crosswalk without gaps in between. Notice in Figure 3.12 there is a section of the circle that overlaps with the sidewalk. This area is important for the other auxiliary predicates discussed below.

Estimation of time to enter crosswalk is done for each filtered pedestrian and the minimum time is selected. Since only one time is truly relevant, the simplification to one variable alleviates the rule engine from having to accommodate for an arbitrary number of pedestrians. When there are no pedestrians around the crosswalk entrances, the estimated time to enter defaults to a large number.

Auxiliary Predicates - Pedestrians

Dynamic objects around crosswalks are localized with respect to the crosswalk to provide the rule engine with additional information for making decisions. The three main predicates for localizing dynamic objects are *approaching*, *at*, and *on*.

As the name suggests, the approaching predicate identifies an area where a pedestrian is considered as approaching. All pedestrians in the semicircle areas around a crosswalk's entrance described in section 3.2.1 are considered as approaching the crosswalk. This does not translate to an overly conservative driving policy because of the other predicates in place.

The *at* and *on* predicates originate from the use case of intersections. A vehicle deemed to be at an intersection is one that is at the boundary of the intersection area whereas a vehicle that is on the intersection is within the intersection area. The definition of the intersection area is not relevant to crosswalks.

All pedestrians that are within the semicircle areas of crosswalks are assigned the approaching crosswalk predicate with an exception. Pedestrians that are within the semicircle area and within the circle centered at the crosswalk center are assigned as *at* crosswalk. Particular interest is paid to pedestrians that are at the crosswalk. Firstly, when a pedestrian is close enough to the entrance of a crosswalk, it is assumed the pedestrian intends to cross and is waiting. When a pedestrian is at the crosswalk and not walking away from the crosswalk, the time to enter for that pedestrian is by default 0 seconds. This forces the autonomous vehicle to stop for the waiting pedestrian. Again, this does not directly result in an overly conservative driving policy when considering pedestrians that are leaving the crosswalk because of how pedestrians are filtered. Advanced methods relying on alternative means of perception incorporate features like body language [29] and may produce more realistic behavior. Pedestrians that have recently left the crosswalk will also step into the at crosswalk zone but they are removed from consideration so they are not marked with *at* crosswalk.

Pedestrians are labelled as on crosswalk when they satisfy the same conditions discussed in section 3.2.1. Figure 3.13 shows the 3 regions where pedestrians are marked as *approaching*, *at*, and *on* crosswalk.

Auxiliary Predicates - Vehicles

Approaching, at, and on are used to localize vehicles around crosswalks as well. Vehicles are assigned approaching a crosswalk when the lanelet they are currently in has a crosswalk regulatory element affecting it and the vehicle is within a certain distance from the crosswalk stopping line. Study has shown that drivers respond to jaywalkers at around 26 meters away [50] therefore, vehicles are set to be approaching a crosswalk when they are within 30 meters before the stopping line.

A vehicle is at a crosswalk when it is within 5 meters before the crosswalk stopping line. Because the crosswalk regulatory element has multiple stopping lines, the relevant line is found by closest distance from ego. The uncertainty is a consequence of representing a crosswalk by only 1 regulatory element. A vehicle is set to be *on* the crosswalk when it is within 10 meters after the stop line of that crosswalk but can be set to be dynamic depending on crosswalk width. Since lanelets end at crosswalk stop lines, this is equivalent to 10 meters within the start of the next lanelet. Figure 3.14 illustrates the areas of interest for vehicle crosswalk localization on a 2 lane road for both directions.

3.3 Rules for Crosswalks

The rule engine combines desirable action in mini scenarios together to form the final driving policy. Rules are translations of common sense reasoning in specifically defined environment states. For instance, it is common sense to drive through a crosswalk when it is free, and to stop when it is not. This behavior is embedded in rules with the use of the two main predicates from section 3.2.1 and 3.2.1.

Before the main predicates are sent to the rule engine, the times are converted into a time spans that represent when the crosswalk is busy. In the case of no pedestrians currently crossing, the time range is [time to enter, inf]. Infinity is used as the upper bound because it is uncertain when the pedestrian will finish crossing. When the pedestrian enters the crosswalk, the time to finish crossing predicate takes over. This leaves an opening for ego to cross before the pedestrian reaches the crosswalk if the time to enter is large enough. If there are no pedestrians entering the crosswalk, the range is reduced to [inf, inf] effectively leaving the crosswalk completely open to cross. When a pedestrian is currently crossing and is expected to leave the crosswalk before any new pedestrians enter, the time ranges are [0, time to cross] and [time to enter, INF]. 0 is the lower bound representing the current time and the time to cross blocks the crosswalk. If a pedestrian A is currently crossing and a new pedestrian B is expected to enter before A finishes, the time range is set to [0, INF]. The pseudo code is displayed in algorithm 1. The windows of time are implemented as a list of lower bounds and a list of upper bounds. A lower bound state of [0, 10] and upper bound state of [5, INF] means the crosswalk is busy from 0 to 5 seconds, free from 5 to 10 seconds, and busy again from 10 seconds to infinity. Figure 3.15 illustrates a potential state with 2 busy windows of time.

The implementation with multiple time ranges make extension for crosswalks at intersections feasible. Additional blocking time ranges can be appended for each crosswalk

Algorithm 1 Convert to range

```
1: procedure CONVERT TO RANGE(timetoenter, timetoexit)
2:     lowerbounds ← []
3:     upperbounds ← []
4:     if timetoexit == 0 then                          ▷ No pedestrians crossing
5:         lowerbounds.pushback(timetoenter)
6:         upperbounds.pushback(INF)
7:     else
8:         if timetoexit < timetoenter then  ▷ Ped exits crosswalk before another enters
9:             lowerbounds.pushback(0)
10:            lowerbounds.pushback(timetoenter)
11:            upperbounds.pushback(timetoexit)
12:            upperbounds.pushback(INF)
13:        else                          ▷ Ped enters crosswalk before currently crossing exits
14:            lowerbounds.pushback(0)
15:            upperbounds.pushback(INF)
16:        end if
17:    end if
18: end procedure
```

affecting ego's path and the list of blocking windows of time would determine when ego can cross.

3.3.1 Predicate State and Maneuvers

All crosswalk rules are available in appendix B. These rules map predicate states to maneuvers in the rule engine. There are seven rules specifically designed for handling crosswalk predicates.

RCW1 selects the *track speed* maneuver with reduced speed when ego is *approaching, at, or on* a crosswalk.

RCW2 selects the *decelerate to halt* maneuver when ego is approaching a crosswalk and the time window conflicts with pedestrians.

RCW3 selects the *yield* maneuver when ego is sufficiently slow enough at the crosswalk stop line and time windows conflict.

In a similar state to RCW3, RCW4 selects the *emergency-stop* maneuver when ego is at the crosswalk but traveling at a speed greater than a predefined limit.

RCW5 selects the *emergency-stop* maneuver when there is a time window conflict and ego is already in the crosswalk area. Since the crosswalk area begins just after the stop line, ego may not be in the road marking but very close to it.

RCW6 selects the *follow-leader* maneuver when following a vehicle around a crosswalk. The main purpose of this explicit rule is to combine vehicle following with a reduced speed limit.

RCW7 selects the *decelerate-to-halt* when following a vehicle and the time windows conflict with pedestrians.

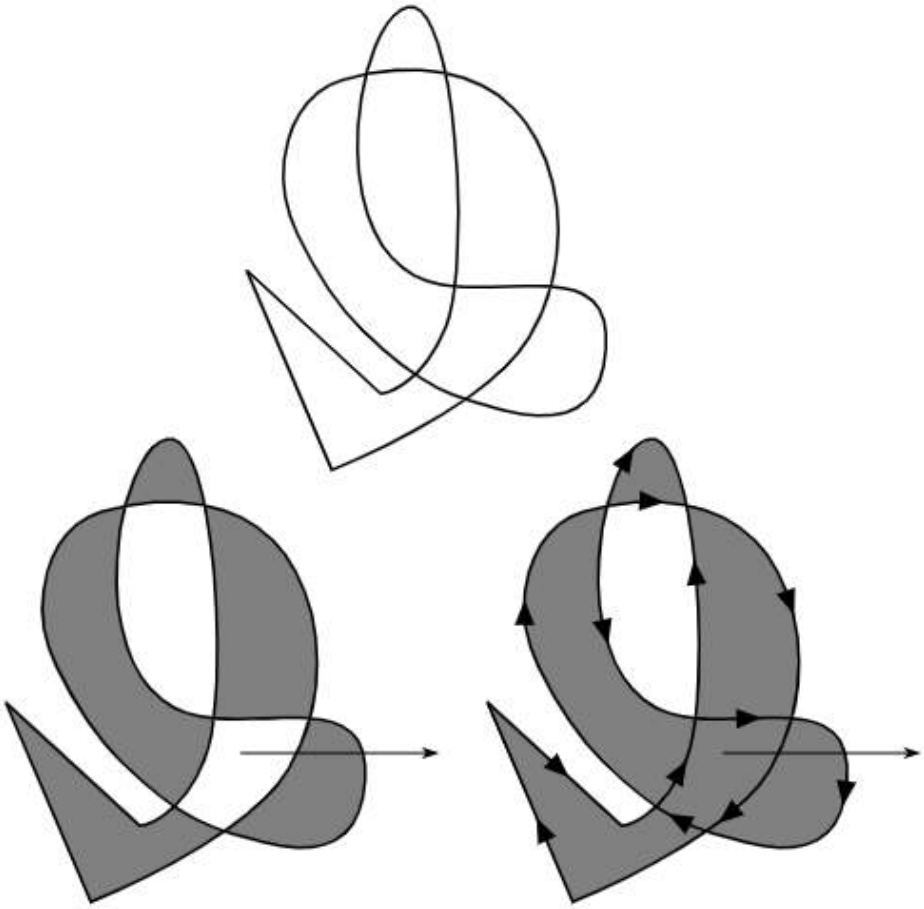


Figure 3.7: Even-odd strategy (left); Non-zero winding strategy (right) for determining point in polygon

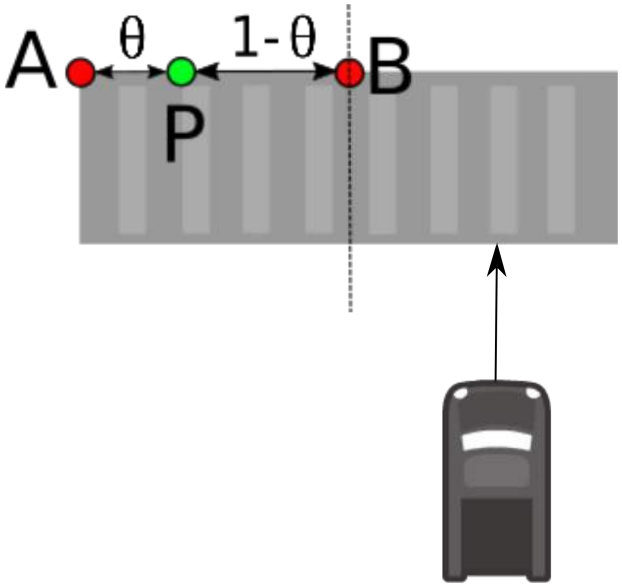


Figure 3.8: Labeled reference points in equation 3.1

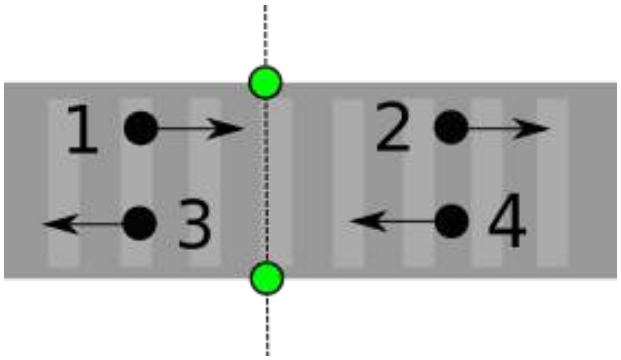


Figure 3.9: Four combinations of pedestrian position and walking direction

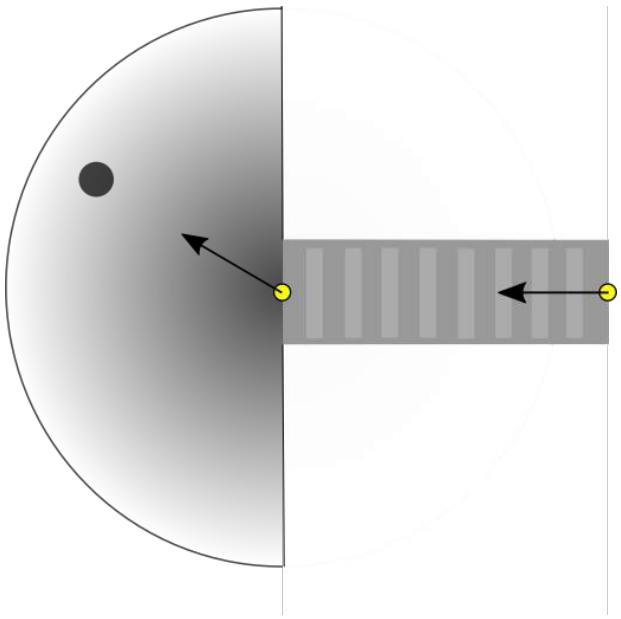


Figure 3.10: Crosswalk with a pedestrian in semicircle area shown on one side and relevant unit vectors

