Product: TransportED
Team: AutomatEd

Abstract

TransportED is a service which helps you automate your warehouse and aims to knockout the biggest barrier to entry – the massive infrastructure investment. Instead of a cost prohibitive capital expenditure we aim to provide a swarm of autonomous robots, powerful software and our engineering expertise as a cost-effective and quick to deploy service that integrates seamlessly in a variety of warehouse environments.

For this demo, we studied the feasibility and practicality of our robot design. This includes market research for lift types, wheels, motors, and the robotic arm. We also attempted integrating scissor lifts in our simulation. Furthermore, we added multi agent and routing functionality to our system of robots. We continued integrating our subsystems together, linking the warehouse generator with our web server and the database. This is accessible through a easy-to-use user interface, currently under development.

1. Project plan update

- Robot Design and Simulation
  - Market Research (Achieved)
  - Scissor Lift Integration (Partly achieved)
  - Fully Realistic Physics Simulation (Partly achieved)
  - Cost of manufacturing (Not achieved)
- Swarm Robot Navigation
  - Routing algorithm (Achieved)
  - Collision detection (Achieved)
  - Multi-agent algorithm (Achieved)
- Database, Environment Simulation and Testing
  - Final database structure (Achieved)
  - Map generator interface (Partly achieved)
  - UI design (Partly achieved)

1.1. Deviations from original plan

We achieved a lot of the tasks mentioned in the original group plan, and second demo's revised group plan, as seen above. However, each task was faced with a lot of challenges, leading to a compromised solution. They were essential to ensure a working product with all the essential components, and were made keeping scalability in mind.

We faced a lot of challenges in accurately integrating the scissor lift mechanism with the rest of our simulation and decided to push this goal back to the final demo. This will include research on the exact mechanics of the scissor lift.

Although the plan for this demo was to sharpen up the UI of the Warehouse Generator/Mapper, We decided to prioritize the UI design related to functionality rather than aesthetics. We will improve the aesthetics by next demo.

1.2. Group organisation

Modelling & Simulation team Fredrik and Reece worked on the multi-agent system, researched different methods for multi-agent path-finding and consulted with an expert. Following this, Fredrik implemented the multi-agent system described later on along with the routing algorithm used by the server to send each robot its specific task. Reece implemented the collision prevention mechanism. He also refined the warehouse generator. Divy worked on refining the line following mechanism.

Mike and Divy researched the feasibility and practicality of the wheels, the motors, the robotic arm and the RFID tags. Mike continued the work by accurately modelling important parts of robot in CAD. Rohan designed the scissor lift mechanism and worked on integrating it with the robot. He also added a drop-off functionality for the robot to place boxes on the shelves.

Jiaqing was assigned the routing algorithm for our robot. This task proved too challenging for him so Fredrik completed it instead. Jiaqing then wrote, structured and managed everyone in writing the general part of this report.

Software team Ufuk continued to work on the backend and user interface for the warehouse management system. Ryan continued to work on the warehouse generator's interface, adding new functionality. Together, Ufuk and Ryan, integrated the warehouse generator into the web server. Dave worked on creating the contents of the website, particularly relating to market research. He also created a document on user interface design guidelines.

Arrangements We stuck to the meeting structure we described in the earlier reports.

Moreover, as each of us worked on our separate git branches, we would review the merge requests thoroughly ensuring integrated and scalable code. This allowed people to work in parallel without causing confusions, delays or code failures.

Most tasks we worked on are from our original project plan. After each milestone, the team leaders came up with addi-

tional tasks to ensure everybody has something to work on. Additionally, we debriefed on the tasks done, understanding the amount of time spent by each member, and difficulties faced by them. We used this information to re-distribute the tasks more equally, and ensured the tasks aligned with everyone's strengths. We found out that during the last weeks, Fredrik had to do too much of the development, while Dave and Jiaqing would be able to take on additional challenges. To adapt to this, Fredrik will step a bit down on the development side, while Jiaqing and Dave will take on more responsibilities when it comes to market research and writing. All others felt that their tasks were a good amount of work and are happy to continue at the current pace.

