Implementation of Pedestrian Recognition in Autonomous Vehicles

Kenneth Hobday, Jacob Yax, Tuan Nguyen
ECE434

Introduction:

In an ever evolving world the introduction of autonomous vehicles appears to be a question of when not if. This inevitability facilitates the necessity for safety procedures to maintain a safe environment not only for the driver but those around the driver. This can include pedestrians, who oftentimes require areas to cross major roads, it is then important that any functioning autonomous vehicle is able to recognize this risk and utilize the correct measures to ensure safety. In a non autonomous environment traffic related incidents already account for nearly 7,500 pedestrian fatalities each year. Through this project we aim to decrease this number drastically in an autonomous world, through the use of a ROS node and image learning model.

Risk Significance:

Traffic related deaths are already exceedingly high, not just in the United States but in many parts of the world. While not all these are on account of pedestrian related issues, a large number, as stated above, are. A high level of safety is required for an autonomous vehicle to even be street legal and part of that safety is pedestrian safety, thus the implementation of an algorithm such as ours is of utmost importance.

The need for pedestrian safety is also facilitated by the fact that crossing rarely if ever has some form of protection, with pedestrians oftentimes just walking an unguarded path in the middle of the street. This means that a vehicle that has no inclination to stop will run down a pedestrian crossing the street at a very high speed.

Ideally the implementation of this algorithm would also protect the driver as well. This is ensured by having the vehicle decelerate at a respectable speed, to mitigate the chance of the driver getting whiplash. Decelerating at a respectable speed, will also prevent the chance for spin outs as can sometimes be the case when a vehicle is moving at high speeds and tries to stop quickly, of course pedestrians are not always predictable, and thus this deceleration will not always happen slowly if say a pedestrian runs out into the crosswalk.

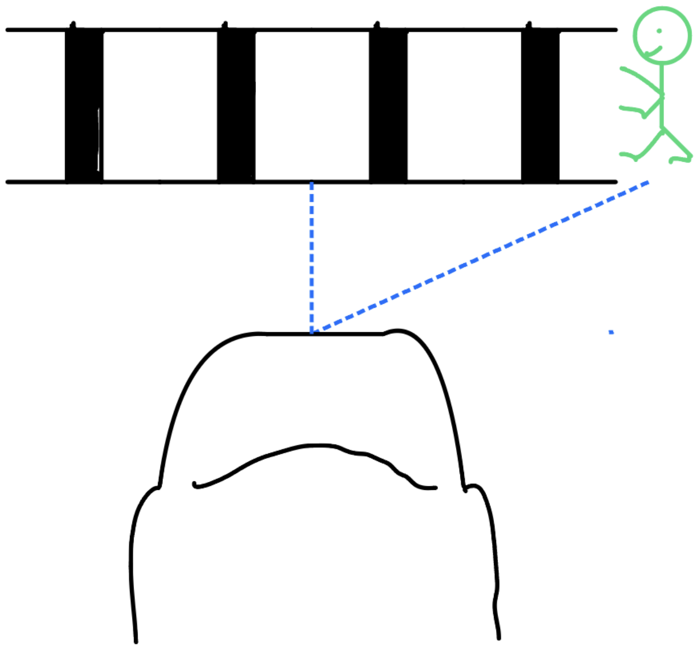


Figure 1. Ideal Implementation of Crosswalk Deceleration

<u>Mitigation:</u>

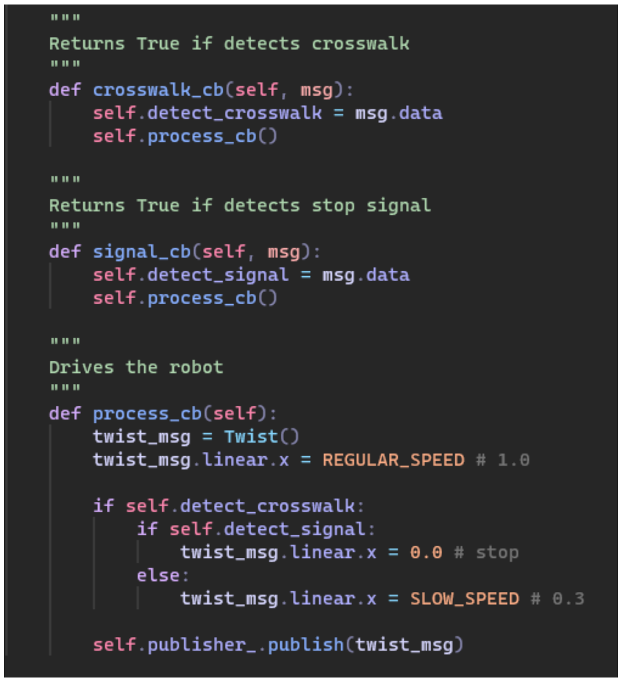
In an ideal scenario autonomous vehicles possess the ability to sense not just the lines that make up a pedestrian crossing but also the pedestrians themselves. Utilizing the position of the pedestrian on the crosswalk and the location of the car with respect to the crosswalk. The vehicle would be able to adjust its stopping speed with respect to the pedestrian's location, offering an almost gradient-esque slowdown. This would mean that if a pedestrian is on one side of the crosswalk and is beginning to walk on it, as the vehicle approaches it would decelerate at a variable rate with relation to where the pedestrian is on the crosswalk, so as the pedestrian approaches the center of the crosswalk, the vehicle would reach a near stop, and as they cross towards the other side the vehicle would begin to accelerate again. This would prevent the need for unnecessary stopping which could lead to traffic congestion. This method provided a high amount of development time though, and proved to be very difficult to implement.

