Team 2 Project Report

Nick Filerman & Stephen Ng

This project implements a solution to support the goals of an automated bus designed to operate at SAE level 4 autonomy. We assume the main SAE level 4 automation tasks are already implemented by the bus and that the system uses the ROS environment for communication, as is the case with MSU's autonomous bus setup. An autonomous bus can have a variety of benefits over traditional ones, such as the ability to operate consistently for longer without needing breaks or driver changes, and consistently increased fuel efficiency across a fleet by taking individual driver characteristics out of the equation, allowing for calculated and optimized acceleration and braking profiles. The main risk of an autonomous solution is if the system was not implemented robustly enough, for example if there are edge cases the system is unaware of that could potentially causing harm to people.

Our project aims to further optimize the bus automation by providing an indication to the bus stopping function of whether it needs to stop at the next bus stop or not. The current autonomous bus configuration is designed to stop at all predefined bus stops, even if nobody needs to get off and nobody is waiting at the bus stop. By giving the bus the ability to skip stops, it will increase the time efficiency of the bus system, as well as reducing the number of times the bus must brake and accelerate, also further improving the fuel efficiency of the bus. Additionally, by lowering the number of times the bus must come to a complete stop, it can help the overall traffic flow in the area.

At a high level, the system we are introducing checks if anyone is at the bus stop, then combines this input with information regarding whether a manual override inside the bus has been triggered (e.g. someone indicating they want to get off). That information is published as a Boolean value, which the bus would take as an input to its bus stopping function. When the bus is approaching a bus stop, it will check the current status of the Boolean value and determine whether to stop or not. The only real risk introduced by our system is if the camera and machine learning model do not see people at the bus stop who are in fact there. In that case, the bus will not stop, which is not ideal, but not a safety risk. This risk is also mitigated by the fact that you still have a manual override for the stop. We are using the front-right camera of the bus, as that gives the view of the side of the bus that the bus stop will be on, as well as gives a view ahead of the bus to the location where the bus stop will be soon.

Our system subscribes to an image, determines if people are spotted within that image, and then if some logical conditions have been met, sets a Boolean as to whether the bus should stop. The system also includes a manual override function which can be used as an input for people on the bus to signal that they want to get off at the next stop. The benefits of the system, as described previously, are that this will reduce the number of stops the bus has to make during a normal trip. A logical flowchart showing our system and describing our code flow can be seen below in Figure 1. The machine learning model used in our system to detect people utilizes the OpenCV library along with the YOLOv4 trained weights, providing high accuracy. To determine if people are close enough to the bus, we original planned to calculate distance, but then realized that taking a minimum bounding box size would work just as well at determining rough distances, and we did not need higher accuracy since people could be at a variety of distances near a bus stop anyway. The system publishes the Boolean to the '/bus_stopping' ROS topic, and that published Boolean is what the bus would use in order to determine whether to stop as it approaches a bus stop.

To test our system, our python module takes the published Boolean value as text well as bounding boxes around the people it is detecting and overlays those onto the video that comes from the camera. In testing our system, it performed pretty much as expected. A few objects were wrongfully detected as people, such as a large person-shaped rock, but in that scenario, if there were a bus stop nearby, we would just see the bus stopping when it is not needed to. That isn't necessarily a problem, as it is the current behavior of the bus without our system. We have yet to see the opposite occur, where the bus is not detecting people who are there; the object detection seems very good at identifying actual people.

Overall, we applied our knowledge of object detection, as well as understanding the ROS environment, to be able to manipulate it to our need. We learned that given a larger codebase, such as the general object detection framework from a previous lab, we may be able to cut it out to fit exactly what we need if we understand the code well. In this case, we took the object detection program and cut it down to only detect people, significantly reducing the amount of code needed. As is generally taught, if you have already coded something once, no need to code it again.