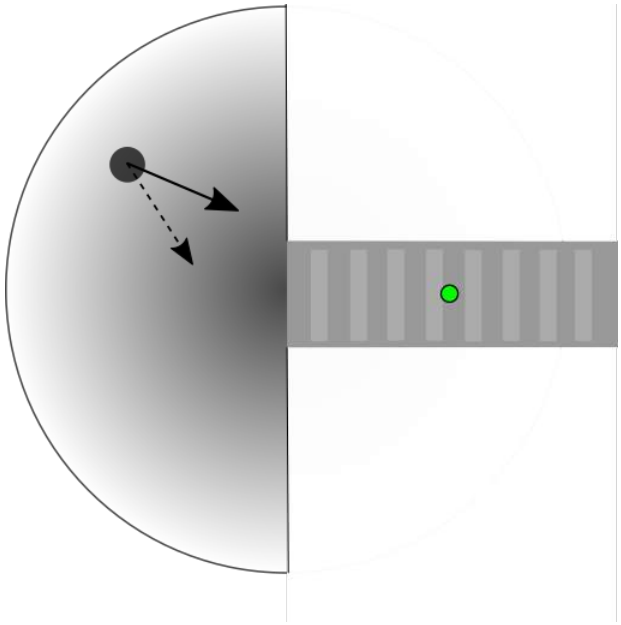


Figure 3.11: Vectors used for determining velocity component toward crosswalk. Dotted arrow is real pedestrian velocity; Solid arrow is auxiliary for computing component vector

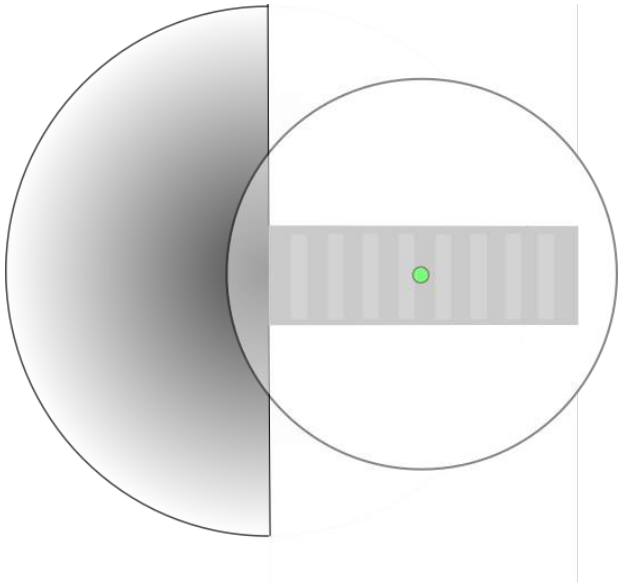


Figure 3.12: Configurable radius circle centered at the middle of the crosswalk

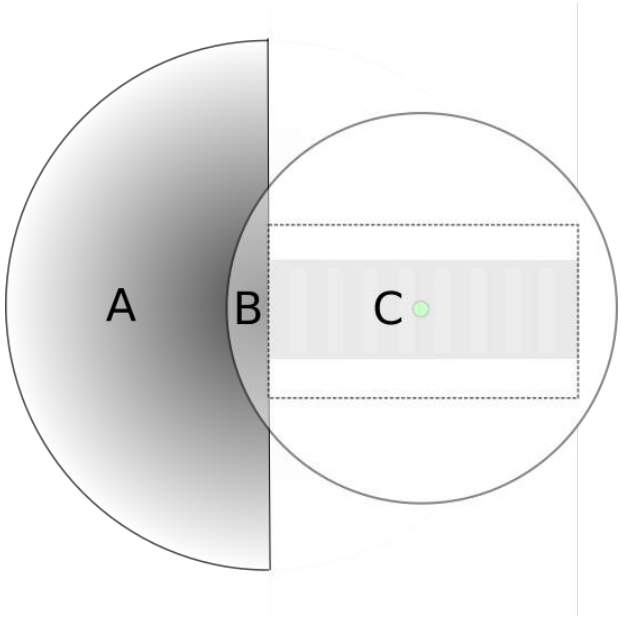


Figure 3.13: A: Approaching; B: At; C: On

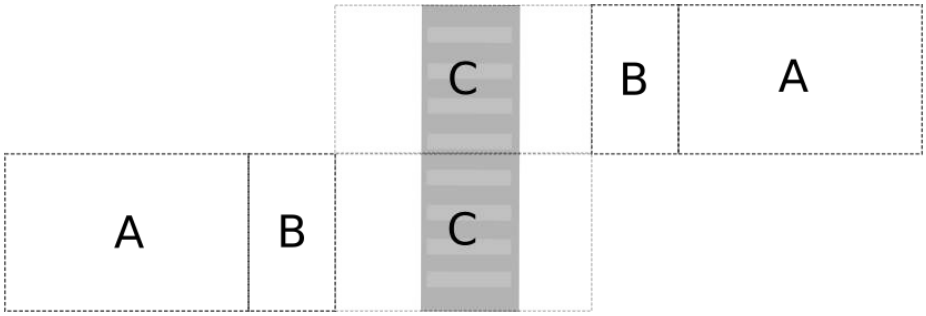


Figure 3.14: A: Approaching; B: At; C: On for vehicles around crosswalks

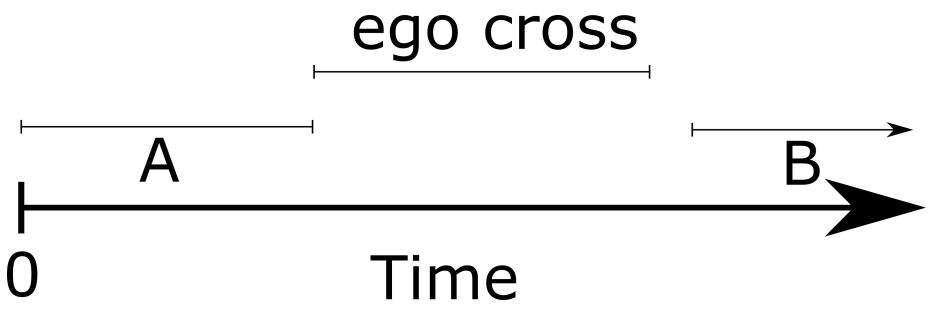


Figure 3.15: Time range when crosswalk is occupied; A: Currently crossing pedestrian; B: Other pedestrian enters crosswalk; C: Time for ego to reach crosswalk and cross

Chapter 4

Observations

4.1 Simulation

Simulation is an important method in testing behavior planning performance. It allows for quick testing and debugging turnaround, leaving out hardware related issues. Faster turn around and iterations in the development cycle allows for quicker bug identification and fixes.

The simulator used to debug and demonstrate driving behavior is based on Unreal Engine developed by Epic Games [17]. Originally developed for first person shooter games, the engine contains a vast variety of features which have been successfully used for other genres. A ROS wrapper is built around the base unreal engine system so that ROS messages can interface with the game engine. The simulator emulates realistic vehicle control dynamics modelled directly using data collected from the real car [48, 25]. The simulator provides only what is necessary as inputs to the autonomous stack. A satellite image is layered on the ground plane to provide general reference of where the ego vehicle is located. The lanelet map is then overlaid on top of the satellite image to visualize how the autonomous vehicle behaves relative to the road map. What is important is how ego drives with respect to the lanelet map rather than with respect to the satellite image. There is a visible offset between the satellite image and lanelet map but since the satellite image is not perceived by the stack, the offset is irrelevant.

In simulation, pedestrians and vehicles are dynamic obstacles that move discretely over time steps. Although it isn't identical to real world continuous movement, it is suitable because perception in the real world ultimately discretizes tracked object motion. For the

sake of analyzing behavior planning performance, stack perception is not used. Pedestrian and vehicle states are input directly from the simulator to the tracker module. This eliminates errors introduced by false detections and missed detections. Scenarios are described in a format known as GeoScenarios [41]. Dynamic objects follow preplanned trajectories where every point in the trajectory can be assigned individual speed and acceleration characteristics. Additionally, triggers facilitate synchronization of interaction between agents and ego. Triggers are simply events that orchestrate other events. The common use case of a trigger is to specify that when ego reaches a particular destination, another agent will begin to move along its specified trajectory.

Pedestrian walking speeds in the hand crafted scenarios described below are selected from studies that observed unobstructed pedestrian motion as well as pedestrians behavior around crosswalks. [23, 49]. Figure 4.1 illustrate the distribution of walking speeds of unobstructed pedestrians [49]. The assumption made here is that the reasonable person will belong somewhere in the distribution so most pedestrian speeds are selected from the range. Some imagined extreme scenarios are also tested.

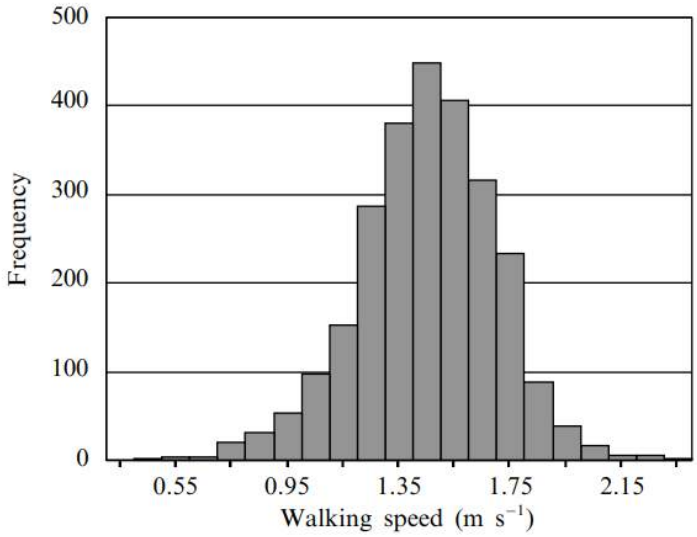


Figure 4.1: Distribution of walking speeds. Frequency distribution of individual pedestrians' walking speeds averaged across the survey area (n = 2613). The data are distributed normally about the mean ($1.47\frac{m}{s}$), with a standard deviation of ($0.299\frac{m}{s}$) [49]

4.1.1 Defining Scenarios

GeoScenario is a framework for defining scenarios based on latitude-longitude coordinates. Individual test cases allow control over agents and their interactions at the level of velocity and acceleration. A basic GeoScenario contains an ego starting position and a number of ordered goal points. The ego vehicle spawns at the starting point and the test is successfully completed after each defined goal point is reached in order.