An easier method would be to have the vehicle stop entirely whenever it sees a crosswalk. For this it would have to know what a crosswalk looks like which can be achieved by running a learning model on a dataset of crosswalks. This method removes any recognition of a pedestrian, which while stopping the vehicle does not tell the vehicle how long to wait between stop and go. Thus this is not a suitable model in most cases.

A final method that the vehicle could implement is reading the pedestrian lights which are located on the side of sidewalks. When this light is red the vehicle will come to a stop like it would at a normal stoplight, and when it sees the pedestrian crossing light it would then go. This proves to be not just a suitable model but an optimal model in the time given, it is for this reason that we chose this model going forward.

<u>Approach:</u>

Utilizing the model stated at the end of the previous section we set out to complete the task in a number of different steps. A screen capture software was used to record a video of the robot moving throughout the virtual environment, screen captures were then taken, and Roboflow was utilized to annotate a number of these captures from the virtual world. These annotated captures were then passed through a YOLOV5 training algorithm to obtain a model that was somewhat precise in its acknowledgement of pedestrian spaces (see below figure). With a model now trained on crosswalks one ROS node was created to detect crosswalks, while another was created to publish a start and stop command to the vehicle, and a final one was created to control the velocity of the robot based on the signals it receives from the prior two nodes. These elements when added together provided a working model for pedestrian crosswalk recognition.



```
"""
Returns True if detects crosswalk
"""
def crosswalk_cb(self, msg):
    self.detect_crosswalk = msg.data
    self.process_cb()

"""
Returns True if detects stop signal
"""
def signal_cb(self, msg):
    self.detect_signal = msg.data
    self.process_cb()

"""
Drives the robot
"""
def process_cb(self):
    twist_msg = Twist()
    twist_msg.linear.x = REGULAR_SPEED # 1.0

    if self.detect_crosswalk:
        if self.detect_signal:
            twist_msg.linear.x = 0.0 # stop
        else:
            twist_msg.linear.x = SLOW_SPEED # 0.3

    self.publisher_.publish(twist_msg)
```

Figure 2. Robot Control Logic

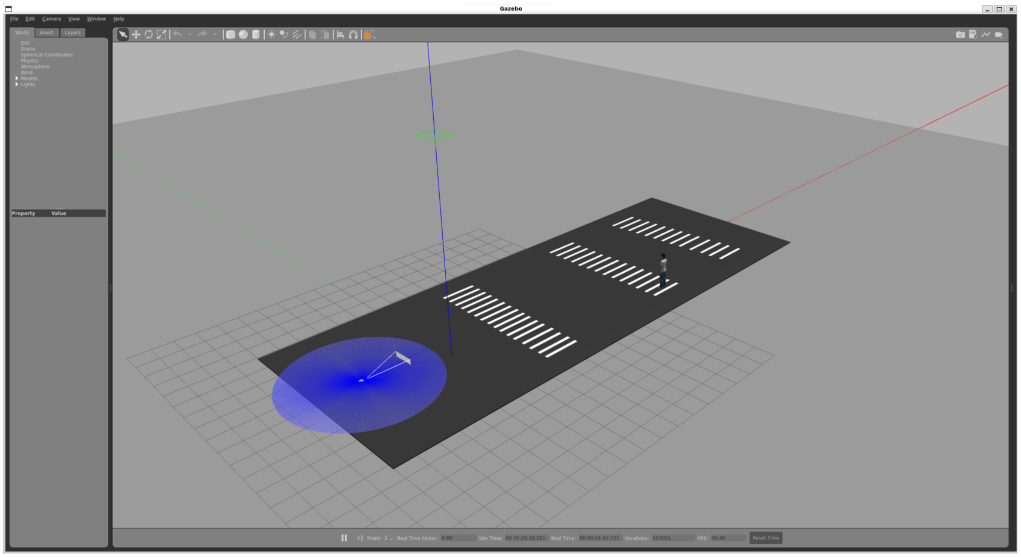


Figure 3. Virtual Environment

Results and Limitations:

The creation of this model provided a working crosswalk recognition software that could prove crucial in the identification and thereby safety of pedestrians when in the presence of autonomous vehicles. Our model was able to correctly identify with high confidence what a crosswalk would look like, as well as, utilize a changing light color to know when to stop. These two elements together allow the vehicle to know not just where to stop on the road, but when to stop on the road.