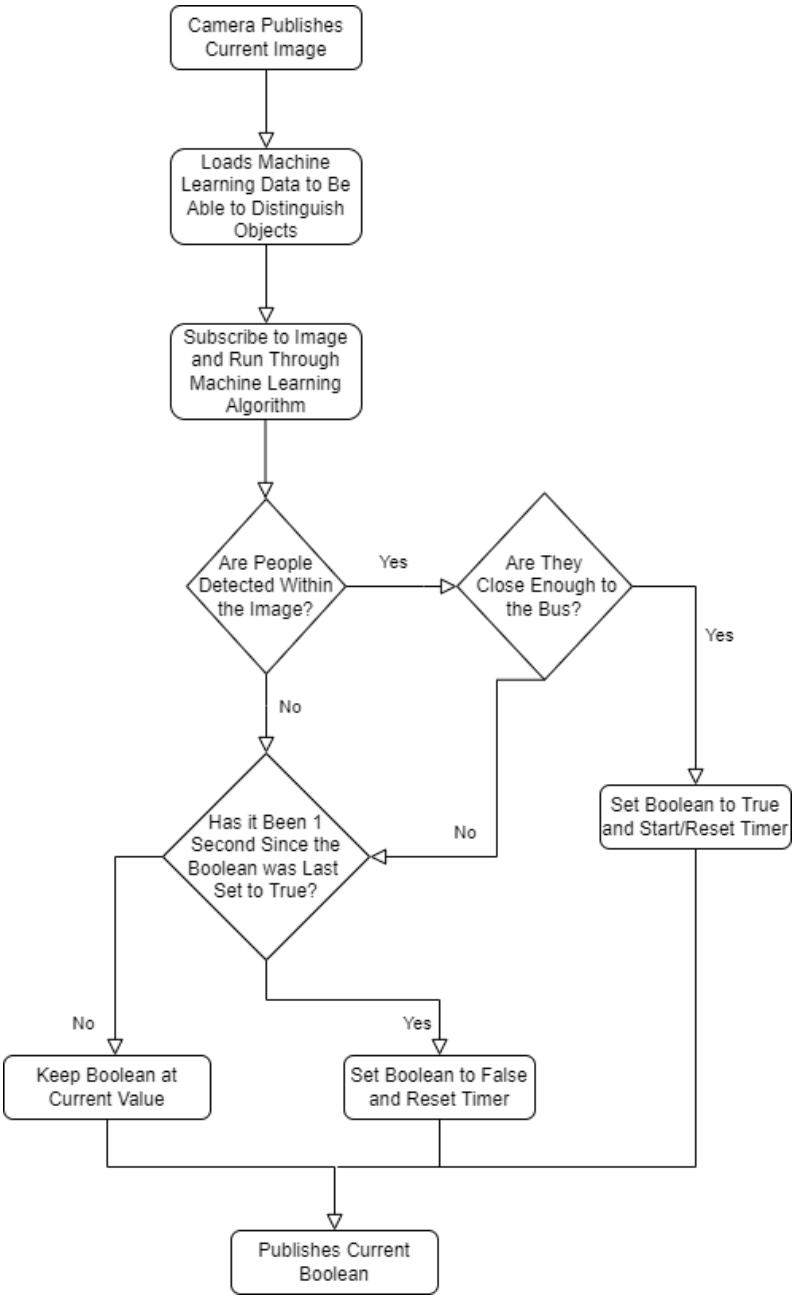

Figure 1. Bus stop detection logic

Concluding this project, our system works well as expected. The next step would be to integrate it into an autonomous bus instead of just viewing the output of our simulation. In future work, we could use the precise position of the people rather than their size to determine if they are close enough to the bus stop, although our current strategy works well. We also could include the use of more cameras to make it even more obvious if people are at the bus stop, and we could test with more bus data to determine the ideal time delay before resetting the Boolean to false, and the ideal minimum bounding box size for a person to be considered close enough to the camera for detection at a bus stop.

Link to video of project running: https://mediaspace.msu.edu/media/t/1_dnw07eds