A GeoScenario is specified in osm format like the lanelet map and is easily configurable in JOSM. A pedestrian is defined as a single node with specific key value pairs that describes it. One of the key value pairs assigned to a pedestrian is the fixed path it follows. A path is defined as a set of nodes which can either represent a basic path or a trajectory. A basic path is traversed by the pedestrian with a constant speed specified as one of the pedestrian's key value pairs. A trajectory is defined by points that each have their own target speed and acceleration values. The agent attempts to match the specified speed and acceleration when it reaches a point along the trajectory. Realistic pedestrian movement can be simulated with enough resolution in distance between points and differences in target speeds. Simulated vehicles are similar to pedestrians except for their bounding boxes and rendered model is a vehicle instead of a person. Figure 4.2 shows an example of what a GeoScenario looks like in JOSM. Triggers are defined as owner-target pairs where the targets' behavior is dependant on the relation between the owner and the trigger point. The triggers used in the following scenarios are location triggers between ego and the trigger point. This means that target pedestrians and vehicles begin to move when ego reaches the trigger point. This mechanism gives control over how ego interacts with agents.

Simulation in Unreal Engine provides perfect sensing in terms of agents' positions; however, the tracker is still responsible for estimating higher order features like velocity and acceleration. Perception using AVOD [31] is available on the real vehicle but is not used in simulation. Instead, the simulator provides exact bounding boxes to the tracking system. Consequently location of all entities in the environment is known exactly with the exception of entities that are out of line of sight from ego. To emulate occluded agents, an object bounding box is available to ego if the center point of the object is unobstructed in the line of sight from ego's center point. When out of line of sight, the object no longer reports a bounding box to the tracker. This is where perception may differ by providing a bounding box even when the agent is mostly occluded. With perfect perception, the tracker is able to provide velocities with less than 0.3 meters per second error.

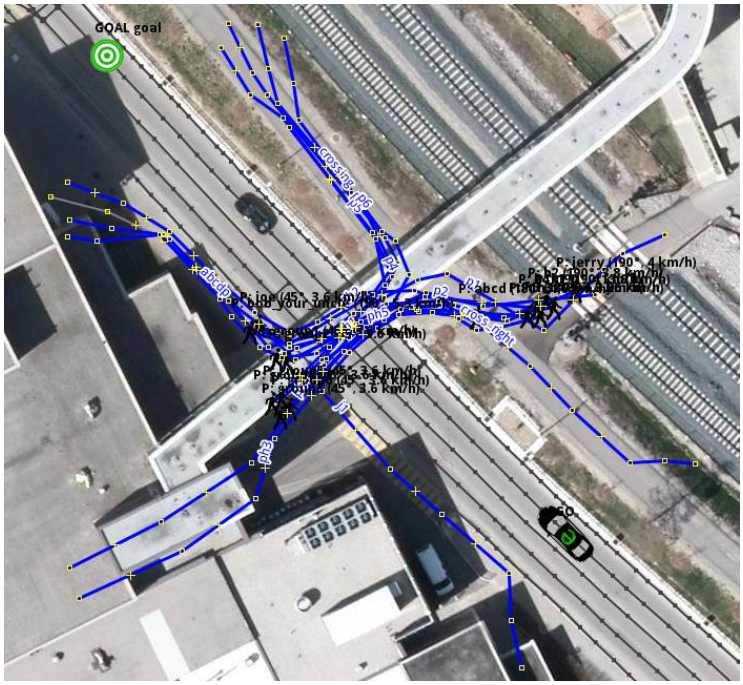


Figure 4.2: A sample GeoScenario of a large group crossing a crosswalk in front of ego

4.1.2 Hand Crafted Test Suite

The following is a number of imagined scenarios created to test the behavior of the autonomous vehicle following the behavior planning policy discussed in previous chapters. Each scenario described in this section tests a particular aspect of the pedestrian crosswalk. Since the set of all possible scenarios is limited by the imagination, there will be many that are not included. The tests below cover both common and some uncommon scenarios imagined in the context of Ring Road at the University of Waterloo. All scenarios are executed on a 2 lane road. The crosswalk dimension is 3 meters wide and 9 meters long from one side of the road to the other. As mentioned in section 3.1, the crosswalk boundary in the map is expanded to 5 meters wide to account for pedestrians that don't walk exactly within pavement marking. In the following scenario descriptions, the use of a reasonable person describes the expected behavior of a hypothetical driver subjected to these scenarios. The term is not well defined as with the criminal law equivalent and is based solely on my understanding of the law and experiences. Recordings of simulated results can be found at reference [10].

Minimum Gap

A fundamental metric that evaluates driving assertiveness in the context of crosswalks is the minimum gap between pedestrians that a driving policy can navigate between. In this scenario, the autonomous vehicle is initially stationary 6 meters from the edge of the crosswalk road marking. Pedestrians walk from the right to the left side with a constant gap in between them with speed varying from $0.55\frac{m}{s}$ up to $2.15\frac{m}{s}$ across multiple test cases selected from figure 4.1. The expected result is a linear relation between pedestrian walking speed and minimum gap distance. As pedestrians walk faster, a larger gap between them is required in order to drive through the crosswalk without collision. Figure 4.3 illustrates the observations made with pedestrian speeds with $0.25\frac{m}{s}$ increments. The effect of sliding the crosswalk division line with parameter theta as discussed in section 3.2.1 is seen to provide about a 5 meter difference between most assertive and most conservative behavior. What is most important here is the linear relation and effect of parameterizing behavior. The exact gap is dependent on how quickly the underlying controller accelerates the vehicle, as well as how far away the vehicle is stopped from the edge of the crosswalk.

Empty Crosswalk

This scenario is the simplest and provides a baseline reference of how the system behaves when it is unobstructed. The start position is on one side of a crosswalk and the end position is on the other side of the crosswalk. A reasonable person is expected to drive about the speed limit of the road and may potentially slow down slightly depending on visibility or obstruction to surrounding pedestrians. Since the scene is completely free of obstructions, a reasonable person is not expected to slow down. The autonomous system behaves as expected, with a lowering of maximum speed when it is within 30 meters of the crosswalk. It cruises along the road and passes without stopping.

100 Meter Dash

The 100 meter dash scenario simulates a pedestrian sprinting across the crosswalk at 44 kph. The autonomous vehicle travels at the speed limit of 30 kph toward the crosswalk. The sprinter is originally stationary and is triggered to cross the crosswalk when ego is 11 meters from the crosswalk stop line. 11 meters is chosen so that the pedestrian reaches the road when the vehicle is at the stop line. 44 kph was chosen because it is the top speed of Usain Bolt, and is unlikely to be topped by most pedestrians around ring road. The sprinter motion is not realistic in that its speed immediately jumps from stationary

Figure 4.3: Pedestrians walking speeds vs. minimum gap distance for ego to cross

to 44 kph once the trigger is activated. It is uncertain whether a reasonable driver would even notice the sprinter when they are already 11 meters from the stop line. If the driver does see a pedestrian sprinting in their peripheral vision, they may slam on the brake and come to a complete stop in 6 meters [45] at 30 kph. Alternatively, it is much more likely that the driver will not notice the sprinter and continue. The behavior planner responds to the pedestrian by switching behavior from track speed to decelerate to halt and local path planning steers slightly away from the pedestrian on the crosswalk.

Single Pedestrian Cross

The scenario involves a single pedestrian crossing the road from right to left and the vehicle begins stationary at the stop line. Although the MTO suggests that drivers wait for pedestrians to completely finish crossing before going, a reasonable person is expected to accelerate when the risk of collision is negligible. A reasonable person can expect the pedestrian to also be a reasonable person. While crossing the crosswalk it is unlikely for the

pedestrian to change their current velocity, so a driver may feel comfortable in accelerating after the pedestrian has passed the area in front of the car. It is commonplace to see vehicles on Ring Road accelerate from a stationary position before the crosswalk area is completely clear. The parameter in section 3.2.1 effectively controls the assertiveness for crossing when a pedestrian is walking toward the side away from ego. In the test, the parameter is set so that the boundary divides the crosswalk in half. MTO states that drivers must yield for pedestrians throughout the entire span of a pedestrian crossing. Like stopping at stop signs for 3 full seconds, it is in general too conservative and realistically obeyed by almost no one. On a road with dense pedestrian traffic, strictly following the rule results in little driving progress. The desired behavior of ego is to track speed when the pedestrian crosses the boundary set by the parameter. Qualitative analysis shows when set to the most assertive setting (halve the crosswalk), it is safe without collision and reasonable.

Pedestrians Do Not Cross

In this scenario, many pedestrians are walking on the sidewalk along the road and none of them cross. A reasonable person in this scenario is expected to be cautious of the many pedestrians that may potentially step on the crosswalk. Since the driver is approaching a designated pedestrian crossing, they may slow down in case any pedestrians decide to cross. Once the driver gets close enough to the crosswalk to safely pass without collision, they would return to the speed limit of the road. Behavior planning for this scenario is effectively the same as an empty crosswalk. The vehicle accelerates to the speed limit of 20 kph and decelerates when it is within 30 meters from the stop line. Since the calculated time to enter for all nearby pedestrians do not interfere with the time range for ego to cross, the maneuver does not switch from track speed. The radius of the circle in section 3.2.1 can be expanded for a more conservative driving style. When the radius of the circle extends past the edge of the road onto the sidewalk by 2 meters, some nearby pedestrians end up in the radius causing ego to decelerate to halt. After the pedestrians move out of the *at* zone, ego returns to tracking speed. Figure 4.4 shows the maneuver decelerate to stop is selected when the radius is extended to 2 meters.

Stationary Pedestrian

In this scenario, a pedestrian enters the crosswalk from the left and stops in front of ego, blocking the path. A variation of the scenario is also tested where a pedestrian walks in from the left side and stops before blocking ego's path. In both scenarios, a reasonable driver is expected to come to a stop at the stop line when the pedestrian enters the crosswalk.

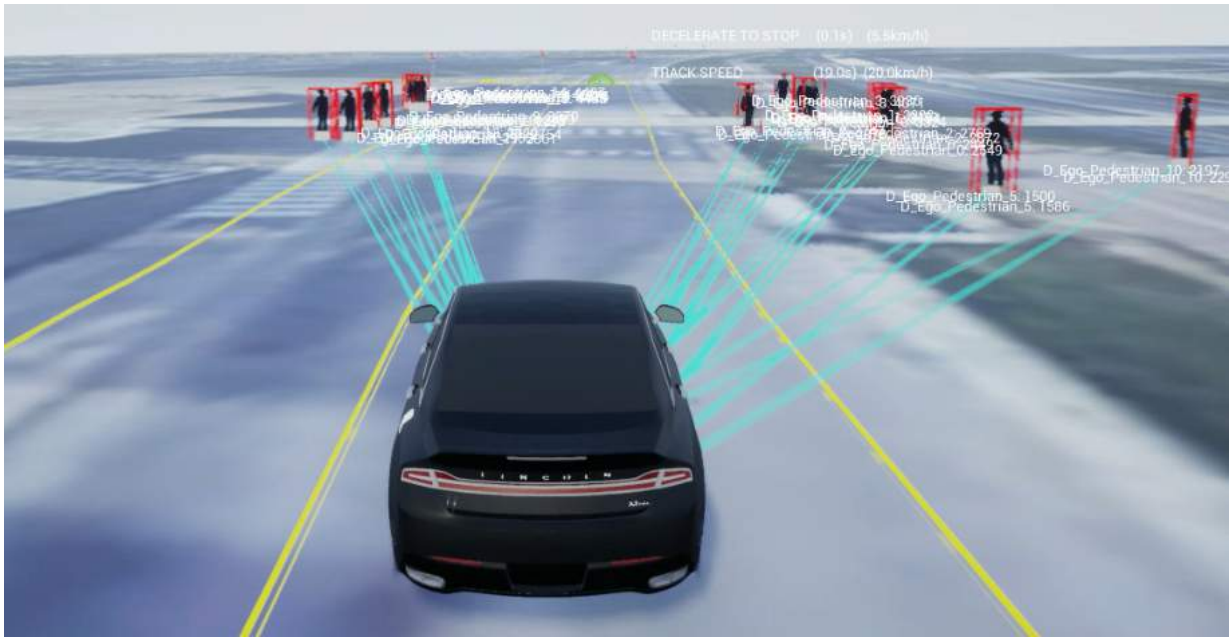


Figure 4.4: Unreal simulation of pedestrians walking along the side of the road (radius extended 2 meters)

When the pedestrian stops and blocks ego's path indefinitely, a reasonable person may run out of patience and drive on the wrong side of the road to pass the pedestrian. When the pedestrian stops and stands on the side of the crosswalk not directly in front of ego, a reasonable driver may continue along its path. The behavior observed in simulation for when the vehicle is blocked is indefinite waiting. Planning is not designed to maneuver onto oncoming traffic lanes so the system has no way of progressing. When the pedestrian stops on the side away from ego, the system recognizes the pedestrian speed has stopped and is safe to continue on its path.

