Tampere University

Tomi Porras

DESIGN AND IMPLEMENTATION OF A HORIZONTAL TRANSPORTATION SIMULATION MODEL

Master's thesis
Faculty of Engineering and Natural Sciences
Examiner: Matti Vilkko
Examiner: Risto Ritala
April 2021

ABSTRACT

Tomi Porras: Design and Implementation of a Horizontal Transportation Simulation Model
Master's thesis
Tampere University
Master's degree programme in automation engineering
April 2021

The large volume of container traffic handled by the world's container terminals requires more and more efficient operation. In this thesis, a simulation model for horizontal transportation (HT) is designed and implemented with MATLAB Simulink. HT operation covers the container handling between terminal quayside, terminal yard, and landside. In this thesis, HT is performed by straddle carriers (SCs), which can pick, ground and stack containers without other cranes. The model produces data regarding per crane productivity in the yard, as well as path-planning information.

The implemented simulation model is composed of the kinematic model of a SC controlled by a horizontal transportation manager (HTM). The modelled SCs can move realistically along the terminal yard and perform the necessary HT operation. The implemented HTM creates a lane system that connects distinct terminal yard areas together, and assigns picking and grounding jobs for the operational SCs. The HTM also handles the path-planning operation with a graph-based path-planning algorithm, to optimize the HT operation.

The kinematics of the SCs are implemented as a discrete time model and the HTM as an event-driven control logic. The model is designed and implemented according to extensive requirement analysis and is verified by the same requirements. The model's realism is validated by simulating a SC in operation and comparing the results to values found in literature. The validation and verification prove that the implemented model can represent HT operation realistically enough. According to simulations the average per crane productivity was approximately 24 moves per hour. When the terminal yard was populated with more operational SCs, the per crane productivity started to diminish and collisions caused by other SCs increased. The implemented model lacked a complete collision avoidance system, and as such the complete effect on productivity caused by increasing number of operational vehicles could not be recorded.

Along with the limitations provided by the lack of collision avoidance, the implemented model also suffers from less-than-optimal job sequencing algorithm. The decrease in productivity caused by increasing number of SCs can be largely attributed to the unevenly distributed jobs among the operating fleet. For the model to be used for extensive HT productivity analysis, the job sequencing algorithm should be changed to a more sophisticated to manage the fleet more optimally. For realistic simulation, the collision avoidance system is required for the implemented model to produce results comparable to real-life systems.

Keywords: HT operation, straddle carrier, modelling, path-planning, simulation

The originality of this thesis has been checked using the Turnitin OriginalityCheck service.

TIIVISTELMÄ

Tomi Porras: Kontinkuljetusjärjestelmän simulaatiomallin suunnittelu ja toteutus
Diplomityö
Tampereen yliopisto
Automaatiotekniikan diplomi-insinöörin tutkinto-ohjelma
Huhtikuu 2021

Konttiliikenteen suuren määrän käsittelemiseksi maailman konttiterminaalit vaativat yhä tehokkaampaa toimintaa. Tässä työssä suunnitellaan ja toteutetaan konttiterminaalin kontinkuljetusjärjestelmän (HT) simulaationmalli MATLAB Simulink -ohjelmistolla. HT tarkoittaa kontinsiirto-operaatioita, jotka tapahtuvat laiturialueen, terminaalipihan ja maaliikennealueen välillä. Tässä työssä HT:n hoitavat konttilukit, jotka pystyvät nostamaan, laskemaan ja pinoamaan kontteja ilman sataman muita nostureita. Simulaationmalli laskee yksittäisten lukkien tuottavuuksia sekä reititysinformaatiota.

Toteutettu simulaatiomalli koostuu konttilukin kinemaattisesta mallista, jota ohjataan kontinkuljetusjärjestelmän hallintaohjelmalla (HTM). Mallinnetut konttilukit liikkuvat realistisesti terminaalipihalla, ja suorittavat niille annettuja nosto-, lasku- ja siirtotehtäviä. Mallinnettu HTM luo terminaalipihalle eri konttialueet yhdistävän kaistajärjestelmän, ja määrittää HT-tehtäviä toiminnassa oleville konttilukeille. Kontinsiirto-operaatioiden optimoimiseksi HTM reitittää konttilukkien reitit graafiin perustuvalla reititysalgoritmilla.

Konttilukin kinematiikka mallinnetaan diskreettiaikaisella mallilla ja HTM tapahtumapohjaisena sekvenssilogiikkana. Malli suunnitellaan ja toteutetaan kattavan vaatimusmäärittelyn perusteella, ja samoja vaatimuksia käytetään mallin varmentamiseen. Mallin realismi vahvistetaan vertaamalla saatuja simulaatiotuloksia konttilukin toiminnasta kirjallisuudesta löydettyihin arvoihin. Tehdyn arvioinnin perusteella malli kuvaa riittävän hyvin realistista kontinkuljetusjärjestelmää. Simulaatioiden perusteella yhden konttilukin keskimääräinen tuottavuus on noin 24 siirtoa tunnissa. Kun terminaalipihalle lisättiin useita toiminnassa olevia konttilukkeja, yhden ajoneuvon tuottavuus kääntyi laskuun, ja lukkien aiheuttamien törmäysten määrä kasvoi. Toteutetusta järjestelmästä puuttui kokonainen törmäysten välttämisjärjestelmä, joten lisääntyvän ajoneuvomäärän todellista vaikutusta terminaalialueen tuottavuuteen ei voitu määrittää.

Puuttuvan törmäyksenestojärjestelmän lisäksi, toteutetun mallin työmäärien jako ei ole optimaalista. Simuloinneista saatu tuottavuuden väheneminen konemäärän kasvaessa voidaan pitkälti pitää epätasaisesti jaettujen työmäärien aiheuttamana. Jotta toteutettua mallia voitaisiin käyttää laajamittaiseen kontinkuljetusjärjestelmän tuottavuuden analysointiin, tulisi työtehtävien jakoon käytettävä algoritmi korvata hienostuneemmalla konttilukkien optimaalisempaa hallintaa varten. Realistista simulointia varten, täysin toimiva törmäyksenestojärjestelmä on välttämätön, jotta mallin tuottamat tulokset olisivat vertailukelpoisia todellisiin järjestelmiin.

