**Automated Farming and the Complexities of the Unpredictable**

**Written by: Hunter Mast**

As of recent years, automated farming has grown more and more popular among the farming industry. According to the second annual Specialty Crop Automation Report, it is stated that roughly 70% of surveyed farmers claimed investments in automation in 2022. Many factors correlate to this rise in usage such as reducing labor costs, reducing time spent in the field, assisting in farmhand shortages, etc. Automated farming has reached many of the different branches including milking cattle, harvesting, planting, surveying, etc. This led to my thoughts of the complexity that comes from systems designed with dangerous machinery that must somehow assure safety in the field. My goal was to attempt to simulate a system like this to view it’s design and mechanics that must be involved to assure safety.

The first step to designing this system was to decide its structure. There are many ways to program this, but a basic state machine seems to be the simplest. It was created in Python by the way of state machine. Of course, I do not have the resources to create or test a real automated machine. The second best option was to try and create my own field. The two-dimensional field was created with the goal of first creating a randomized two-dimensional view of a field that we could use to simulate a vehicle moving through. When designing, I also wanted to have some options. If an automated vehicle is moving through a field, why is it there? It could have many different tasks as of harvesting grown crops, planting seeds in an empty field, surveying the soil and growth, etc. A lot of this can become very complicated, so I decided to have the field exist in two states: “Empty” and “Grown”. If the field is “Empty”, we can maneuver through the field and plant seeds through every tile we drive through, creating a new “Planted” tile. Same can be with the “Grown” state where we turn “Grown” tile into a “Harvested” tile.

Recognizing the current state of each tile and continuing to move the vehicle, changing the previous state should be a great baseline for a system like this. Yet, it still is not realistic. This would assume that the field is completely empty. I was raised on a farm and helped my father in the fields all my life. Not once have I ever encountered a field that never has any objects or obstacles. I wanted to see if I could create an object detection system that could avoid stationary objects. Again, this is not fully realistic. Many of the objects in a field may just be rocks or sticks blocking our path, but many objects also happened to be wildlife. Complexity rises when we introduce a random variable into our system. I decided to imagine all objects are stationary for this reason. With object avoidance, we also want to make sure that we stay in the lines of the field. Edge detection can be very complex also as some fields have sections that are too small to fit the vehicle, but also weird and unusual shapes. I went with just a basic square that could be sized up and down, depending on the need.

The final part of the field that was designed was the random finished sections. Sometimes, there may be sections of the field that we have already finished working in. When going through the field, it would be unwise to continue to have the blades running while going through an already harvested space. I wanted to try and simulate this with these finished spaces to show a state change into a safer drive state. This would also be beneficial if moving obstacles are implemented. If an object happens to get under the vehicle, it would be safer if the blades or machinery part of the vehicle is not running. Together, this prints out a randomized field with multiple objects to avoid and sections of finished tiles for an overall effective field environment A black and red line

Description automatically generatedsimulation.

Figure 1.1

Now that we have a fully realized two-dimensional field simulated, we can now start to simulate the states our state machine will switch between. The main states I decided on were “idle”, “drive”, “harvest”, “plant”, “obstacleDetection”, “edgeDetection”, and “safety”. There were some other states I attempted to include also, but was shown how complex including systems like this could be.

A drawing of a line with a line and a cross

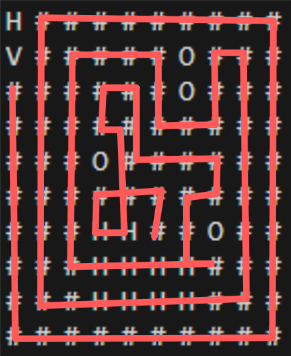
Description automatically generated with medium confidence“setSpeedUp”, “setSpeedDown”, “turnLeft”, and “turnRight” were all added initially, but were removed for one reason or another I will cover later along with “obstacleDectection” and “edgeDetection”. For the main states we included, our system would be started in the “idle” position. This is to prevent accidental movement. It will not move or perform any action until the vehicle receives the “start” command. Then, it will be set into “drive”. This state is where movement is performed and how the vehicle will get around the field. “harvest” and “plant” are similar, but these actions change the field’s states so that the “Grown” tiles become “Harvested” tiles and the “Empty” tiles become “Planted” tiles. “safety” is our final main state we have. This would be our error state. If any unexpected outcome occurs such as a mechanical error or obstacles are hit, then the vehicle stops and only allow manual commands.

Figure 1.2

The commands that I removed were “setSpeedUp”, “setSpeedDown”, “turnLeft”, and “turnRight”. “obstacleDectection” detects any stationary obstacle in our way and will attempt to continue around it using the “turnLeft” and “turnRight” states to maneuver our vehicle past. This will become very complex though as we are constrained between the edges of the field. If we have a field like Figure 1.1, this is how obstacles are normally handled. We attempt to turn left to avoid the obstacle and continue around it.

Yet, what if we add an edge on the other side of the obstacle like in Figure 1.2? If we try to go around, we will hit the edge. We have to adjust ourselves and turn left to avoid the edge and continue.