Pedestrian Changes Mind

This scenario is described with a pedestrian entering the crosswalk from the left. When in front of ego, the pedestrian turns around and begins walking back toward the left side. Before exiting the crosswalk, the pedestrian turns around again and walks toward the right side. Assuming a reasonable driver, the driver would consequently assume a reasonable pedestrian. The driver may be confused by the large change in pedestrian velocity and may take a couple of seconds to understand the scene. After the pedestrian turns back

around and exits the crosswalk, the driver is expected to accelerate and continue on its path. Since this is an uncommon scenario, it is more difficult to imagine what a reasonable driver do. If abiding by the MTO standard, the driver should remain stationary until the pedestrian has completely exited the road.

For the first test, the crosswalk divider parameter is set to 0 (split the crosswalk in half). The system responds to the pedestrian by first decelerating to halt as it enters the crosswalk from the left. After crossing more than half the crosswalk, the pedestrian changes direction and starts walking back toward the left. Under the assumption that a pedestrian will likely continue to exit the crosswalk in the direction of which it moves, planning determines it is safe to cross and proceeds to accelerate. When the pedestrian turns back around, the crossing time window for ego is closed and decision returns to stop until the pedestrian exits the crosswalk. With the most assertive parameter setting, if the pedestrian repeatedly switches direction across the boundary between ego's side and other side of the crosswalk, the system will repeatedly inch forward until it is on the crosswalk at which point the pedestrian is ignored. When the crosswalk divider parameter is set to 1 (no split) the observed behavior is constant yield until the pedestrian finally exits the crosswalk.

Large Groups of Pedestrians

In this scenario, a large group of pedestrians enter the crosswalk from the left side while ego approaches the crosswalk. Before the group completely exits the crosswalk on the right side, another large group enters from the right side. This scenario aims to test the common event on Ring Road crosswalks where pedestrians cross in large and frequent groups.

A reasonable driver in this scenario is expected to decelerate to a stop when they see a lot of traffic on the crosswalk. Because the crosswalk is constantly busy, the driver is expected to remain stopped until an opening is clear. After the second group exits the crosswalk, the driver is expected to accelerate and continue on their way.

Since simulation provides perfect perception, the system acknowledges a group of pedestrians crossing well before reaching the crosswalk stop line and decides to decelerate to stop. The time to finish crossing is estimated for each pedestrian currently crossing and the maximum is used in calculating obstruction. When the other off roadway group approaches the crosswalk, the time to enter is approximated simultaneously for each pedestrian. The earliest time to enter is used for calculating obstruction. Since the time gap is not enough for ego to cross without collision, the system remains stopped until the last pedestrian of

the second large group has passed. Figure 4.5 is a snapshot showing the second group crossing toward the left and the first group finished crossing on the right.

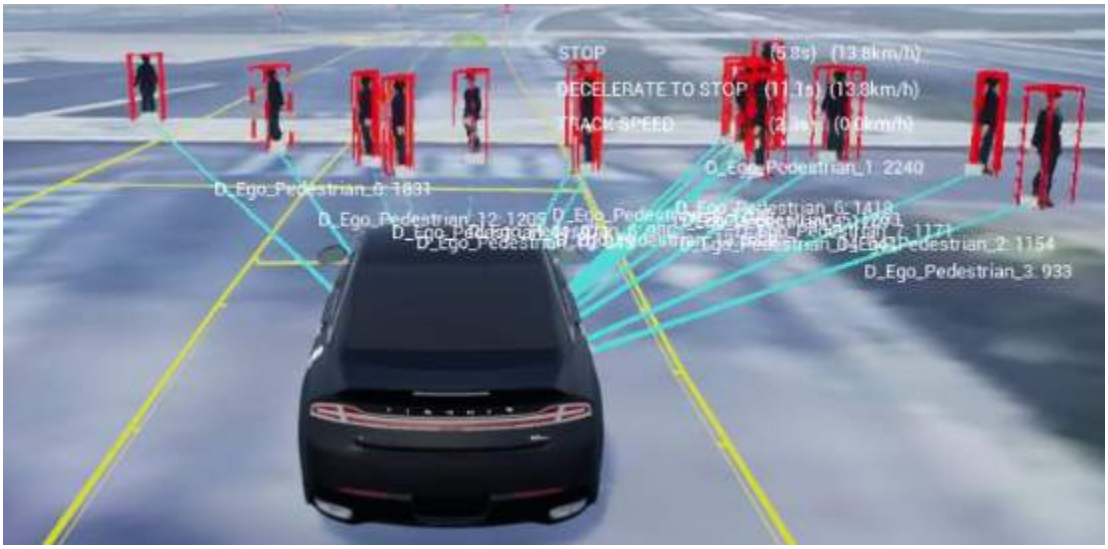


Figure 4.5: Unreal simulation of pedestrians two large groups crossing

4.2 Real World Data Test Suite

Testing the system on real world data is a method that allows direct comparison of behavior between real drivers and the autonomous system. Testing with recorded pedestrian trajectories is not equivalent to directly testing on hardware because the trajectories do not react to the autonomous system. The pedestrian will continue to travel along its trajectory even when there is an obstacle placed in front of it in simulation. An unsignalized pedestrian crosswalk on Ring Road at the University of Waterloo campus is the scene of the experiment. This crosswalk is one of the main pathways to the campus where the majority of pedestrians are university students. Figure 4.6 shows a side by side view of the drone video with trajectories played in simulation.

4.2.1 Data Capture and Annotation

A drone is hovered about 100 meters from the ground above the crosswalk and records in approximately 10 minute intervals. An hour of video is recorded throughout the day,

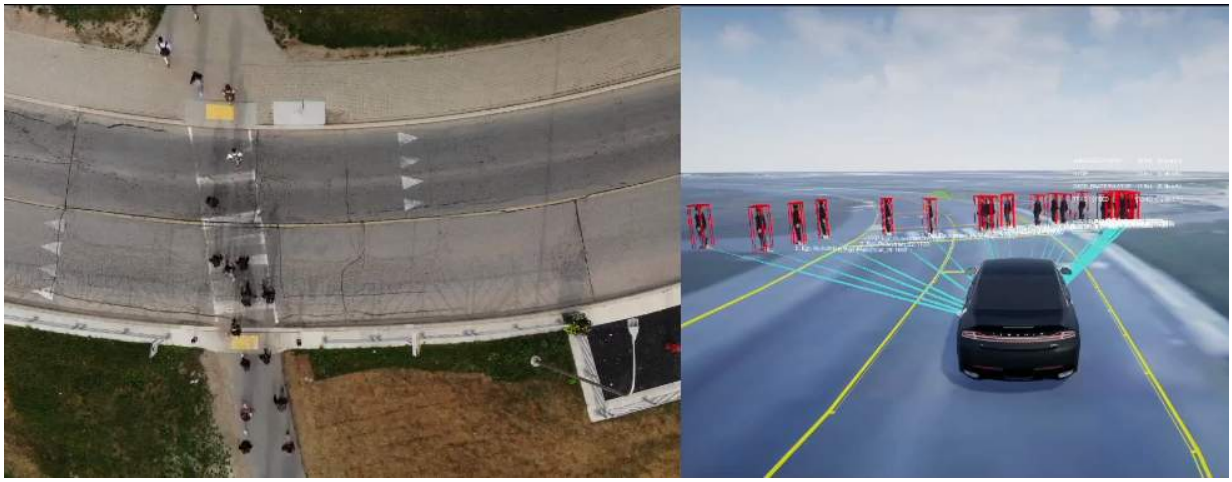


Figure 4.6: Unreal simulation using trajectories recorded from above

recording times between classes when road traffic is the busiest.

The video is corrected for distortion, stabilized, georeferenced, and annotated with pedestrian and vehicle position, speed, and acceleration by DataFromSky [13]. The trajectories contain position in latitude-longitude, speed in kilometers per hour, and acceleration in meters per second.

4.2.2 Metrics and analysis

The goal of the analysis is to quantify exactly when drivers commit to crossing the crosswalk in relation to pedestrians. One of the metrics that correspond directly with the algorithm is the pedestrian distance after entry edge (PDAEE). This is the distance of the pedestrian from the side they entered when a driver decides it is appropriate to cross. In this scenario, the pedestrian is crossing from right to left in front of the driver on a right sided driving road. Figure 4.7 shows the metric. For the sake of this analysis, a driver commits to crossing when their acceleration reaches 0.3 meters per second squared and never decelerates throughout the crossing. Figure 4.8 shows the result where $n = 56$ with a mean of 3.36 meters, and standard deviation of 1.6 meters. Since the road is 8.5 meters in length, this results shows that drivers on average are comfortable with committing to cross just after the pedestrian has passed the middle of the lane. Most drivers do not wait for pedestrians to cross completely before crossing themselves. Comparing this result to the autonomous algorithm, the average driver is about as assertive as the most assertive

setting where the crosswalk is divided into 2 equal halves.

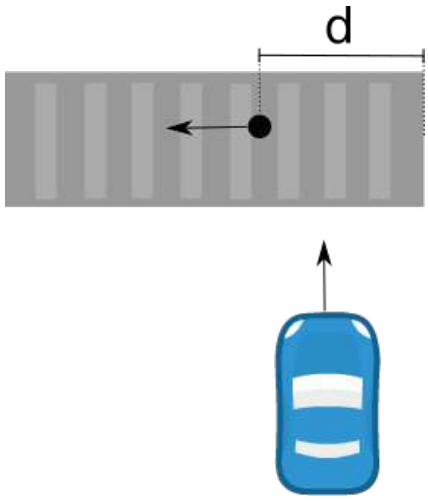


Figure 4.7: PDAEE d when driver commits to crossing

Another metric exploring driver behavior with pedestrians on the crosswalk is the distance of a pedestrian from the leaving edge of the road (PDFLE). Figure 4.9 illustrates the pedestrian and driver interaction. Figure 4.10 shows the result where $n = 35$ with a mean of 2.21 meters, and standard deviation of 1.15 meters. Since half of a lane is 2.125 meters, drivers on average commit to crossing after the pedestrian has crossed half the lane. Comparing the average driver behavior to the autonomous algorithm on this metric, they are about equal.

So far, the scenarios analyzed correspond to pedestrians on the crosswalk. The third metric: distance-speed before entry edge DSBEE illustrated in Figure 4.11 captures the distance-speed relation at the moment a driver commits to cross when the pedestrian is approaching the crosswalk. Figure 4.12 illustrates the interaction where $n = 52$. As shown by the linear interpolation, there is a positive correlation between speed and distance. When a pedestrian is close to the crosswalk walking slowly, a driver is comfortable with crossing. Drivers are not comfortable with crossing when pedestrians are close to the crosswalk and walking quickly. Some drivers decided to quickly pass the crosswalk maintaining speed at approximately 20 kph rather than stopping for a pedestrian that is about 2 m away from the curb walking at 6 kph. Comparing the results of this metric to the autonomous system, the autonomous system can be configured to be as assertive as the most assertive drivers. This can be done by reducing the radius of the circle in section 3.2.1 and removing the reduced maximum speed limit when approaching the crosswalk.