Avainsanat: Kontinkuljetusoperaatio, konttilukki, mallinnus, reititys, simulaatio

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck –ohjelmalla.

PREFACE

This thesis was commissioned by Kalmar, which is a part of Cargotec Finland Oy. I would like to thank Hannu Santahuhta for providing the opportunity to work with such an interesting topic in a new field. I would also like to thank Johannes Mansikkala for supervising and sanity checking my work during the whole process. From Tampere University, I would like to thank professors Matti Vilkko and Risto Ritala for their invaluable guidance and expertise, as well as their patience. Special thanks to my wife for supporting me for the entirety of my studies, and especially during the long process of writing this thesis.

In Tampere, on 26 April 2021

Tomi Porras

CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

AGV	Automated Guided Vehicle
ASC	Automated Stacking Crane
BFS	Breadth First Search
CHE	Container Handling Equipment
DES	Discrete Event System
DFS	Depth First Search
DoF	Degree of Freedom
FS	Functional Specification
HT	Horizontal Transportation
HTE	Horizontal Transportation Equipment
HTM	Horizontal Transportation Manager
MAPP	Multi-Agent Path-Planning
OOG	Out Of Gauge
RMG	Rail Mounted Gantry
RTG	Rubber Tyred Gantry
SC	Straddle Carrier
ShC	Shuttle Carrier
SPT	Shortest Path Tree
STS	Ship-To-Shore
TEU	Twenty-foot Equivalent Unit
TTU	Tractor-Trailer Unit
UDP	User Datagram Protocol
UR	User Requirement
mph	Moves per hour

1. INTRODUCTION

A maritime container terminal is a facility that handles container traffic from arriving ocean vessels to inland carriers and vice versa. The container terminal carries out multiple functions in container handling, including receiving, storage, staging and loading, for both incoming and outgoing containers.

Container shipping has a major role in global cargo transportation. Over 17% of the global sea trade is moved in containers. As over 80 per cent in volume of the world's merchandise trade is handle by ports, the importance of efficient and well-functioning ports is indeed notable. With the handling operations provided by the ports ranging from seaside to the yard and all the way to landside, the need for efficient container handling equipment becomes significant to ensure overall efficiency within the terminal. [3] Two types of automated container terminals can be generally considered: fully automated and semi-automated terminals. Both types incorporate automated stacking operations within the yard, but their difference is the horizontal transportation between the yard and quay, where semi-automated terminals use manned vehicles for the operation. A major advantage of manned horizontal transportation is that it is far better at handling unexpected situations. To compete with manned equipment and to ensure safe operation in any condition, better solutions must be made in both the handling of a single automated vehicle, and the terminal equipment manager. [24] This work focuses only on the solutions for the manager.

The goal of this thesis is to design and implement a model of Horizontal Transportation (HT). The desired model should be designed to be integrated into a terminal-scale operations model. The questions this thesis aims to answer could be described as:

- How to design and implement a HT simulation model for traffic control in terminal environment?
- Which features are needed for the system to be used for realistic simulation?
- Which features are needed for the system to be used as an optimization tool?

The modelling is done by using the waterfall model as a framework. The waterfall model is a tool for software development, where the development process is divided into discrete segments. These segments are executed in a sequence starting from the top-most segment and flowing to the bottom.

MATLAB Simulink is used to create the simulation model. Stateflow, a toolbox of Simulink, is used to model the event-based functions of the model, while Simulink is used to model the time-based kinematic functions. The simulation's visualization is set up with game-engine called Unity. HT operations handle the container transportation between the terminal quay cranes and stacking areas. In the model, a basic representation of a vehicle is created that can handle moving between commanded waypoints, and hoisting operations. The model also includes a HT manager, that manages the fleet of vehicles allocated to it by planning the paths for each transport to optimize container handling.

The structure of this thesis can be divided into 7 parts. In chapter 2, a brief theoretical overview of the maritime container terminal operations is given as a background for the reader to better understand the factors within HT modelling. Chapter 3 provides a more in-depth approach to different HT operations currently in use. Chapter 4 provides some insight in different methods of path planning, with a few examples of the different algorithms. Chapter 5 includes simulation and modelling theory found in literature. Chapter 6 shows the implementation of the system with explanations of the software used with modelling and visualization. In chapter 7, the validity of the implemented system is checked with similar tests found in literature. In chapter 8, conclusions and future steps of the problem are presented.

2. CONTAINER TERMINAL

Maritime container terminals serve as transshipment points for cargo traffic. The terminal acts as a temporary storage, so that the unloading of incoming vessels and the loading of trucks and trains for hinterland operations do not have to be synchronous.

This chapter provides a general background on container shipping and container terminals. In chapter 2.1, the current state of container shipping is explained. Chapter 2 explains the physical features of intermodal shipping containers. Chapter 2.3 showcases some container terminal layouts and their influence in horizontal transportation operations. Chapter 2.4 provides information on different available container handling equipment (CHE)

2.1 Container shipping

Container shipping is mostly done by liner operators. They transport goods with the use of high-capacity vessels that transit between terminals on a regular schedule. Three global alliances of liner operators provide the most capacity on the major East-West transit routes. These alliances collectively share over 90 % of the total deployed capacity. [3]. These alliances allow the operators to combine their fleets to better cater to their customers' needs. The required capacity changes with seasonal circumstances and market changes, among others. [26]

The average size of container vessels is growing. The largest vessels currently can transport over 20,000 TEUs (twenty-foot equivalent unit). [3] This growth also sets heightened requirements for the ports. The handling capacity of a single port might not be enough for the largest vessels. This forces the vessels to call for several ports, which in turn lengthens the round voyage time. [26]. Container vessels work on specific schedules with set number of port calls. In schedule planning, the carriers must make decisions in service frequency and port calls. Adding port calls may provide more revenue for the liner, with the downside of longer round times. Lower volume vessels allow for more frequent service, which meets the demand for shorter transit times, while larger unit sizes allow the operators to utilize larger vessels. [26]

Shipping operators prefer exact timetables as delays have negative impact on the reliability of the liner service as well as incur additional costs. Delays in shipping schedules can be divided into four groups: terminal operations, port access, maritime passages, and chance. The most common source of delay is terminal operations. As the

demand for more port volume increases, the availability of berths may not be guaranteed due to capacity constraints if the vessel misses its time window. The handling capacity of ports also add to the delays caused by terminal operations.

