Minireach: Autonomous Forklifts

Kim Byström, Lovisa Jansson, Anton Johansson, Joar Manhed, David Sandmark, Niklas Stenberg, Pär Sörliden, Gustaf Westerholm

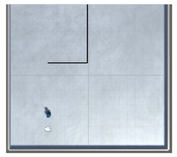
Introduction:

The goal of the project has been to create an environment in Unity3D where an autonomous forklift and its different modules - control and planning - can be simulated. The algorithms for control and route planning have been developed separately in two different modules. Both modules have the capability of communicating with the truck through Toyota's communication protocol called Smartness. The simulation environment is run on a host computer that the modules communicate with. This project was done in collaboration with Toyota Material Handling.

NAVIGATION

Map:

The route planning is using a map that is described by a binary matrix with ones (unreachable) and zeros (reachable) according to figure 1. In the picture, the left side is the Unity environment, while the right side is the matrix that it is represented by. The matrix is thus describing a grid based on the environment, where the grid coarseness is variable. The grid resolution has been set to 10x10 cm according to the Smartness standard. The map is updated once every second through analyzing Collider-blocks in Unity.



```
1 0 0 1 0 0 1
1 0 0 1 0 0 1
1 0 1 1 0 0 1
1 0 0 0 0 0 1
1 0 0 0 0 0 1
1 0 0 0 0 0 1
1 1 1 1 1 1 1
```

Figure 1. Left side is in Unity and the right is a binary map

Route planning:

The route planning is using an A*-algorithm to find the shortest path, which is developed from Dijkstra's method. The algorithm uses arc costs and the euclidean distance between current node and destination to efficiently work its way toward the shortest path in an effective way. The example shown in figure 2 is demonstrating that even if the arc cost is lower for the top right node, the total cost would be higher than going to the diagonal node. It is this knowledge the A* algorithm is using to avoid calculating unnecessary nodes.

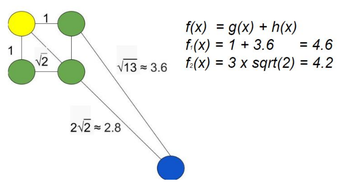


Figure 2. Illustration of the A*-algorithm

Obstacle detection:

Each time a new map is received, the navigation module makes a comparison between the planned route and the new map to see if any obstacles have been detected. If so, the pathfinding algorithm is run once more with the new map as input. The result of this can be seen in figure 3.

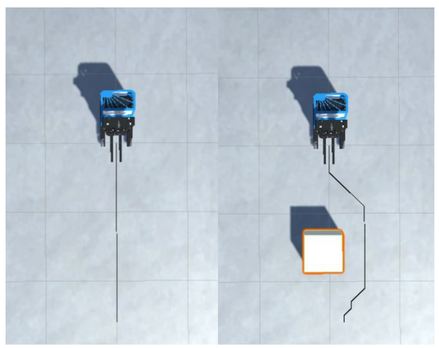


Figure 3. Dynamic pathfinding

Velocity planning:

To make sure that the forklift can follow the trajectory even in sharp corners, appropriate velocities are calculated for each node on the route. In corners, the speed is penalized with a factor K depending on the curve's angle, see figure 4 for the different cases. With the suitable velocity set, the velocity vector is backtracked where the previous elements are scaled with a deceleration factor to achieve desired braking. This can be seen in figure 5.

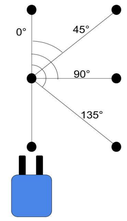


Figure 4. Different curve cases

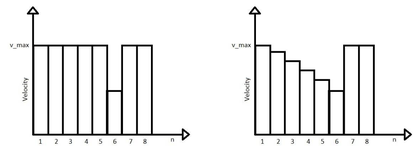


Figure 5. Backtracking of velocity vector

CONTROL

Modelling:

The forklift is modelled as a tricycle using the following equations, with coordinates and angle defined as in figure 6. α is the angle of the steering wheel.

$$\dot{x} = v_s \cos\alpha \cos\theta$$
$$\dot{z} = v_s \cos\alpha \sin\theta$$
$$\dot{\theta} = \frac{v_s \sin\alpha}{l}$$

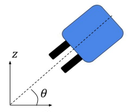


Figure 6. Definition of coordinates and angles

Linearization:

The equations describing the model are nonlinear. If (x,z) is the output, the system can be input-output linearized by differentiating the model equations and setting the control signals to certain values. Then the third order derivatives of the coordinates (x, z) can be controlled with reference signals r_1 and r_2. In summary, we get $\dddot{x} = r_1$, $\dddot{z} = r_2$.

B-splines:

The reference signals r_1 and r_2 are set to be the third order derivative of the desired trajectory. Thus, the controller needs a trajectory with continuous derivatives up to order three. Since the planning module sends discrete points representing a route with desired velocities, a tool for creating a smooth trajectory is needed. For this purpose, B-splines are used. These create a smooth trajectory described by polynomial functions that can be used by the controller.