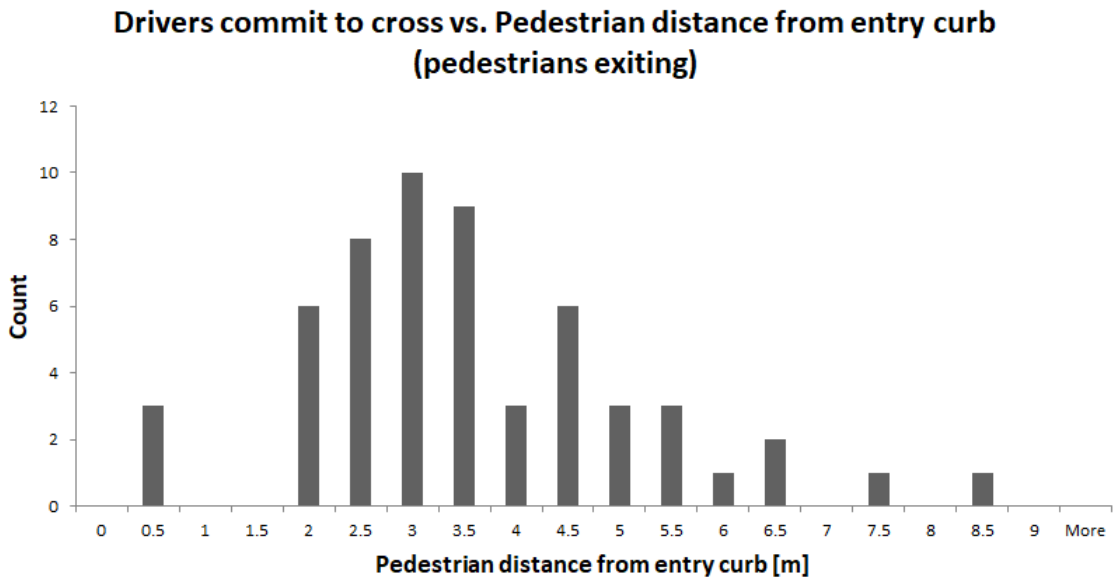


Figure 4.8: Histogram of distances shown in figure 4.7. mean=3.36 m; stdev=1.6 m

4.2.3 Real World GeoScenarios

Testing the autonomous algorithm on real world pedestrian behavior is accomplished by translating video recordings and trajectories into GeoScenarios. The key attributes required to do so is pedestrian latitude-longitude, instantaneous speed in moments of time, and time stamps. This section describes several interesting scenarios from the recordings. Vehicles and cyclists are excluded from the simulation as the main focus is validating the autonomous driving behavior with respect to pedestrians only. Original trajectories from video are labelled at 30 Hz but pedestrian trajectories are extracted at 0.5 Hz to limit the GeoScenario file size. Simulation interpolates between the points to achieve a smooth trajectory that closely resembles the original. GeoScenario position triggers are used with time delay to synchronize the start of movement of each pedestrian with the recording. Without manual additions to pedestrian trajectories, they can only start and end where they are first and last seen in the video. As a result, they appear frozen before they start walking and after they have crossed the crosswalk. The videos can be found at the same link [10].

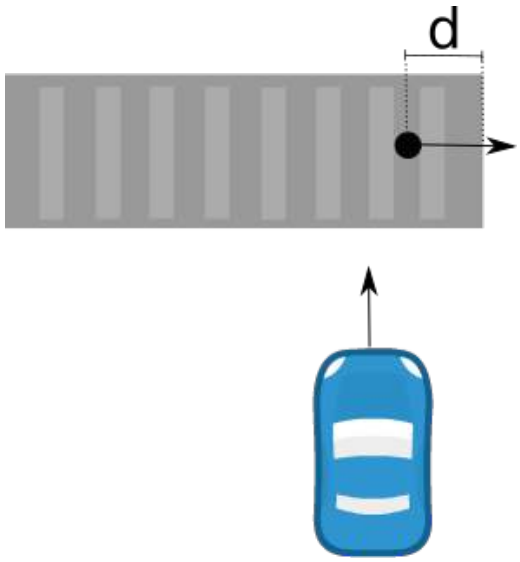


Figure 4.9: PDFLE d when driver commits to crossing

Stream of Pedestrians With Gap

In this scenario a stream of pedestrians cross from right to left in front of a driver stopped at the stop line. The driver waits until there is a sizable gap between pedestrians and crosses before all pedestrians have passed. As seen in section 4.2.2, the driver does not wait for pedestrians to fully cross before passing. The expected behavior of the autonomous system depends on tuning the assertiveness parameters. The autonomous system is tested with the crosswalk dividing line set to half of the crosswalk and the result is very similar to the real driver. The system crosses the crosswalk at almost the exact same time as the real driver.

Sparse Group of Pedestrians

In this scenario, a group of runners run along the sidewalk without crossing while a couple pedestrians cross from right to left. There is no vehicle in the scene to compare with actual driver behavior, but a reasonable person is expected to cross when the last pedestrian has crossed halfway. In the simulation, the autonomous system approaches the crosswalk as pedestrians are on it. Before the last pedestrian crosses, the system nudges forward a little but stops for the entering pedestrian. The running group does not appear to affect behavior. The system commits to crossing after the pedestrian is half way across the

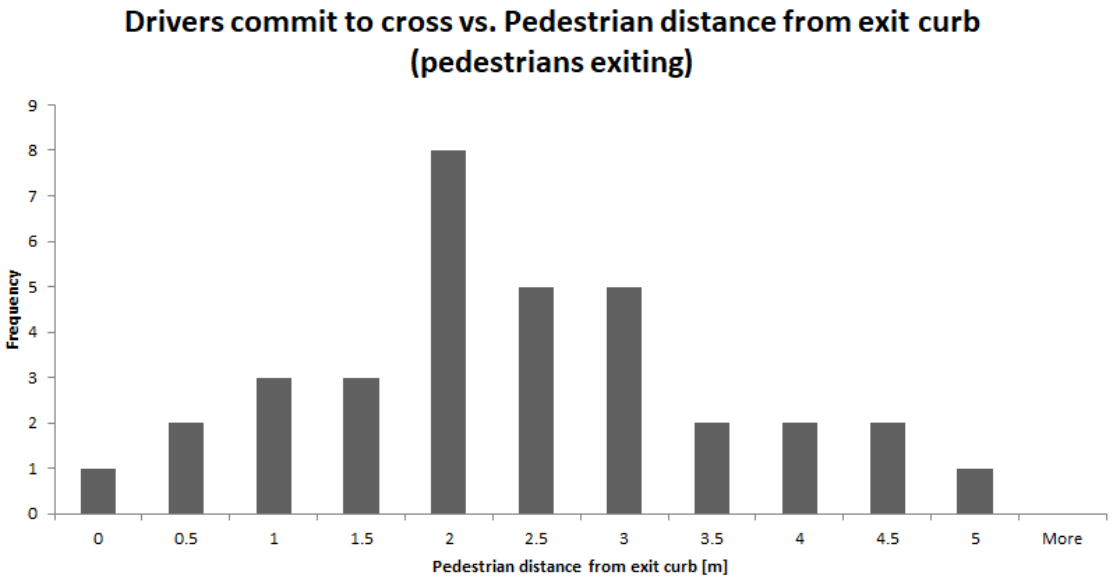


Figure 4.10: Histogram of distances shown in figure 4.9. mean=2.21 m; stdev=1.15 m

crosswalk.

Large Group of Pedestrians

In this scenario, a large group of pedestrians cross the crosswalk with very little gap in between them. A human driver is waiting at the stop line and commits to cross after the last pedestrians has crossed half way. The autonomous system waits like the real driver until the last pedestrians have crossed halfway. Since some of the pedestrian's trajectories do not extend away from the edge of the road far enough, the system recognizes a stopped pedestrian potentially waiting to enter and slows down a little. The system finishes the cross almost exactly like the driver.

Sparse Pedestrians With Small Gap

This scenario is similar to the first sparse pedestrians with gap scenario except the gap is smaller. In the scene, a driver approaches the crosswalk with pedestrians crossing and crosses with a gap of about 9 meters. The entering pedestrian is about 3.5 meters away from the vehicle when the vehicle is on the crosswalk. The autonomous system successfully

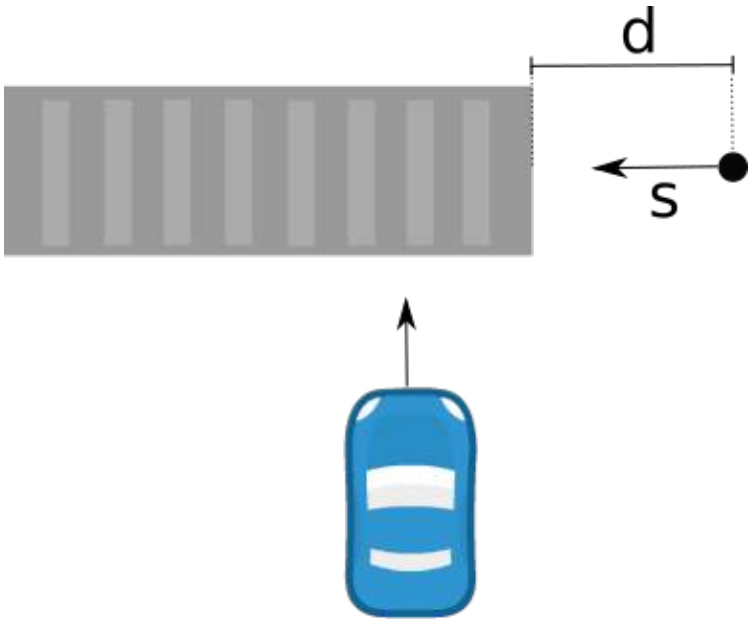


Figure 4.11: Pedestrian distance d and speed s when driver commits to crossing (DSBEE)

crosses the same gap. The system begins accelerating a little later than the real driver resulting in a smaller gap between the entering pedestrian and the vehicle.

Conservative Vs. Assertive

This scenario demonstrates the difference between the most assertive setting and the most conservative setting. In the scene, the human driver waits at the stop line until the very last pedestrian has crossed half way. In simulation, the most assertive driving behavior inches forward when there seems to be a gap large enough to pass. When entering pedestrians get too close, the system stops and waits for another gap. This repeats until the vehicle is far up enough that a pedestrian walks into the vehicle. Since the simulation only plays back prerecorded trajectories, the pedestrians cannot react to different driving behaviors. In the real world, a reasonable pedestrian is expected to recognize the vehicle is crossing and slow down for it to pass. The most conservative setting behaves very similar to the real driver by waiting until the very end of the sequence. The system waits until the pedestrians are almost completely off the crosswalk before crossing.