2.2 Intermodal container

An intermodal container is a standardized shipping unit designed for efficiency in cargo transportation. The efficiency comes from standardized outer dimensions to allow uniform handling regardless of the actual content. Container cargo is mainly shipped by sea, but also by land on trains and trucks.

The base dimensions and permitted gross weights of containers are defined by two ISO standards, ISO 668, and ISO 1161, which define the dimensions for the containers themselves and the corner fittings, respectively. In table 1, few standardized containers are presented. The most common types are the 20 feet and 40 feet standard containers. The standard width and height are 8 feet and 8 feet 6 inches, respectively. A taller version of container can also be found, called a High-Cube container, which has a height of 9 feet 6 inches. Every standardized container, regardless of size, has the same corner fittings to allow uniform lifting. The lifting is handled with a specialized device, called spreader. The structure of the containers along with the uniform corner fittings allow containers to be stacked firmly, which is essential for cargo shipping efficiency. [22]

Table 1: *Intermodal containers in ISO 668 and ISO 1161 specifications. [22]*

ISO	**Length**		**Height**		**Width**		**Max. Gross weight kg**
	ft	mm	ft	mm	ft	mm	
1AA 1AAA	40’	12192	8’6’’ 9’6’’	2591 2896	8’	2438	30480
1CC	20’	6058	8’6’’	2591	8’	2438	30480
1EEE	45’	13716	9’6’’	2896	8’	2438	30480

Occasionally, terminals may have to handle goods that do not fit in standard containers. This Out Of Gauge (OOG) cargo is stored in special containers, such as open top containers, or flat racks and platforms and must be handled with manned CHE. Break bulk cargo, such as cars or railway engines, cannot be containerized and also requires special handling procedures. Other cargo requiring special care and equipment is perishable and dangerous goods. Perishable goods are transported in reefer containers with applied cooling systems. These containers require an electric supply, so they are assigned to specific locations, both on shore and on a vessel. Dangerous goods that require special care are also stacked in specific locations. [5]

2.3 Terminal layout

A container terminal can be roughly divided into three areas: seaside, on which the vessels are berthed, the container stacking area and landside, from which any other means of transportation operates. [23] Seaside operations include loading and unloading of cargo vessels with the use of lifting equipment either on-board or on the quay itself, as well as moving the containers between seaside and container stacking area. Container stacking area handles the intermediate storage of the container terminal, by stacking inbound and outbound containers for vessels and storing export containers for train and truck deliveries. The stacking work can be done with HT equipment or specialized yard cranes. Landside operations consist of loading and unloading the hinterland transportation equipment. The CHE present in the landside operations vary based on the mode of hinterland transportation. [5] Figure 1 depicts the three operational areas of the terminal.

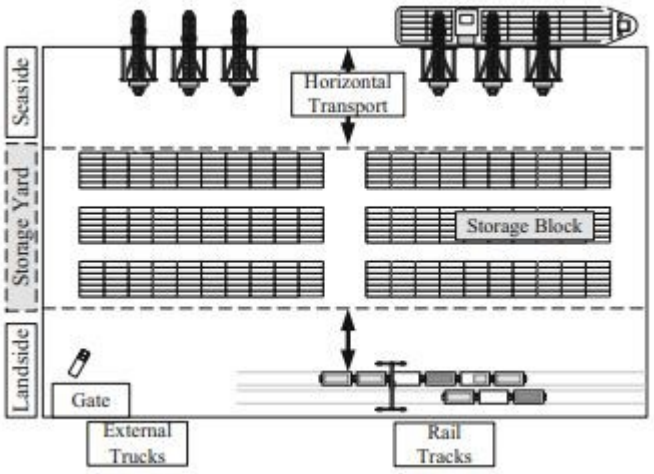


Figure 1. *Operational areas of a container terminal. [5]*

At seaside, vessels are moored within the quay boundaries. There are several ways the quay wall can be organized to optimize the vessel turnaround time. In discrete layout, the quay is divided into separate sections, called berths. A single berth can service one vessel at any given time. Continuous layout does not incorporate separate berths, so a vessel can moor anywhere within the quay boundaries. Combination of these two layouts is called a hybrid layout, in which the quay is divided to separate berths, but large vessels can occupy more than one section at a time and smaller vessels can share a singular berth. A special case of hybrid layout is so called indented berth, where a vessel can be serviced from both sides of the berth. [25] Layout options are presented in figure 2.

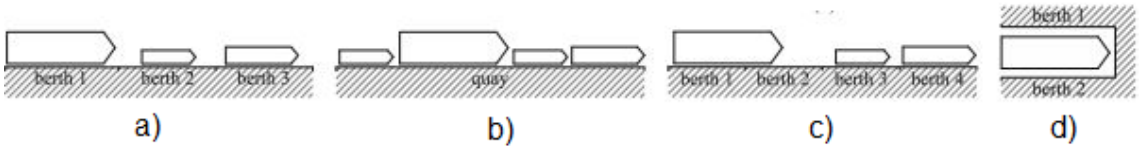


Figure 2. *Quay layout options. a) Discrete, b) Continuous, c) Hybrid, d) Indented. Adapted from [25].*

In the stacking area, containers can be arranged either parallel or perpendicular to the quay wall. This is largely based on the available stacking equipment. The yard is separated into driving lanes and storage blocks. The storage blocks are further divided into container slots, which are the containers logical position declared in bay, row, and tier. Bay and row describe the containers position in consecutive longitudinal and lateral directions, while tier informs the vertical stacking position of the container, starting from the container on ground.