Feedback:

The feedback is introduced as follows to handle errors in the model and incorrect initialization:

$$\dddot{x} = r_1 + k_a(\ddot{x}_r - \ddot{x}) + k_v(\dot{x}_r - \dot{x}) + k_p(x_r - x)$$
$$\dddot{z} = r_2 + k_a(\ddot{z}_r - \ddot{z}) + k_v(\dot{z}_r - \dot{z}) + k_p(z_r - z)$$

The gains k_a, k_v and k_p are chosen such that $\lambda^3 + k_a\lambda^2 + k_v\lambda + k_p$ has its roots in the left half of the complex plane. The variables marked with subscript r are the reference values and the other values are the states of the forklift. The higher-order derivatives of the states are estimated using model equations. Figure 7 shows the controller structure.

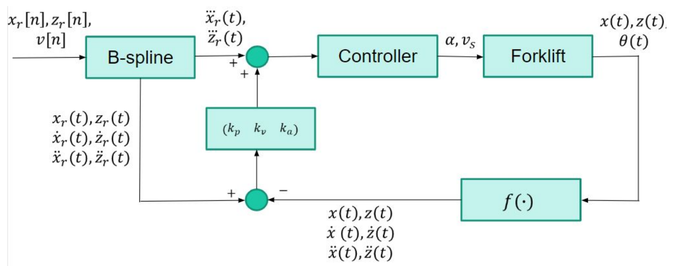


Figure 7. Illustation of controller structure

Alternative control algorithm:

In order to determine the performance of the feedforward controller, a simple PID algorithm was implemented. The PID controller uses the angle error shown in figure 8 below to follow the trajectory.

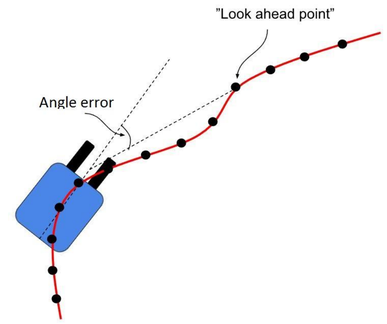


Figure 8. Illustation of angle error

Results:

Some comparisons between the actual trajectory and the desired trajectory are shown in figure 9. This is a case where the controllers are performing relatively well. Do note that the velocity while using the feedforward controller is several times higher than what the angle PID controller is using. In figure 10 only the speed of the feedforward controller is shown.

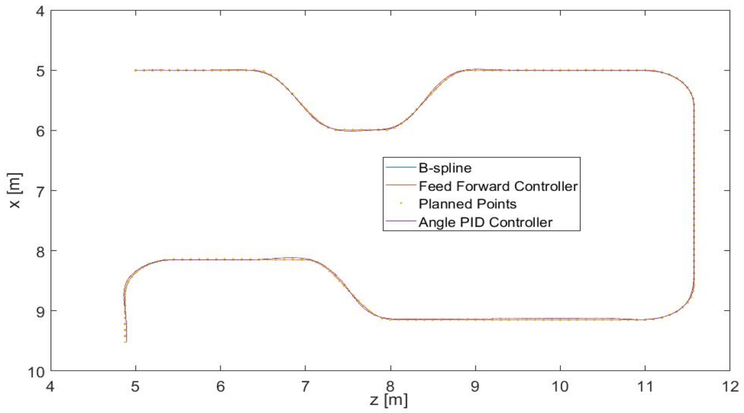


Figure 9. Desired and actual trajectory for different controllers

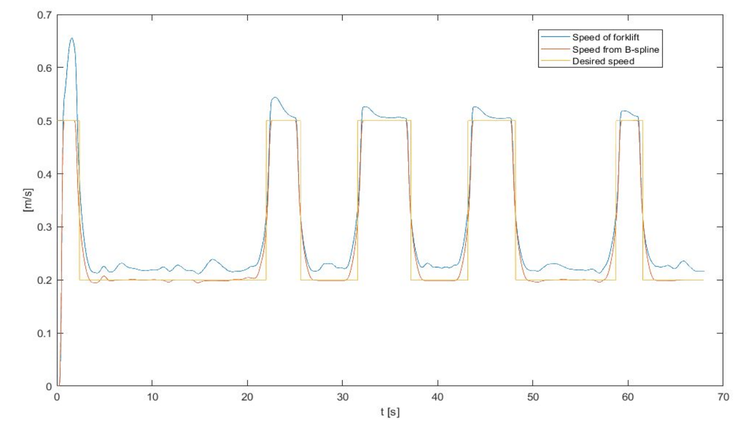


Figure 10. The speed of the feedforward controller

LINKÖPINGS UNIVERSITET
TOYOTA
MATERIAL HANDLING