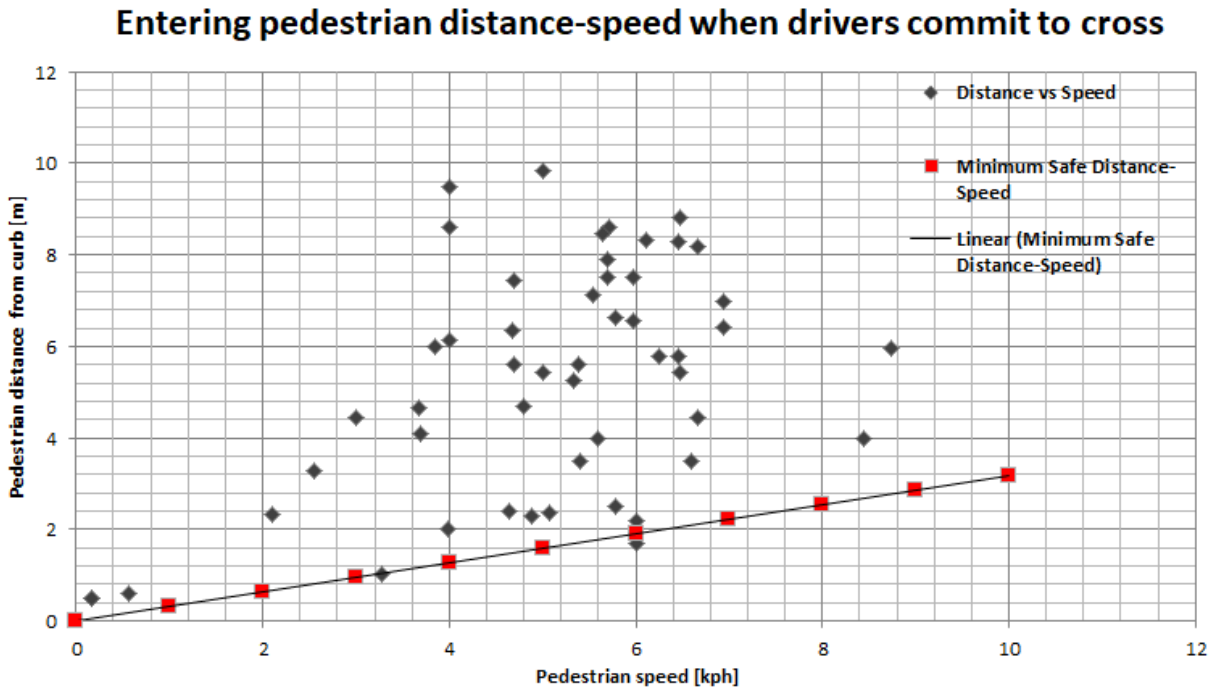


Figure 4.12: Approaching pedestrian distance and speed at the moment when human drivers commit to cross. Minimum safe distance is with respect to a theoretical vehicle 6.4 m from crosswalk marking travelling at 20 kph.

Pedestrians From Both Sides

This is a common scenario with pedestrians crossing from both sides with small gaps forcing drivers to wait. In the scenario, pedestrians cross one by one from both sides of the road and a driver waits to cross at the stop line. The driver crosses at the end of the sequence when no other pedestrians enter. With the crosswalk divider set to the whole crosswalk, the autonomous system also waits until the end of the sequence to cross and accelerates just after the human driver begins accelerating.

Chapter 5

Conclusion

The pedestrian crosswalk scenario is one of many scenarios an autonomous vehicle can encounter with each one slightly different from the others. Key features of a crosswalk scenario can be extracted so that a general approach for handling crosswalks can be made on the abstraction.

This work builds off of the lanelet representation of road geometry and expresses crosswalks using a minimal set of elements that are congruent with how they appear in real life. The simplicity of the crosswalk representation allows it to be applied to all marked pedestrian crossings. Reliably known features available from the lanelet map are used to localize pedestrians and other vehicles to construct an abstraction of the crosswalk scenario. The abstraction to a few time predicates is a powerful technique that is robust to variability in number of pedestrians and their velocities.

Adoption of the lanelet map format has been a simple way of providing knowledge of the drivable road surface with sufficient detail. The process of creating an accurate and precise lanelet map has proven to be difficult with inherent uncertainty from global positioning and offsets in satellite images, however. Consequently map procuring requires a great deal of attention during verification analogous to ground truth labelling for data sets in the field of machine learning. In theory under ideal conditions, this approach of mapping works without issue across the entire globe; however, errors from atmospheric delays, multipath signal interference, satellite position calculation, and clock correction intermittently corrupt the reliability of the map. Mapping based off satellite images are associated with challenges of their own in the process of flattening a globe onto a 2 dimensional, potentially distorted, image and assigning accurate coordinates to every pixel.

This work also introduces a method of behavior planning for the pedestrian crosswalk

scenario. The crosswalk scenario is a relatively simpler one given that the position of crosswalks are well defined and pedestrian behavior around crosswalks are generally easily predictable. These constraints reduce the variability of the scenario to a degree where a carefully engineered solution can handle it. It is intractable to implement a solution for all possibilities of human imagination; however, the approach described in the work reduces the infinite set down to a few critical predicates making it appropriate for most scenarios. Additional effort can be put toward analyzing passenger acceptance of different driving styles and tuning the assertiveness of behavior planning. For future improvement, single crosswalk behavior planning can be extended to handle crosswalks at intersections in a similar way. At a 4 way stop sign intersection for example, the time span specifying when an area is free to cross can be extended to include multiple instances of crosswalks in ego's planned path. For instance, one time span can be calculated for the crosswalk immediately in front of ego and another time span can be calculated for the crosswalk on the other side of the intersection.

The challenges associated with mapping are mostly dealt with real time kinematics and post process verification. Given the experience after working with lanelet maps and global positioning, it is recommended in future iterations to reduce the dependence for small details. Instead, it would be better to localize using Global Positioning System (GPS) to a general area of a prebuilt map and leverage SLAM techniques for finer details like curb edges. The algorithm introduced in this thesis is one of many possible solutions to a rather structured problem. It sets a baseline benchmark for future development perhaps with the use of reinforcement learning or methods that incorporate advanced perception like pose recognition or trajectory prediction.

References

[1] Md Zahangir Alom, Tarek M Taha, Christopher Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Brian C Van Esesn, Abdul A S Awwal, and Vijayan K Asari. The history began from alexnet: a comprehensive survey on deep learning approaches. *arXiv preprint arXiv:1803.01164*, 2018.

[2] Pichamon Anantasech and Chotirat Ann Ratanamahatana. Enhanced weighted dynamic time warping for time series classification. In *Third International Congress on Information and Communication Technology*, pages 655–664. Springer, 2019.

[3] Mateusz Loskot Adam Wulkiewicz Menelaos Karavelas Vissarion Fisikopoulos Barend Gehrels, Bruno Lalande. *Boost Geometry Documentation*, 2019 (accessed 2019-07-03). https://www.boost.org/doc/libs/1_65_1/libs/geometry/doc/html/index.html.

[4] Sven A Beiker. Legal aspects of autonomous driving. *Santa Clara L. Rev.*, 52:1145, 2012.

[5] Philipp Bender, Julius Ziegler, and Christoph Stiller. Lanelets: Efficient map representation for autonomous driving. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 420–425. IEEE, 2014.

[6] Donald J. Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD Workshop*, 1994.

[7] Frederic Bouchard. Expert System Program Synthesis for Urban Behaviour Planning. Master's thesis, University of Waterloo, Waterloo, Ontario, Canada, 2019.

[8] Guillaume Bresson, Zayed Alsayed, Li Yu, and Sébastien Glaser. Simultaneous localization and mapping: A survey of current trends in autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 2(3):194–220, 2017.

[9] Xiaohua Cao, Daofan Liu, and Xiaoyu Ren. Detection method for auto guide vehicles walking deviation based on image thinning and hough transform. *Measurement and Control*, page 0020294019833073, 2019.

[10] Edward Chao. Unreal simulations pedestrian at crosswalk, 2019. https://doi.org/10.5683/SP2/MRMSEY.

[11] Sihan Chen, Libo Huang, and Jie Bai. Robust multi-lane detection and tracking in temporal-spatial based on particle filtering. Technical report, SAE Technical Paper, 2019.

[12] Jaewoong Choi, Junyoung Lee, Dongwook Kim, Giacomo Soprani, Pietro Cerri, Alberto Broggi, and Kyongsu Yi. Environment-detection-and-mapping algorithm for autonomous driving in rural or off-road environment. *IEEE Transactions on Intelligent Transportation Systems*, 13(2):974–982, 2012.

[13] DataFromSky. *Advanced traffic analysis of aerial video data*, 2019 (accessed 2019-08-10). http://datafromsky.com/.

[14] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[15] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, Dec 1959.

[16] Centers for Disease Control and Prevention. *Impaired Driving: Get The Facts*, 2019 (accessed 2019-07-01). https://www.cdc.gov/motorvehiclesafety/impaired_driving/impaired-drv_factsheet.html.

[17] Epic Games. *Unreal Engine Landing Page*, 2019 (accessed 2019-07-03). https://www.unrealengine.com/en-US/.

[18] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012.

[19] T. Gindele, S. Brechtel, and R. Dillmann. Learning driver behavior models from traffic observations for decision making and planning. *IEEE Intelligent Transportation Systems Magazine*, 7(1):69–79, Spring 2015.

[20] Andrew Goldberg and Chris Harrelson. Computing the shortest path: A* search meets graph theory. Technical Report MSR-TR-2004-24, July 2004.

[21] Andrew V Goldberg, Haim Kaplan, and Renato F Werneck. Better landmarks within reach. In *International Workshop on Experimental and Efficient Algorithms*, pages 38–51. Springer, 2007.

[22] Mordechai Haklay and Patrick Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing*, 7(4):12–18, 2008.

[23] Mohammed M Hamed. Analysis of pedestrians behavior at pedestrian crossings. *Safety science*, 38(1):63–82, 2001.

[24] Yoriyoshi Hashimoto, Yanlei Gu, Li-Ta Hsu, and Shunsuke Kamijo. Probability estimation for pedestrian crossing intention at signalized crosswalks. In *2015 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pages 114–119. IEEE, 2015.

[25] Hosking, Bryce Antony. Modelling and model predictive control of power-split hybrid powertrains for self-driving vehicles, 2018.

[26] NovAtel Incorporated. *VEXXIS Antennas GNSS-502*, 2019 (accessed 2019-07-02). https://www.novatel.com/assets/Documents/Papers/GNSS-502-PS.pdf.

[27] Tesla Incorporated. *Tesla Autopilot*, 2019 (accessed 2019-07-02). https://www.tesla.com/en_CA/autopilot.

[28] Shinpei Kato, Eijiro Takeuchi, Yoshio Ishiguro, Yoshiki Ninomiya, Kazuya Takeda, and Tsuyoshi Hamada. An open approach to autonomous vehicles. *IEEE Micro*, 35(6):60–68, 2015.

[29] C. G. Keller and D. M. Gavrila. Will the pedestrian cross? a study on pedestrian path prediction. *IEEE Transactions on Intelligent Transportation Systems*, 15(2):494–506, April 2014.

[30] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[31] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander. Joint 3d proposal generation and object detection from view aggregation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8, Oct 2018.

[32] Shaoshan Liu, Jie Tang, Zhe Zhang, and Jean-Luc Gaudiot. Computer architectures for autonomous driving. *Computer*, 50(8):18–25, 2017.

[33] Gaston Maspero. *Manual of Egyptian Archaeology and Guide to the Study of Antiquities in Egypt*. Grevel, 1895.

[34] Anish Mittal and Richard Kwant. Method and apparatus for pixel based lane prediction, February 19 2019. US Patent App. 10/210,403.

[35] Joseph Needham. *Science and civilisation in China*, volume 5. Cambridge University Press, 1976.

[36] Tom Occhino, Jing Chen, and Pete Hunt. Hacker way: rethinking app development at Facebook. 2014. https://www.youtube.com/watch?v=nYkdrAPrdcw, https://facebook.github.io/flux/.

[37] Ministry of Natural Resources and Forestry. *Geodesy*, 2019 (accessed 2019-07-02). https://www.ontario.ca/page/geodesy.

[38] Ontario Ministry of Transportation. *Driving laws for pedestrian crossovers and school crossings*, 2019 (accessed 2019-08-24). http://www.mto.gov.on.ca/english/safety/pedestrian-safety.shtml.

[39] Ryosuke Okuda, Yuki Kajiwara, and Kazuaki Terashima. A survey of technical trend of adas and autonomous driving. In *Technical Papers of 2014 International Symposium on VLSI Design, Automation and Test*, pages 1–4. IEEE, 2014.

[40] F. Poggenhans, J. Pauls, J. Janosovits, S. Orf, M. Naumann, F. Kuhnt, and M. Mayr. Lanelet2: A high-definition map framework for the future of automated driving. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 1672–1679, Nov 2018.