2.4 Container handling equipment

The equipment that is used for container handling depends on the area of operations as well as other variables, such as:

- Vessel sizes
- The number of containers handled.
- Dwell time of containers
- Types of containers
- Available operating area (stacking density and geographic constraints)
- Type of hinterland transport
- Environmental factors (e.g., snow, ice, wind) [5]

On the quayside of the terminal, operations for loading and discharging of vessels are carried out. The vessel size and container volume largely determine the equipment required for this task. The type and number of quay cranes as well as their throughput set the scale for the required horizontal transportation equipment and yard cranes. The geographical constraints and the wanted stacking density further determine the required stacking equipment. [5]

In seaside operations, Ship-To-Shore (STS) gantry cranes handle most of the loading and unloading of vessels. While other quay cranes exist, the STS crane is the most common one used in container terminals due to its efficiency. STS cranes traverse the quayside via a track system parallel to the quay wall. The crane has a trolley, which

moves horizontally between the vessel and quayside. The trolley has a hoist equipped with a spreader, which allows vertical movement and gripping of containers. STS cranes can perform unloading and loading of vessels with landwards and seawards movement, respectively. [5]

While the STS cranes have attained the role of the usual quay crane, many a type of horizontal transportation exist. These include:

- Straddle carriers (SCs)
- Reachstackers
- Terminal tractors with trailers (Tractor-Trailer-Unit, TTUs)
- Shuttle carriers (ShCs)
- Automated guided vehicles (AGVs)

The selection of used modes of HT depends on the same variables mentioned before. Each HT equipment has its advantages and disadvantages. TTUs, AGVs and ShCs cannot perform stacking operations, while SCs and reachstackers can. However, stacking with HT requires more operating area, which leads to low density stacking due to the vehicles' vertical stacking constraints. SC stacking also requires space between container rows. While SCs can perform every picking, grounding, and stacking operations, and ShCs and reachstackers can perform picking and grounding, AGVs and TTUs cannot pick or ground containers and require other equipment to stack them. The ability to pick and ground containers removes the delay for these operations, while non-picking vehicles can move the containers faster. The combination of required HT equipment depends on each terminal. [5] Main features of HTE are compiled in table 2.

Table 2: *Features of HT equipment*

HT equipment	Automatic	Picking ability	Stacking ability	Stacking density
Reachstacker	No	Yes	Yes	Low
SC	Yes	Yes	Yes	Medium
ShC	Yes	No	No	High*
TTU	No	No	No	High*
AGV	Yes	No	No	High*

** paired with yard cranes*

When using HTE without stacking capabilities, the stacking operations fall on yard cranes. These include automated stacking cranes (ASCs), rubber-tyred gantry cranes (RTG cranes) and rail-mounted gantry cranes (RMG cranes). These cranes have similar stacking capabilities, but the main difference is in their interchange areas. The area

between a yard cranes' support legs is called a portal. ASCs' interchange areas are in both ends of its portal, while and RTGs interchange within their portal. This means that ASC has more stacking space compared to RTG. Like ASC, RMG allows interchange beyond the portal area. In terms of stacking capacity, the cranes are very similar, over 1000 TEU/ha. [5] Table 3 compiles the stacking features of these yard crane types.

Table 3: *Stacking features of common yard cranes.*

Yard equipment	Locomotion	Interchange area	Train loading	Truck loading
ASC	Rails	Both ends of stacking area	No	Yes
RMG	Rails	Between and beyond portal	Yes	Yes
RTG	Rubber tyres	Between portal	Yes	Yes

Landside operations are largely determined by the modes of hinterland transportation and the related interfaces. If most of the hinterland transportation is done with trucks, the operating area is often within the stacking yard. The trucks are loaded and unloaded by SCs or yard cranes. In the case of trains as hinterland transportation, the loading area should not be integrated into the stacking yard to avoid the yard equipment crossing the rails. Trains are often loaded with yard cranes, with the addition of HT to transport containers to and from the railway station. [5]

3. HORIZONTAL TRANSPORTATION

Horizontal transportation is performed in all the three main operating areas in a terminal. HT handles the container transportation between quay cranes and stacking area. Within the stacking area and landside operation area, HT can either perform the stacking of containers or relay them for yard cranes for stacking. Chapter 3.1 showcases most common forms of HT equipment. Chapter 3.2 further describes different operations done by the HTE in different terminal layouts. Chapter 3.3 compares the efficiency between the equipment.

3.1 Horizontal transportation equipment

Straddle carrier (SC) is a vehicle designed for freight transport. Unlike conventional trucks which carry their load on top of a bed, SC straddles its load underneath. This means that it can pick and ground containers without the need for extra equipment like cranes of forklifts. SCs have a lifting height of over 9 meters with the carrying capacity of 40 tons. [14] The high lifting height means that SCs also can stack containers atop each other, further diminishing the need for specialized stacking equipment. The usual stacking height of a SC is four containers high, which means a low stacking density in the stacking area. SCs exist in both manned and unmanned configurations, which provide trade-off between stacking height and labor cost, respectively. [5] In manned SCs, the operator is sitting sideways, which provides them a clear view both in front and behind the vehicle. Figure 3 depicts a typical straddle carrier.

Figure 3. *A straddle carrier. Adapted from [14].*

Shuttle carriers (ShCs) are in function similar to SCs as they straddle their load. They can pick and ground containers without cranes, which streamlines container handover in the terminal buffer areas. The typical lifting height for a ShC is over 6 meters with lifting capacity over 40 tons. [13] Shuttle carriers lack the stacking height of SCs, so they are better suited for seaside and landside handover operations. As SCs, ShCs are available in manned and unmanned variants. Shuttle carriers offer their operators high level of maneuverability which in turn can reduce terminal congestion. [13] Figure 4 shows a shuttle carrier straddling a container.

Figure 4. *Shuttle carrier transporting a container. [13]*

Reach stacker is a vehicle that uses a container spreader at the end of an extendable boom to pick, ground and stack containers. Reach stacker allow second row access in stacking due to the booms reach. This means denser stacking without the use of dedicated yard cranes. Reach stackers have a lifting capacity of over 40 tons. [12] Along with stacking operations, reach stackers can perform transportation jobs, so additional equipment is not needed in smaller terminals. Figure 5 shows a typical reachstacker.