1.3. Current budget spent

At this point in time we have not incurred any costs. So far, we have spent 15 minutes of our technician time with Gary to talk about the hardware feasibility of our robot.

1.4. Modifications to future milestones

We will use our map generator to simulate various warehouse environments, demonstrating robustness of our robot. We will continue improving our UI, making it more user friendly and carrying out qualitative surveys. Based on these various tests we will make optimizations to our system. We will also continue with market research, and cost analysis of our robot when built, ensuring that TransportED is the efficient and affordable warehouse automation service small-medium sized warehouse owners go to. We will present this in a neatly setup website, finish a user guide for our system, and prepare presentations for the industry day.

2. Technical details

2.1. Hardware

2.1.1. Robot design

Platform Design: For the arm to move from one side of the platform to the other a linear actuator will be required. As it is to be mounted below the platform it has been designed in a low-profile manner, to allow the platform to be as closer to the ground. The arm is attached to a sliding platform on the driving thread, which crosses the platform layer through 2 narrow long slits in the platform. These slits allow for the movement of the platform while reducing the chance of boxes getting caught in the gap. As shown in figure 2.

Based on a MariTerm AB study, work on the static friction factor for different material combinations (COF) it was determined that the maximum force required to drag a box onto the platform will be 50% of the weight of the box, assuming a cardboard box on a wooden shelf. Changing the box to wood or shrink film and changing the shelf to a steel or aluminium plate, will reduce the force needed.

Choice of wheels: We chose mecanum wheels (mec) over standard wheels as it allows omni-directional movement for our robot. This would allow our robot to move in very tight spaces, and also allow warehouse owners to have more shelves in a smaller area increasing warehouse storage capacity. Moreover, moving laterally allows us greater control over grabbing of the parcels, which is impossible with standard wheels. One disadvantage of mecanum wheels is the small contact patch on which the weight of the robot acts through, as up to 40kg could be loaded over 2 wheels in the worst case. Similarly, being aware of the various disadvantages of mecanum wheels like weight capacity, and durability we found that their advantages provide a bigger benefit, and hence decided to continue using them. These disadvantages can be reduced by using a roller core of a stronger material, such as steel or aluminium alloy rather than a plastic.

As for driving the wheels it was decided to use simple brushed DC motors, using a PWM (PWM) controlled supply to control the speed and an optical encoder to monitor the speed of each of the motors in a closed loop control system. This decision was made as they are relatively inexpensive, compared to brushless and stepper motors, and would require a simpler control system.

2.1.2. Scissor Lift

We decided to use scissor lifts for our platform. We took advice from an expert to make this decision and ensured its feasibility through market research, and settled on it.

However, we faced a lot of challenges implementing these realistically in WeBots. Hence, we are forced to use an unrealistic representation of it meant as visual aid for the simulation. Note however, we still take into account all the other physics which involve the weight, shifting center of gravity, inertia, friction of all other components of the robot.

We observed that the platform, which is connected to the final scissor, becomes disengaged from the rest of the robot for a short while after the lift is raised or lowered. This resulted in very odd occurrences where the platform would be left hanging in the air behind the robot.

The mechanics of the scissors have also changed to make it look more realistic: the bottom most scissors are able to slide along the base, which gives us the ability to achieve the symmetric pattern found in real scissor lifts.

2.1.3. Parcel Drop-off

We implemented the ability to place a parcel on any given shelf. This means our system is now capable of restocking the warehouse with minimal effort.

2.2. Software

2.2.1. Routing Algorithm

As discussed in our previous reports, our warehouse is represented as a set of nodes with connections between adjacent nodes. Shelves are attached to these nodes and can be reached from them. To ensure efficient execution

of tasks, we need to minimize the time spent to go from one location to another. As our environment is static, we decided to pre-compute the shortest paths between any two nodes in our warehouse. To do so, we decided to run Dijkstra (DIJ) from every node in the warehouse. We considered other algorithms like Floyd-Warshall (FLO), as its time complexity is independent of the number of edges. However, our graph has only a small number of connections to other nodes (<6), which makes Dijkstra with a time complexity of $O(V(V+E)log(V) \rightarrow O(V^2 log(V))$ the more efficient option. After the pre-computation step, route-finding can be completed in constant time by simply looking up the shortest path, reducing the load on the central server which should further reduce the costs.