[41] Rodrigo Queiroz, Thorsten Berger, and Krzysztof Czarnecki. GeoScenario: An open dsl for autonomous driving scenario representation. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019.

[42] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[43] Atri Sarkar, Krzysztof Czarnecki, Matt Angus, Changjian Li, and Steven Waslander. Trajectory prediction of traffic agents at urban intersections through learned interactions. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8. IEEE, 2017.

[44] Junaed Sattar and Jiawei Mo. Vehicle lane detection system, March 7 2019. US Patent App. 16/124,502.

[45] Random Science Tools. *Stopping Distances*, 2019 (accessed 2019-07-13). https://www.random-science-tools.com/physics/stopping-distance.htm.

[46] Association For Safe International Road Travel. *Road Safety Facts*, 2019 (accessed 2019-07-01). https://www.asirt.org/safe-travel/road-safety-facts/.

[47] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, MN Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.

[48] Van Gennip, Matthew. Vehicle dynamic modelling and parameter identification for an autonomous vehicle, 2018.

[49] Alexandra Willis, Nathalia Gjersoe, Catriona Havard, Jon Kerridge, and Robert Kukla. Human movement behaviour in urban spaces: Implications for the design and modelling of effective pedestrian environments. *Environment and Planning B: Planning and Design*, 31(6):805–828, 2004.

[50] Yinan Zheng, Thomas Chase, Lily Elefteriadou, Bastian Schroeder, and Virginia P Sisiopiku. Modeling vehicle–pedestrian interactions outside of crosswalks. *Simulation Modelling Practice and Theory*, 59:89–101, 2015.

[51] Julius Ziegler, Philipp Bender, Markus Schreiber, Henning Lategahn, Tobias Strauss, Christoph Stiller, Thao Dang, Uwe Franke, Nils Appenrodt, Christoph G Keller, et al. Making bertha drive: An autonomous journey on a historic route. *IEEE Intelligent Transportation Systems Magazine*, 6(2):8–20, 2014.

Appendix

Appendix A

Lanelet Map Generation

A.1 Map Generation

The lanelet format having been based off of osm is inherently reliant on GNSS. To satisfy the basic description of a lanelet node, there must be at least a latitude and longitude with respect to the World Geodetic System (WGS 84) default datum used in GPS. The number of techniques for acquiring this data is discussed in the following sections. Fundamentally, data that is collected are latitude-and-longitude points that determine the position of each node in the lanelet map. Semantics such as regulation, road network connections, and how nodes relate to one another are typically incorporated in post processing. Generating a lanelet map is roughly a 5 step process through a series of automated and manual input as well as verification. The process was developed and improved as part of this thesis and include: Road network segmentation planning and labelling, latitude-and-longitude points acquisition, lanelet assembly through script, fine manual adjustments, and in-vehicle testing. A map can theoretically be crafted completely manually but labeling and automation significantly reduce production time.

A.1.1 Hardware Based Mapping

This method of map data collection uses the VEXXIS GNSS - 502 antenna [26] installed on the autonomous vehicle alongside a base station setup. The benefit of direct measurement using the vehicle is the data that is used for generating the map is theoretically equivalent with the same hardware to the data that is available during run time. This method

does not introduce conversion errors between different measurement equipment. Real Time Kinematics (RTK) enhances the precision of latitude - longitude data from GNSS. Firstly, a static base station is set up and left to calibrate over a period of time. The base station continuously receives position estimates over the course of a couple of hours or days depending on the desired accuracy. Since its true position is fixed for the entire duration, it uses all noisy position estimates and produces a single "true" estimate of its position. Once it has its "true" position, the difference between every instantaneous position estimate and "true" position is deemed to be the instantaneous error. All GNSS antennas within about a 10 kilometer radius of the base station are likely to have reception to the same satellites. Therefore, the instantaneous error can be applied to any receiver near the base station to acquire a more precise position. In practice, the setup attenuates the standard deviation of position to as low as single centimeter accuracy. After the base station has completed its calibration, another GNSS receiver installed on the vehicle receives satellite positioning estimate as well as real-time corrections from the base station over radio. Stack coordinate frame transformations then converts the position from the antenna to the center of the rear-axle of the vehicle. This position is referred to as the origin point of the base-link frame.

Labeling - Lane Indicator Counters

Labels provide an interface for organizing individual latitude-and-longitude points into semantic groups; indicating the start and end of ways, intersections, and lanelets. Segmenting the road with labels correspond to segmenting the road network into lanelets. The first type of labels are Lane Indicator Counters (LIC)s, which are used to distinguish individual lanes on multilane roads. Additionally, LICs represent the adjacency between lanes. Numbering starts at 0 with the lane line that divides direction of traffic flow. One direction of traffic is chosen arbitrarily to be identified with positive numbers and the other with negative numbers. The polarity is simply to distinguish between direction of traffic but not the direction itself. Figure A.1 shows how a 4 lane road is labelled with LICs. Right side lanes are chosen at random to be numbered positively. The polarity of LIC assignment only matters for unidirectional roads because boundary line 0 is acquired differently. Consequently, unidirectional roads have only positive LICs.

Labeling - Road Boundaries

Since a lanelet's properties are invariable throughout the entire length of the lanelet, a new lanelet begins when road properties change. Lanelets that compose an intersection

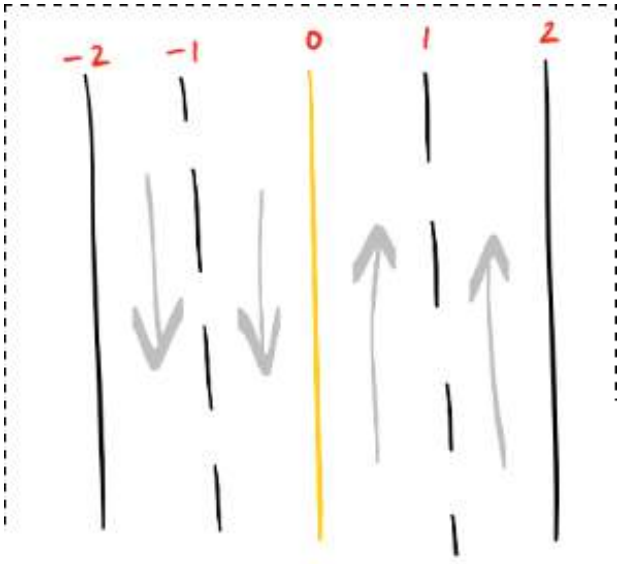


Figure A.1: Lane indicator counters on a 4 lane road

are uniquely separate from other lanelets therefore all lanelets approaching and leaving an intersection end and begin at the boundary of the intersection. Part of planning the road boundary labels is ensuring all segments are identifiable by a unique id.

Data collection using hardware relies on continuous position estimates from the system described in section A.1.1. A ROS node subscribes to the global positioning topic and records a number of properties. To avoid spamming the output file with redundant positions when the vehicle is stationary, a threshold is set to only record points when the straight line Euclidean distance from the previously recorded point is at least 1 meter. The recorded positions correspond to the center of the rear axle; the origin of the base-link coordinate frame.

In addition to subscribing to the global positioning topic, the ROS node also listens for strings published on a separate topic. The published labels are tagged along with recorded points by issuing publish commands on a separate linux terminal. The labels that start and end a segment are the same. Each segment is identified by a string under the format: *seg_x_y* where x is the unique segment number and y is the LIC number. Segments that make up an intersection have an additional field *seg_x_y_z* where z is the unique intersection number. Segments corresponding to the same intersection identifier are assembled together as an intersection object. Figure A.2 shows a high level plan around a loop.

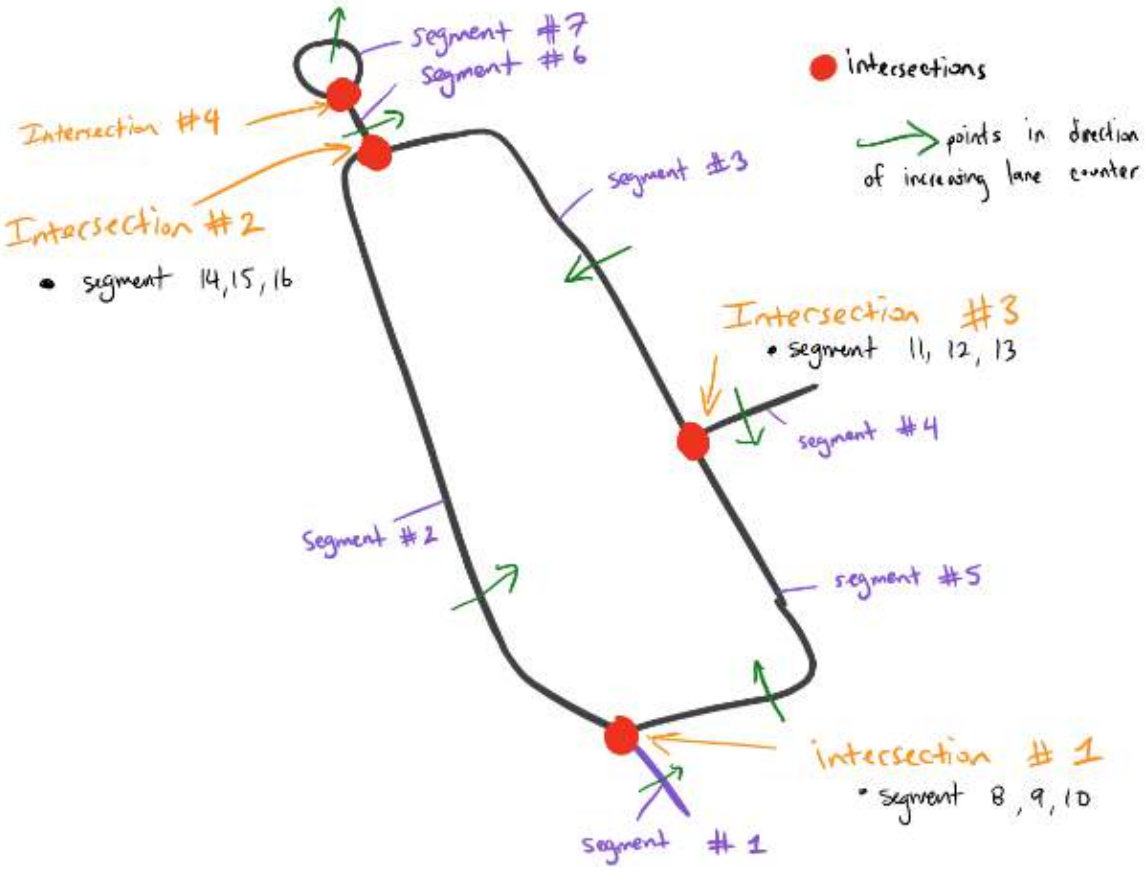


Figure A.2: Example plan for mapping Colby Drive in Waterloo, Ontario

Labeling - Stop signs

One of the simplest regulatory elements to include in a lanelet map is a stop sign. A stop regulation is represented by a single point and a 2-point way. The single point represents the position of the stop sign, and the 2-point way represents the stop line. Stop relations are captured in the data collection process with labels of the format: *stop_x* where x is the unique segment number that the stop relation affects. The specific lanes that are affected are selected by the post processing script. It identifies the side of the road that ends nearest to the sign position and applies the stop regulation on each lanelet on that side. Capturing more elaborate regulatory elements involving additional elements is not automated but can be incorporated in the map with an editor like JOSM.

Collection Process

After creating a collection plan including all labels, the lane lines and curb boundaries are tracked using the wheels of the vehicle. While alternative methods for lane line detection and localization use cameras and lidars [34, 44, 9, 11], this direct method is sufficient for constructing small-scale maps for research purposes.