Figure 5. *A reachstacker. [12]*

The most conventional way of transporting containers between the terminal is trailers towed by a tractor unit. This is a low-cost system, with a drawback of not being able to

pick and ground containers on its own. On the other hand, when serviced by quay cranes and yard cranes, TTUs can transport containers quickly with little downtime while loading and discharging. A single tractor unit can be fitted with several trailers. TTUs provide their operator a clear view both in front and to the rear platform which enables safe operation. Figure 6 shows a typical terminal tractor unit. TTUs are a robust solution for HT operations when the automation level of a terminal is low.

Figure 6. *A terminal tractor unit. [15]*

Automated guided vehicle (AGV) is mobile robot, that either follows marked paths or uses sensor data, including camera vision, radio waves or lasers to name a few, to navigate its environment. For container handling, the AGV is a flat-bed vehicle with the carry-weight of up to 70 tons. [11] AGVs are used only for horizontal transportation between the main operational areas as they lack any kind of picking or grounding abilities. As an unmanned vehicle, it offers reliability in terms of variables in transit and is suitable for high labor cost areas. AGVs offer a variety of safety features in operation as they are easily accessible by on-site personnel. They also incorporate anti-collision systems, collapsible bumpers and manual emergency stop buttons for any unexpected situation. Figure 7 depicts an AGV transporting a container.

Figure 7. *AGV laden with a container. [11]*

3.2 HTE application

Depending on the terminal's layout and container handling equipment, the operations for HT can vary. The selection of HTE is case specific for each terminal, with their own standards of optimality. This chapter provides some cases of HT operations with different solutions for HT operations. Each case has an STS crane working seaside.

3.2.1 Reachstacker with TTU

Using reachstackers and TTUs for stacking and horizontal transportation operations provides the terminal yard with versatility. STS cranes load and unload TTUs, which then manage the transportation to the container stacking area. Reachstackers perform the stacking operations in the yard so the terminal does not need additional stacking equipment. Reachstackers can also manage the loading and unloading for hinterland transportation, be it by train or trucks.

As reachstacker are easy to operate, they can be utilized very effectively in countries with little trained labor. Due to the ability of the reachstacker to perform HT and stacking operations, it is very well suited for small and medium sized terminals. The versatility of a reachstacker means it can be used as the only equipment (along with a quay crane) on the smaller terminals. Containers can be stacked 4-deep due to the reachstackers' reach, with the height of the stacks being 5 at maximum. Typical density for containers with the height of 4 containers is approximately 500 TEU/ha, and approximately 350 TEU/ha for 3-high stacks. The main advantages for a system like this are low investment and capital costs due to the TTU's and reachstackers' relatively low cost, and the ease of use for both of the equipment. Disadvantages for the system are two separate handover operations as different equipment for transportation and stacking procedures, and inability for the TTUs to pick or ground containers on their own. The lack of

automated systems for this case's equipment can be seen as an advantage or disadvantage, depending on the labor costs of the country the system is applied in. [5]

3.2.2 Straddle carrier system

As straddle carriers can perform both stacking and transportation operation, they can be employed as the only equipment on a terminal yard in addition to quay cranes. SCs pick and ground containers from the STS cranes' portals and transport them to the stacking area. SCs can also service landside transportation equipment, further reducing the need for additional equipment. The lack of fixed positions, such as rails for yard cranes, means that the layout of the terminal can be easily altered when using this system. The system requires clear traffic lanes on the yard, and so the stacking density is medium, approximately 500 TEU/ha with 2-high stacks and approximately 750 TEU/ha for 3-high stacks. The maximum stacking height for a SC is 4-high. [5]

Advantages for a pure SC system are versatility of the SCs, with the capability of performing all the necessary operations within the terminal yard. As no further equipment is needed, container handover times are either non-existent or very short, which in turn enables the STS cranes to operate highly efficiently. The SCs versatility of operation also reduces the number of vehicles needed in operation while simultaneously maintaining high number of concurrent moved containers. This reduces the labor costs for the terminal operations as well as the disturbance on the yard when loading and unloading trucks. Downsides for a system like this are high investment and maintenance costs for the equipment. Compared to yard cranes, SCs require more space for stacking with lower density. If the transportation distances are long SCs quickly become a worse choice because they are slower and more costly compared to TTUs. [5]

3.2.3 HT with yard cranes

Utilizing yard cranes for container stacking in the terminal yard gives the system freedom when choosing transportation equipment. The most common types of yard cranes (see table 3) share a similar stacking density of 1000 TEU/ha for 4-high stacks. Without the need for stacking ability for the HT equipment, any type of equipment can be used. The choice of equipment then stems from the size and layout of the terminal, and the performance requirements for the vehicles. Some vehicles are faster than other, some are more maneuverable. Overall cost of equipment varies noticeably between vehicles, as does the level of automation. When applying HT system with high level of automation (such as AGVs or automated SCs/ShCs), the labor costs can be kept very low, but investment cost can be very high.

Utilizing yard cranes provides an advantage for space management, as the stacking density is so high. Yard cranes' hoisting operation requires no traveling lanes between the container rows, which further increases the density. The rubber-tyred cranes also provide flexibility over the areas of operation, as they can be transported between different operational areas. The investment costs per piece of equipment are at medium range. The use of different equipment for HT, stacking, and landside operations necessitates multiple handover procedures. This along with loading/unloading trucks in the stacking area delay the operations of the yard, which is disadvantageous for the system. [5]

4. PATH PLANNING

For a vehicle, manned or unmanned, to move in an environment collision-free requires planning. For a single vehicle, this process is straightforward. The situation becomes more complex when the number of vehicles increases, or the environment has obstacles. In this chapter basic information about the configuration and environment along with some algorithms for single entity and multiple entity problems are explored as well as specific applications in a terminal context.