A computer screen shot of a maze

Description automatically generatedA drawing of a diagram

Description automatically generatedA diagram of a graph

Description automatically generatedFigure 1.3 shows another possibility. What if there is an edge 1 away from the obstacle?

Figure 2.1

Figure .4

Figure 1.3

Figure 1.4 shows what if there is an edge, but against the side? All of these are outcomes we have to consider with our vehicle. We have to be constantly adjusting for all of these possibilities. Still, this does not even cover the surface of moving obstacles with an undeterminable factor to them. “obstacleDetection” and “edgeDetection” work together in tandem to assist our movement throughout the field. “edgeDectection” is a more simplified version of object detection though. It checks more so the bounds of our field, rather then continuing past an unencounterable position. We also need to consider tiles we have harvested or planted previously. We do not want to continue on the same path, so we need to replace our bounds and when we reach a state we already harvested or planted, we turn and continue down a space one row closer to the middle. Originally, I wanted to continue up and down the rows. Yet, I realized that it may be more efficient if we continue in a spiral coil to move through the field. Figure 2.1 shows the difference this could bring to us. It allows us to avoid having to continue through positions we have maneuvered through before and continue until we finish in the middle. A concept that I believe could be added to this is a Markov Decision Process (MDP). Using a MDP, we could possibly go optimize further the most optimal route to reach every “Grown” or “Empty” field state, while minimizing repeating positions.

“setSpeedUp” and “setSpeedDown” were both concepts I originally experimented a bit with. In a normal system, it would be necessary to include this for how fast the vehicle moves throughout a field. The state machine though does not have this full capability though. Since this is a two-dimensional graph, we are either in the next state or not. There is no need to slow or to pick up speed through our field. A way around this I thought of was including time. Time in any form of cyber-physical systems instantly becomes much more complex. With ours, I wanted to print the field and move us to the next state depending on the speed. If we move faster, the vehicle will move to the next position faster and vice versa. My issue with this though is that it does not have the same effect as real-life speed change. Speed change is a gradual A tractor with red wheels

Description automatically generatedchange that happens until you reach desired speed. With my two-dimensional representation, we cannot show that without creating an entire visualizer to show the in between movements. Using floats could help us with a theoretical position we would be, but we cannot just print that as we would either be shown in the previous state or the next one. Since our goal is to keep complexity down at the moment and understand move of general movement and safety of the vehicle, I removed this feature.

Figure .1

Similar to speed, there were some other concepts I considered involving, but removed to keep complexity to a minimum. As I had mentioned before, time would complicate systems along with speed. Soil is one of the largest factors in agriculture and crop growth. I had not worked on anything regarding soil, but factoring in how the soil is when planting and harvesting to get an idea of if these crops happen to be ready would be very beneficial and add a lot of depth. With soil, weather conditions would be a factor that would be important for such a system. Knowing when it is best to pause and wait for better weather or when the best time could be for harvesting or planting. Out of all the different factors that could be added though, machinery size and different forms of machinery would be the next addition I would make. I had a focus on a tractor-like vehicle . I represented this as a single tile on the two-dimensional grid. Yet, if we had a trailer, we would need to factor how it would turn and the tile behind us. Combines are another that could be three tiles wide, for example. Something not factored in is backing up. I always have my vehicle turn or move forward. Though, there are scenarios that reversing my solve. Larger machinery adds to this and creates additional scenarios that need to be considered. There are many directions to evolve this project into a fully fleshed out automated farming system, but this will raise the complexity exponentially.

While creating this project, I had done some research on similar automated systems to view how they function. AGCO and Apex.AI are some of these top of the line businesses researching this field today. Last year, Apex.AI was helping AGCO introduce the Apex.OS development kit into AGCO’s Fendt Xaver vehicle. Figure 3.1 shows us what these vehicles can look like. The Fendt Xaver seems to be somewhat small compared to a normal tractor, but this software ensures safety and speed. Using this example, we can see how a system like mine could be used. The complexity of avoiding obstacles and ensuring that all users and the environment are safe, while keeping it’s liveliness.

The Fendt Xaver is a marvel to see when thinking about the world of agriculture. As we move more towards a cyber-physical world, we must be sure that we can rely on the technology we use. My project showed how safety happens to be ensured though multiple cases and scenarios at even a elementary level. Many other factors need to be well thought out and considered for an actual system such as time, speed, and the machinery itself. In the end, my project shows us a glimpse into what farming and agriculture is moving ever so closer towards.

**Work Cited**:

Marston, Jennifer. “70% of Specialty Crop Growers Now Investing in Farm Robots, Says New Western Growers Report.” *AgFunderNews*, 8 May 2023, agfundernews.com/70-of-specialty-crop-growers-now-investing-in-farm-robots-says-new-western-growers-report#:~:text=Labor%20costs%20and%20availability%20issues,invested%20in%20automation%20in%202022.

Staff, Robotics 24/7. “AGCO to Use Apex.OS SDK for Fendt Xaver Self Driving Agricultural Vehicle.” *Robotics 24/7*, 9 June 2022, www.robotics247.com/article/agco\_uses\_apexos\_sdk\_fendt\_xaver\_self\_driving\_agricultural\_vehicle.