In general, curbs and lane lines are localized by aligning the front right wheel of the vehicle to the desired position. The front left wheel is used for tracking when the LIC is 0 on a unidirectional road. Assuming lanes are approximately equal in width, the dividing lane line on bi-directional roads do not need to be tracked since missing ways can be interpolated in post processing. Once the vehicle is aligned in the proper position, the waypoints collection node is started and records both the latitude-longitude position as well as labels published on the topic /anm waypoint. Labels are published using the command rostopic pub -1 /anm_waypoint std msgs/String A where A is the label i.e. 'seg_1_1' to indicate segment 1 with LIC 1. The collecting vehicle drives, tracking a line and if the road is blocked by a parked vehicle, the label 'blockstart' is published. After maneuvering around the blocking vehicle, the label 'blockend' is published. Points are interpolated at 1 meter intervals for elevation and latitude-longitude between blockstart and blockend. After reaching the end of a segment, the same label that started the segment is published, marking the end of the segment.

Stop sign regulations are captured by recording a point near the stop sign and providing the corresponding label described above. Other regulatory elements requiring additional information are not easily captured with this method so they are added in JOSM after the map has been built.

The labelling process described in the previous paragraphs was improved on over multiple revisions by taking out the labelling from raw data collection. Labels are added in JOSM resulting in 2 benefits: significant reduction of time in vehicle, and clean lines without interruption. This change changed the collection process to a simple tracking-the-curb exercise. There is no need to frequently stop for adding labels and introducing position estimation errors.

A.1.2 Post processing

Post processing automates the majority of the lanelet map building process by parsing the labels embedded in raw latitude-and-longitude points. Raw data points are stored in GPX file format and is parsed to extract position and labels. The labels are indicators of where

a new way begins and ends. The labels also indicate which ways should represent the left and right boundary of a lanelet.

Since the recorded position is the origin of the base-link frame, the points are first projected to the points where the front wheels touch the ground. Depending on the LIC, the points are either projected to the front right wheel if the LIC is not 0, or to the front left wheel if it is 0. Since the choice of projecting to one of the wheels is only determined offline when evaluating labels, the internal frame transforms of the stack from base-link to front wheels are unavailable. A measurement from base-link to the front wheels provide the relative heading and distance used for projection. Heading between consecutive points is calculated, rotated toward either the front left or right wheel, and translated forward from the center of the rear axle to the wheel while accounting for altitude. Figure A.3 illustrates the point projection process. Imperfections appear with this method of data collection around sharp right turns. Because of vehicle dynamics, the front right wheel is incapable of sticking to the curb without the back wheel leaving the road. In practice, this resulting offset has not been an issue.

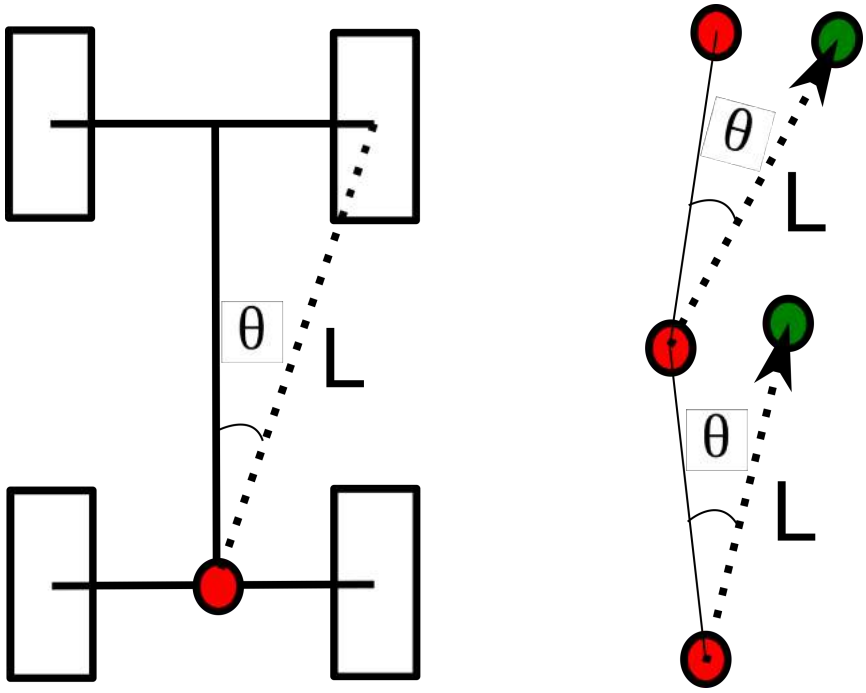


Figure A.3: How raw data points are repositioned offline in post processing

Missing ways are automatically generated when adjacent ways are present on both sides.

On simple 2 lane roads separated by a single dividing line, only curbs are tracked and the dividing line is interpolated between the curbs. In similar fashion, lane center lines are generated from the left and right bound after a lanelet relation is created. The process can be improved with dynamic time warping [6, 2] to produce equally spaced points aligned side by side along each way.

Stop lines are marked by finding the closest end of the segment to the recorded stop sign position. Since the stop label is stop_x where x is the segment number, it does not explicitly identify where the stop line should be drawn. The closest end of the segment to the stop sign is found and a stop line is defined as 2 points from LIC 0 to the largest in magnitude. Consequently the stop relation affects all lanelets with the same drive direction.

The rest of post processing builds elements from simpler ones and connects them together. Ways reference nodes, lanelets reference ways, intersection objects reference lanelets, and lanelets reference regulatory elements. To represent the road network graph, the final setup sets the next, previous, left, and right references between lanelets.

Once the lanelet map is built, the map is revised and tested both in JOSM, simulation, and on the vehicle, and a final automated check for artifacts identifying polylines that exhibit unnatural properties of angle and distance between points.

A.1.3 Satellite Imagery based data collection

One alternative to using the vehicle as the measurement instrument is collecting latitude-longitude points from satellite imagery. This method of mapping is tested to compare against the points collected from direct measurement. An advantage of referencing an image is that the image remains constant for all data points. So long as there is negligible distortion in the image, all points are reasonably accurate relative to each other. This contrasts from direct measurement where every point is more independent because it is affected by instantaneous reception to satellites. Surrounding buildings can reflect and block reception to the antenna resulting in greater standard deviation at certain areas of the map. The disadvantage of using satellite imagery even when it is geo-referenced is that the image may be out of date with the current state of the road.

For the initial viability test, geodetic benchmarks laid out around the University of Waterloo are surveyed to confirm their location. These geodetic marks are mapped to highly accurate longitude, latitude, and elevation available on the national geodetic website [37]. Plotting the marks' latitude-longitude on a high definition satellite image allows us to inspect for image distortion.

The ArcGIS software provides highly detailed satellite imagery for plotting points directly on the curb edge. Figure A.4 shows the curb annotation of a four way stop sign intersection as an example. In addition, road markings including crosswalks and stop lines are clearly visible in the satellite image. It can be used for accurate relative positioning whereas acquiring road markings by positioning the wheel is prone to human error. After tracking points in the software, map generation progresses similarly as described previously with point labeling. Since points already track the curb, they do not need to be projected like in section A.1.2.

Figure A.4: Curb lines traced in ArcGIS with visible road markings

Although the satellite image may possess an overall shift with respect to the real road, points are geometrically stable. Alignment verification compares the visible curbs in LIDAR scans with the map. With post processed GPS correction, the "true" LIDAR scan of curbs is compared against the map points from satellite image and the entire map is adjusted to correct for misalignment.

The main difference between positions acquired by hardware and points from high definition satellite imagery is local consistency. Every position recorded by the vehicle throughout a collection is subject to constant fluctuation in accuracy and satellite reception. Error is also introduced in the process of transformation from base-link to wheel. It is also subject to the driver's ability to track lines consistently. Positions recorded straight from satellite imagery is locally consistent and does not require further transformation other than a global shift. In practice, the observed difference between vehicle collection and image collection is fewer manual adjustments of individual points, and smoother lines when mapping based on image.

Appendix B

Rule Engine Crosswalk Rules

Figure B.1 is a snippet of the source code in the rule engine for identifying time window conflict. Variable *newState.crosswalk.timeTillClear* is the upper bound list and *newState.crosswalk.timeTillPedestrianEnters* is the lower bound list described in algorithm 1.

```
const ttc = newState.crosswalk.timeTillClear;
const ttpe = newState.crosswalk.timeTillPedestrianEnters;

if(isNearbyCrosswalk(ego) && ttc.length === ttpe.length) {
    const threshold = hasSpeed(ego)
        && __NAVIGATE_CROSSWALK__
        || __NAVIGATE_CROSSWALK_STOP__;
    const lowerBound = newState.crosswalk.distance / Math.max(KMH_TO_MS(ego.speed), 1);

    const range = {
        min: lowerBound,
        max: lowerBound + threshold
    };

    const bounds = ttc.map((time, idx) => ({
        min: ttpe[idx],
        max: time
    }));

    crosswalkIsConflicted = !!bounds.find(bound => OVERLAP_BOUND(range,  bound));
 }
```

Figure B.1: Rule Engine source code for identifying time window conflict

Set IV – Crosswalk

System Id : RCW1
Decision : track-speed
Constraint: Reduced Speed
Goal: Reduce ego's speed when nearby a crosswalk.

```
ALL {
  ego {
    location: {
      SOME-OF {
        approaching: 'crosswalk',
        at: 'crosswalk',
        on: 'crosswalk'
      }
    }
  }
}
```

System Id : RCW2
Decision : decelerate-to-halt
Constraint: Crosswalk
Goal: Yield to pedestrians on the crosswalk.

```
ALL {
  ego {
    location: {
      approaching: 'crosswalk'
    }
  },
  travel: {
    crosswalk: {
      timeTillPedestrianEnters: <= THRESHOLD,
      timeTillClear: >= THRESHOLD
    }
  }
}
```

System Id : RCW3
Decision : yield
Constraint: Crosswalk
Goal: Remain stationary until the crosswalk is freed

```
ALL {
  ego {
    location: {
      at: 'crosswalk'
    },
    speed: <= THRESHOLD
  },
  travel: {
    crosswalk: {
      timeTillPedestrianEnters: <= THRESHOLD,
      timeTillClear: >= THRESHOLD
    }
  }
}
```

System Id : RCW4
Decision : emergency-stop
Constraint: None
Goal: Warn the safety driver about a potential unavoidable collision at a crosswalk.

```
ALL {
  ego {
    location: {
      at: 'crosswalk'
    },
    speed: > THRESHOLD
  },
  travel: {
    crosswalk: {
      timeTillPedestrianEnters: <= THRESHOLD,
      timeTillClear: >= THRESHOLD
    }
  }
}
```

System Id : RCW5
Decision : emergency-stop
Constraint: None
Goal: Warn the safety driver about a potential unavoidable collision on a crosswalk.

```
ALL {
  ego {
    location: {
      on: 'crosswalk'
    }
  },
  travel: {
    crosswalk: {
      timeTillPedestrianEnters: <= THRESHOLD,
      timeTillClear: >= THRESHOLD
    }
  }
}
```

System Id : RCW6
Decision : follow-leader
Constraint: Vehicle + Reduced Speed
Goal: Keep following the leading vehicle with a reduced speed when nearby a crosswalk.

```
ALL {
  ego {
    location: {
      SOME-OF {
        approaching: 'crosswalk',
        at: 'crosswalk',
        on: 'crosswalk'
      }
    }
  },
  a-vehicle: {
    isLeading: true,
    location: {
      on: 'not-off-road'
    }
  }
}
```

System Id : RCW7
Decision : decelerate-to-halt
Constraint: Vehicle + Crosswalk
Goal: Yield to pedestrians on the crosswalk with a gap for the leading vehicle.

```
ALL {
  ego {
    location: {
      approaching: 'crosswalk'
    }
  },
  a-vehicle: {
    isLeading: true,
    location: {
      SOME-OF {
        approaching: 'crosswalk',
        at: 'crosswalk'
      }
      on: 'not-off-road'
    }
  }
  travel: {
    crosswalk: {
      timeTillPedestrianEnters: <= THRESHOLD,
      timeTillClear: >= THRESHOLD
    }
  }
}
```