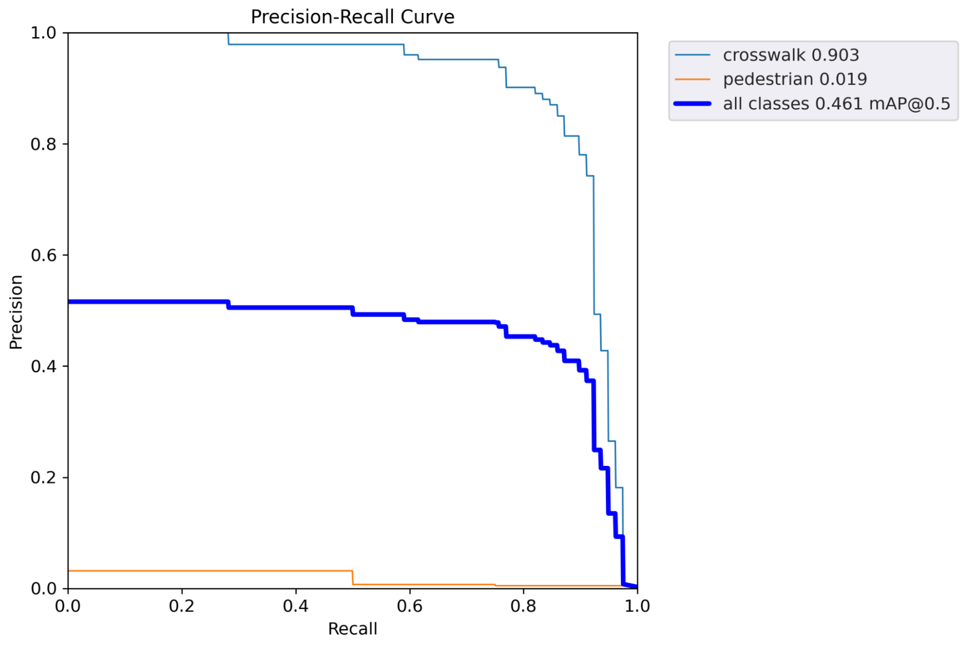


Figure 4. Precision-Recall Curve of Virtual Environment

A question can be raised here about why our training proved to be inefficient at detecting pedestrians. This is entirely based around the fact that there was simply a lack of photos with a pedestrian in them, thus the training model lacked a good base of pedestrian images and could not say with confidence what a pedestrian was. There also exists a ton of models for detecting pedestrians, where we focused more on detecting the crosswalks.

While this algorithm is functioning and a viable option for pedestrian safety it does come with a number of setbacks. For one the vehicle does not actually acknowledge the pedestrian on the crosswalk, thus if a pedestrian were to say be crossing at the wrong time (i.e. J-walking) the vehicle would not stop at the crosswalk, just slowdown. This would have to be compensated for by the vehicles other on board object detection algorithms that it would have already been utilizing to identify other vehicles on the road and obstacles that may have fallen into the road.

The algorithm, while slowing down when coming to a crossing,will completely stop when a light is detected regardless of if there is a pedestrian present at the crossing; this proves less than desirable in terms of traffic flow, since the vehicle stopping at a non-required stop could lead to some level of bottleneck. The reasoning behind this as stated above is that many of the image libraries we found, while having crosswalks as part of them, lacked images that had pedestrians crossing them. This meant that when trying to build a learning model for the robot the recall of pedestrians was incredibly low.

There are also concerns on how the algorithm would react given no light but a pedestrian crossing. Take for example the crossing outside of the engineering building on campus, while technically a guarded crosswalk, it lacks any form of light urging vehicles to stop. In our algorithm this would lead to our algorithm detecting the crosswalk, slowing down, but never getting the indication that it may need to fully stop. An idea would be to have it fully stop and wait a set amount of time before going, but this idea falls apart when you realize that a pedestrian could start crossing at any time, and thus starting to drive again after a set time would probably lead to even more pedestrian related issues. The real solution here would be to train a second model to identify pedestrians so that the vehicle may stop when one is seen. Ideally you want a vehicle that possesses predictable behavior for the pedestrian, since pedestrians rarely, if ever, possess entirely predictable behaviors for the vehicle.

Conclusion:

In a world where the addition of autonomous vehicles can prove to have some safety concerns, it is increasingly important to quell fears over the safety of pedestrians when crossing the street. All agencies require a high level of guaranteed safety within a couple hundred tests to even qualify as road legal. To accomplish this an entirely separate algorithm is necessary to implement a start-stop method at pedestrian crossings, in hopes to ensure safety.

In conclusion, while our model does contain a number of faults and misses some edge case scenarios (such as the guarded crosswalk without a light). With highlights such as a high statistical recall of pedestrian crossings, it proves to be a good skeletal model for a functioning pedestrian safety algorithm that can be built upon as a viable option for autonomous vehicle safety.