4.1 Configuration and environment

Any path planning problem contains two major factors: the moving entity and the environment. The environment consists of free space and space that is occupied by possible obstacles. The start and goal positions of a path are in the free space, so the vehicle can traverse to them while avoiding possible obstacles. If the environment contains moving obstacles, it is considered a dynamic environment, otherwise it is a static one. [17] The vehicle is defined by a set of parameters, including position and orientation, which tell the number of degrees of freedom. These parameters are called a configuration and provide the space for every possible configuration and transformation of that vehicle. [16]

For path planning purposes, the environment must be presented in a mathematical format for the algorithms to process it. The configuration space can be presented in a reduced subset of configurations that include start and goal position configurations. The remaining free space can be divided into any number of intermediate configurations and transitions between them. Actions in a graph can be weighted, so that some configurations become more or less optimal to transition to. [17] A graph can be described as a network of nodes, called vertices, and transitions between them, called edges. In directional graphs, some edges may be traversed in only one direction. Figure 8 shows an example of a weighted graph and a directed and weighted graph.

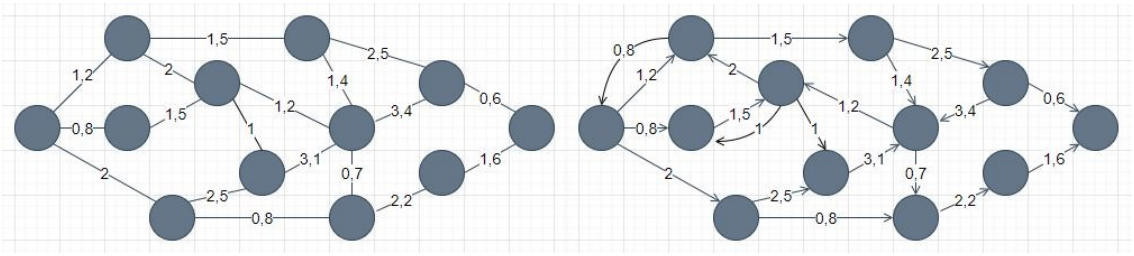


Figure 8. *Example of a weighted graph (left) and a directed and weighted graph (right).*

Another way is to partition the environment to simple geometric shapes called cells. The environment can be decomposed to cells either accurately or approximately. In an accurately decomposed environment, the cells are entirely in the free space or entirely in the obstacle space. This means that the free space cells completely cover the configuration free space. In approximate decomposition, cells are formed so that some cells can contain configuration free space and parts of an obstacle. Every cell that contains at least a part of an obstacle are marked as occupied space, and the rest are marked free. If the cells are the same size, the decomposition forms an occupancy grid. This decomposition, however, is not a lossless decomposition as uniform sized cells cause the obstacles to become enlarged and some passages between obstacles can be lost. Variable cell size can be used to minimize the loss of environment information. The environment is initially divided into large cells. If the cell is entirely in either free space or obstacle space, it remains as is. If a cell is partly occupied by an obstacle, that cell is divided into smaller partitions until a suitable resolution is formed. [17] Figures 9 and 10 depict examples of accurate cell composition and approximate uniform cell composition, respectively. In the approximate representation, the cells that are categorized into object space have a gray background.