After computing the path, we provide atomic tasks to the robot. These tasks consist of following the line until a specific node is reached or turning until a certain number of lines are crossed (to reach the next node). To determine the number of lines to cross, we look at the specific configuration of connections of the current node.After adjusting the robot's heading direction, giving the command to reach the next node is trivial, as you simply have to put in the node ID of the next node in the path.

2.2.2. Multi-Agent

We initially wanted to implement cooperative pathfinding as demonstrated in (CP), which is based on the idea of doing A* through time, where at each timestep the robots' would reserve the position at that time which should avoid collisions. We decided against this, as one of the assumptions of it is a discrete timespace, and handling this correctly in non-discrete environments like the real world would require an implementation of a precedence graph (MAP). The expected gains from this approach over others for our target of small to medium warehouses would be quite small and not worth the dramatic increase of time and complexity this would introduce.

Nonetheless, ideas were still taken from the (MAP) seminar with the solution we are settling on being a modification of their "highway" system, which uses highways as a suggestion for the pathfinder. We decided to change this into a full one way system, where each line must be designated a direction and bots can only move in that direction (Shown in figure 1). This reduces deadlocks, pileups and avoids situations where two robots go into opposite directions using the same path.

By creating a one-way environment, safety is also increased for anyone in the operating environment as the bots will always move in a consistent and predictable way. A distance sensor was added to the robot to detect objects directly in front of it. Robots are also able to detect the next closest robot along the line its trying to merge onto and get its distance from it to ensure the other robot is at a sufficient distance.

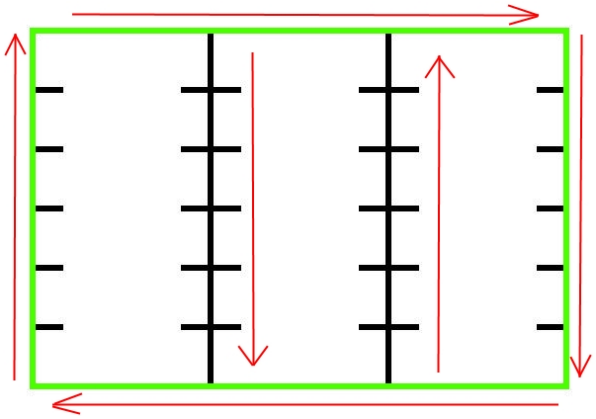

Figure 1. Simple representation of a warehouse with highways marked in green and red arrows denoting the direction of a path

2.2.3. Navigation

New features to the robot navigation system were added to facilitate the multi-agent aspect of our project. Firstly, robots are now confined to a one-way system, this was picked as it is an effective way to cut congestion in the system while also increasing the robustness of the solution, which is highly important due to the general solution aspect of the system. Also, highways can be added to the map which are just paths the robot can take with a higher priority when compared to other paths.

With these changes, there are now two main collision cases to consider: merging onto or crossing another line and dealing with an obstacle in front of the robot (i.e., robots, staff or boxes). To deal with the first case, whenever we are approaching a junction (defined as a node with more than one incoming connection), we are starting a search using Dijkstra for the closest robot that can possibly move onto that junction. We only search for robots that are on a higher priority lane. (e.g. a highway) If we find a robot that is closer than a certain threshold, we block forward movement until that robot is far enough away. This gives a natural solution to the merge-and-cross problem, as long as all incoming lanes are of different priorities. Namely, the robot on the highest priority lane will not find any close robots and therefore cross the junction first, after which the robot on the second-highest priority lane crosses the junction, which continues until all robots have crossed the junction.

We deal with the second case by attaching a distance sensor to the front of the robot which blocks forward movement if an obstacle is too close defined by a value called "Safety Distance"

2.2.4. Warehouse Mapper

Continuing from the last demo, the networking side of the mapper was successfully implemented. This allows user to visually map the warehouse's layout and load all the details in the database. These include shelves, their IDs and pathing for the robots. The generator is now fully integrated

with the rest of our warehouse management system and can be found under domain-name.com/generator.

Based on demo2's feedback, UI expert Julian Habekost's advice, and a small feedback session with our peer group we identified the following issues: initial grid was unclear. Node connections difficult to identify. It was cumbersome to use and fill up a complex warehouse layout. It was unappealing and difficult to understand. We started tackling each issue, starting with the functionality related ones, and by next demo we will address the UI issues.

In this iteration, grid lines are removed by default and are only visible if a connection exists between nodes. Similarly, blank nodes are removed. We also added a button selector for placing shelves, robots, and paths. A lot of other basic functions like batch deletion and specifying grid dimensions and implementing a legend to identify node types by their colour were also added.

3. Evaluation

3.1. Navigation

3.1.1. Routing Algorithm

We set up test cases where each case includes different starting point, picking point and ending point from other cases for verifying the correctness of our routing algorithm. Then, we calculate total time taken for each case.

Table 1 shows the the successes and time taken for multiple different routing conditions. It was found that the failures occured due to incorrect turning/ RFID tag positions, causing the robot to follow the incorrect path. Hence, as seen in 1 the routing algorithm works well. The node configuration is in the appendix and can be seen in figure 3.

Test No.	Starting Node	Shelf Node	Success	Time(S)
1	47	36	×	N/A
2	7	25	✓	95.39
3	50	38	✓	106.04
4	0	39	×	N/A
5	2	42	✓	29.36

Table 1. Robot routing and moving to shelf and picking up box

3.1.2. Multi-Agent Interaction

Next we tested various conflicts at various different four way junctions, namely, as seen in 2: Test 4 was added when it was found that the tests 1-3 all failed at the same junction. This allowed us to find the issue in RFID tags being detected earlier than desired, and led us to research more about our requirements for RFID emission range.

3.1.3. Pick Up Mechanism

Table 3 shows the results of running three different package pick-ups at the same time on three bots to test the multiagent aspect. Again as with the tests in Table 1 and 2 the bot failures were caused by issues of overturning at a certain junction. In the test with two failures the bot overturning and going off course ended in a dead-lock situation between it and another bot as it got in an area it never should have. The pick up and drop off location for the robot that succeeded were very close to the starting location, which explains the fast completion time.

Conflict type	Times Tested	Success
One bot merging	5	4
Two bots merging	5	4
Two bots merging, one passing	5	4
The failure junction from previous tests	5	0

Table 2. Junction Conflict Testing

Bots Failed	Bots Succeeded	Time of Final success (s)
1	2	156.57
0	3	168.52
0	3	121.93
2	1	43.02

Table 3. 3 Agent Pickup and drop-off

3.2. Distance From Object When Stopped

Despite most of our tasks being sent via server to the robot, most of our safety mechanisms are implemented locally to avoid latency delays. Below we test the robot's braking mechanism to avoid collisions. We have taken the mass, inertia and friction into account while modelling.

Robot Velocity	Collided?	Average stopping distance
0.2m/s	×	4.4cm
0.35m/s	×	10.7cm
0.5m/s	×	19.2cm
0.65m/s	✓	31.2cm

Table 4. Amount of distance travelled after braking

4. Budget

We plan on researching the potential costs of creating our system when we fully developed the robot design.

5. Video

This is the link to the video.

References

Inclination tests to determine the static friction factor for different material combinations, mariterm ab. URL http://www.mariterm.se/wp-content/uploads/2016/

08/N-208-SE-Friction-tests.pdf. Online; accessed 12 March 2021.

Cooperative pathfinding; david silver-university of alberta. URL https://www.davidsilver.uk/wp-content/uploads/2020/03/coop-path-AIWisdom.pdf.

Dijkstra algorithm. URL https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm. Online; accessed 14 March 2021.

Floyd-warshall algorithm. URL https://en.wikipedia.org/wiki/Floyd%E2%80%93Warshall_algorithm. Online; accessed 14 March 2021.

Multi-agent path-finding seminar; given by sven koenig of usc viterbi. URL https://www.youtube.com/watch?v=p7sUCIB1jqc.