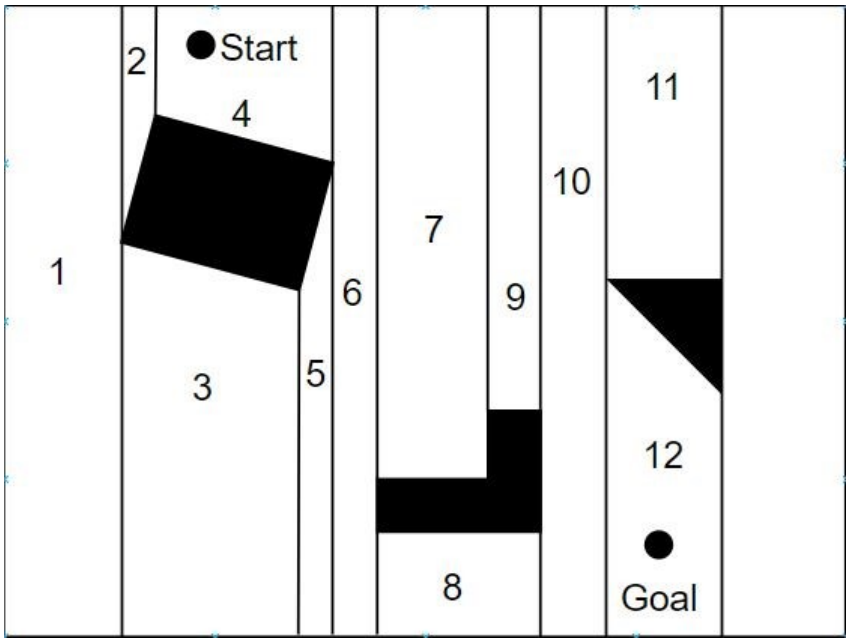


Figure 9. *Example of accurate cell composition*

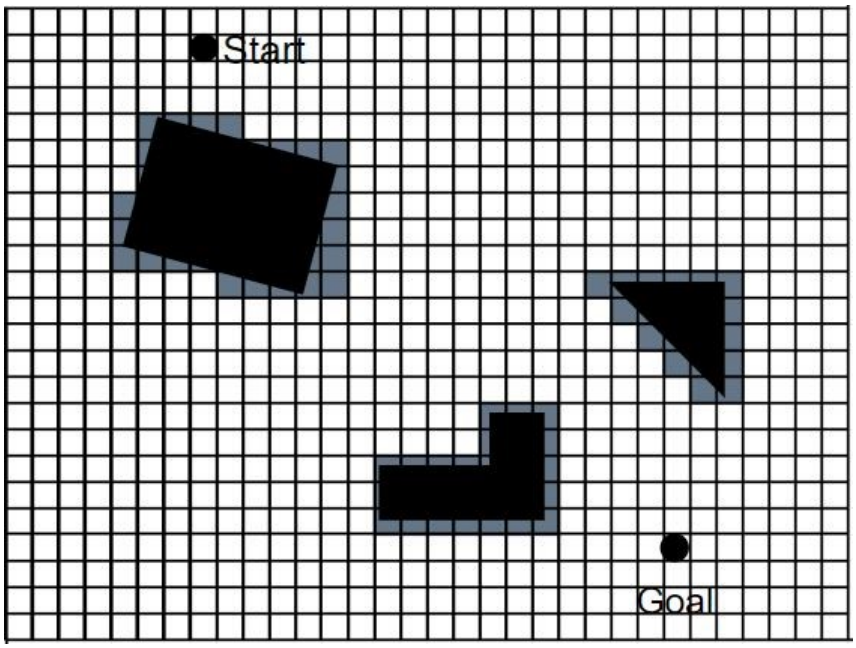


Figure 10. *Example of approximate cell composition with uniform cells.*

The environment can be modeled into a potential field, where it is divided into attractive and repulsive sections. The start configuration and the obstacles are modeled as repulsive fields while the goal is modeled attractive. Potential field method serves innately as obstacle avoidance model as the robot can be guided with the potential fields towards the goal. This method, however, can produce problems as concave obstacles may produce local minima, in which the vehicle can be stuck. This is because the vehicle position is used to calculate the negative gradient of the potential field, which leads the vehicle to the goal position. [17,16]. Figure 11 shows an example of a potential field, in

which the start position is the highest point and goal the lowest. The path-planning problem could be explained as motion of the vehicle rolling down the hill. Obstacles are modeled as heightened areas along the path.

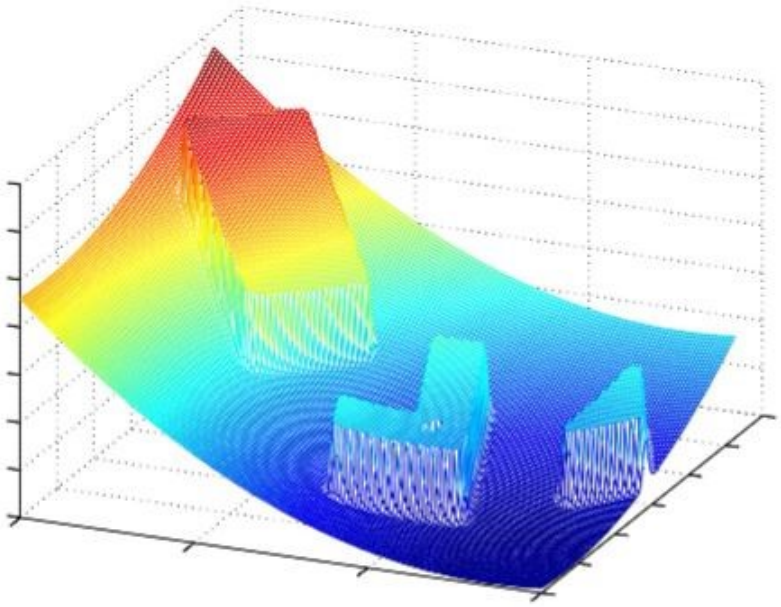


Figure 11. *Example of a potential field. [17]*

4.2 Path planning algorithms

The point of path planning is to find a continuous path from the given start location to the goal location. The entirety of the path must be in the environment free space. By performing actions, the vehicle changes configuration, and traverses in the configuration space. By following this logic, a path is a sequence of actions within the configuration space which guides the vehicle from the start to the goal. Between the start and goal positions, any number of paths can be created. Of these possible paths, an optimal path is chosen. The optimality changes based on the criteria on each application and algorithm. Some criteria could be shortest path, shortest time, farthest from obstacles and without sharp turns or other motion constraints.

Simple path planning algorithm does not necessarily need global map information. A very basic bug algorithm uses only local knowledge to traverse the environment. As such, the algorithm requires low memory usage, but also the generated path usually is far from optimal. Bug algorithms use two main behaviors: moving in a straight line toward the goal and following obstacle boundaries. While the basic behavior is similar between different bug algorithms, interaction with the obstacle contour varies. Some determine a main line that connects the start and goal positions and follow that while following encountered

obstacle contours until they can return to the main line. Others traverse the entire contour and continue towards the goal from the point with the shortest Euclidean distance. [17]

If the environment is known and sufficiently modeled into a graph, more advanced path planning algorithms can be used. These algorithms generally proceed by checking the start position to determine if it is also the goal position. Usually that is not the case, so the search is expanded to the neighboring nodes. Depending on the algorithm, a neighbor node is chosen as the best next step and the search continues until the goal node is reached or all possible solutions are searched. Two simple graph-based search methods are breadth-first (BFS) and depth-first (DFS) searches. As their names suggest BFS iterates the graph in a breadthward and DFS in a depthward motion, this is depicted in figure 12.