Controlling brushed dc motors using pwm. URL https://www.machinedesign.com/materials/article/21125511/controlling-brushed-dc-motors-using-pwm. Online, accessed 15 March 2021.

Warehouse automation market: Post covid-19 opportunities worth $30b by 2026. URL https://www.thelogisticsiq.com/research/warehouse-automation-market/.

Mecanum wheel. URL https://en.wikipedia.org/wiki/Mecanum_wheel. Online, accessed 15 March 2021.

A. Additional figures/data/experiments/descriptions

Robot Speed	Collided?	stopping criteria	stopping dist
0.2m/s	×	100cm	98.0cm
0.2m/s	×	70cm	64.7cm
0.2m/s	×	50cm	44.8cm
0.2m/s	×	30cm	24.8cm
0.35m/s	×	100cm	88.2cm
0.35m/s	×	70cm	59.0cm
0.35m/s	×	50cm	41.7cm
0.35m/s	×	30cm	18.4cm
0.5m/s	×	100cm	78.6cm
0.5m/s	×	70cm	53.6cm
0.5m/s	×	50cm	28.8cm
0.5m/s	×	30cm	12.1cm
0.65m/s	×	100cm	72.7cm
0.65m/s	×	70cm	40.8cm
0.65m/s	×	50cm	18.9cm
0.65m/s	✓	30cm	NaN

Table 5. Remaining distance after collision detection and full stop

B. Other supplementary materials....

B.1. Market Research

We have increased our focus on market research, as it is an important aspect of our project.

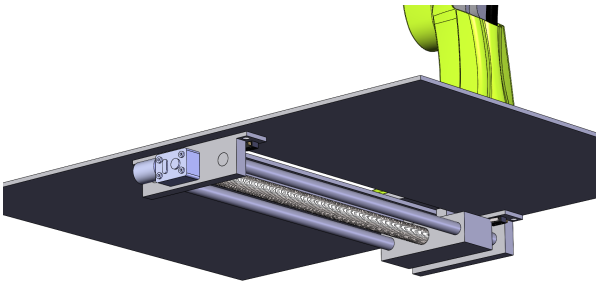

Figure 2. Platform Solution, created in Solidworks

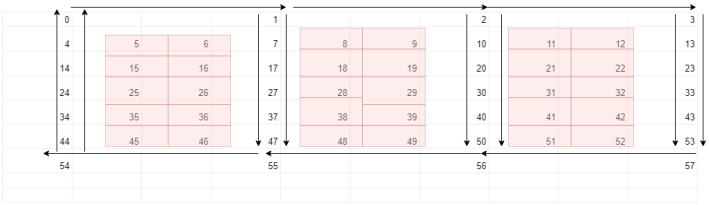

Figure 3. Node layout for the warehouse the tests were performed in.

In 2019, the warehouse automation market was valued at USD 15 billion and is expected to grow 14% compound annual growth rate (CAGR) for the next 7 years, reaching USD 30 billion by 2026. (WAM)

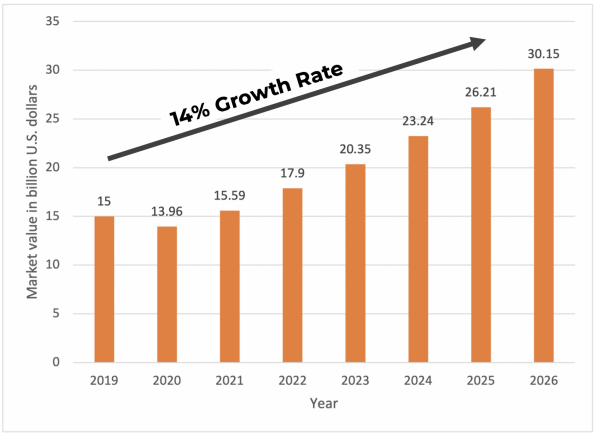

Figure 4. Global market valuation for warehouse automation from 2019-2026

We have set our unique selling points:

1. Rapid deployment with minimal cost as our system fits into existing warehouses.
2. Robot-as-a-Service business model to demolish big barrier to entry for automation by providing subscription-based service.
3. Maximum flexibility and scalability of system.