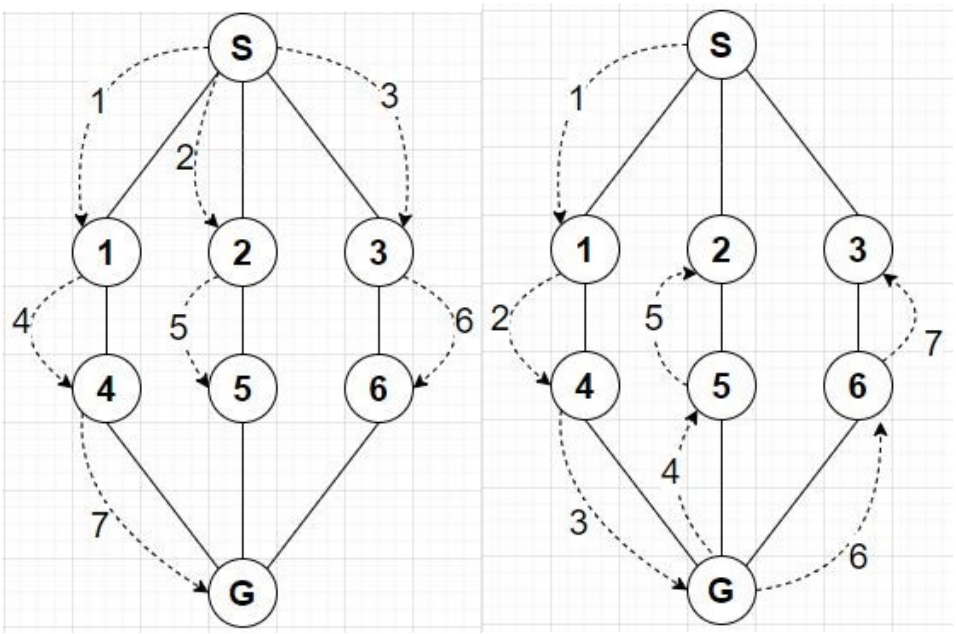


Figure 12. *Graph iteration of BFS (left) and DFS (right)*

One such algorithm is Dijkstra's algorithm, which uses breadth-first search to find shortest paths from a single source vertex to each other vertex. Dijkstra is a noninformed algorithm, which processes the vertices in increasing sequence according to their distance from the source vertex. After exploring each vertex, a shortest path can be deduced between any two vertices. Such path is the one with the least number of edges. [7] The basic workflow of Dijkstra's algorithm is as follows:

- Create a shortest path tree set (SPT set), which tracks of the vertices that have been processed. Initially this is empty.
- Initialize the cost to each vertex, 0 for the source vertex, INF for the others.
- While the SPT set does not include all vertices:

- Pick the vertex with the minimum value that is not yet in SPT set.
- Include that vertex in the SPT set.
- Update the distances to the neighboring nodes of this vertex. This is done by calculating the cumulative cost from source to each neighboring vertex.

If some vertices are accessible from multiple other vertices, the final cost will be the minimum cumulative cost from the source node. This means that some vertices may be processed multiple time during execution. The algorithm can be modified to work on a single source, single target implementation by breaking the execution loop after finding the goal vertex.

Other well-known algorithm for graph-based path planning is A* (A star), which is an informed algorithm as it uses some heuristic or other additional information for path calculation. Heuristic is the cost estimate for a path from current node to the intended goal node. This allows the algorithm to choose among promising vertices, which may lead to the solution more efficiently. The heuristic can be calculated by any number of functions depending on the problem at hand, but common choices include Manhattan (sum of horizontal and vertical moves) and Euclidean distances.

A* algorithm calculates the cost for the whole path for each node. This includes the cost to that node from source, and the cost to goal from that node. The algorithm's execution is like that of Dijkstra's, as A* also uses two sets for nodes: processed (closed) nodes and yet-to-be processed (open) nodes, which initially only has the source node. The execution is as follow:

- Take the first node from the open, which is sorted in increasing order of cost-of-the-whole-path.
- For all neighboring nodes calculate:
  - cost-to-goal
  - cost-from-source
  - cost-of-the-whole-path
- Store this information for each respective node. If one of these nodes already has information stored, compare the two and store the lower value. Store current node as the previous node for the other nodes.
- Visited and updated neighboring nodes are added to the open list. The current node is removed from open and added to the closed list.

The algorithm finishes when the goal node is added to the closed list. If the estimated cost to goal is smaller or equal to the true cost, the algorithm is guaranteed to find the optimal path. [17]

4.3 Multi-agent pathplanning (MAPP)

When multiple vehicles operate in the same configuration space, the path-planning problem becomes even more complex. MAPP consists of finding each vehicle a suitable path to their goal, while simultaneously avoiding collisions with other vehicles. For individual vehicles, other vehicles within the configuration space could be described as dynamic obstacles, with the exception that vehicle paths are controlled while obstacles might not be [19]. Vehicles collide when multiple agents occupy the same position, or when traversing the same edge in different direction. Graph positions are accessible for vehicles if no other vehicle occupy it on current time, or if no other vehicle plans to occupy it in the following time steps.

MAPP traditionally can be divided into two approaches: centralized and decentralized. Centralized method describes the multitude of vehicles as one multi-bodied robot, with a single decision maker. This enables the use of basic path planning methods for the problem. While this approach may prove theoretically efficient, in practice it becomes very complex system which scales poorly for many vehicles. Decentralized method, however, handles the route calculation for each individual vehicle separately and considers the interactions between agents as a secondary phase in the planning. This may reduce the computation needed, with the added loss of completeness. [19]

MAPP is used in a variety of applications, including transportation and warehouse management, which are the most evident in the scope of this thesis. Transportation applications include autonomous vehicles that traverse in an area and abide to specific traffic rules. Warehouse management problems focus on a fleet of agents working together to retrieve and process packages.

4.4 Terminal environment

As discussed in chapter 2, the terminal is a dynamic environment, which affects the decision choices for the path planning for horizontal transportation operations. For this thesis, the noteworthy problems are separated into two categories: restricted movement and dynamic obstacles. Restricted movement can be defined as the limited movement options provided by the HTE constraints and regulated traversal options, such as lanes and area interfaces. The dynamism of obstacles can be described as the movement of

other equipment in the terminal and the changing location of handled containers within the areas of the terminal yard.

The various equipment used in HT have different options for movement. Some, such as TTUs and reachstackers, can move in only one direction, with the ability to reverse to maneuver. Other, such as SCs, ShCs and AGVs, are bidirectional and can traverse in both directions along their longitudinal axis while retaining full maneuverability. The terminal yard is often divided into areas of operation for different container handling equipment with interface areas for the HTE, areas are connected to each other by a network of lanes, which are used to restrict and control the movement of equipment and personnel within the yard. These lanes can be uni- or bidirectional depending on the terminal design choices. The combined restrictions provided by the equipment constraints and the lane network lay the basis for the path planning design for the terminal yard operations.

This, however, would only prove adequate in the special case that the only object populating the yard is the transport for which the path planning was performed. The more general case contains other equipment on the yard along with containers in storage and handling. With only one HTE, the problem for other equipment getting in the way occurs in the area interfaces. With several HTE, the yard is even more populated with obstructive equipment, traversing in the main environment for HT operations. The addition of containers to the yard further complicates the path planning design, as they act as obstacles on the yard areas. Some vehicles (SCs and ShCs) can traverse through an occupied container position if the stacked height of the container(s) does not exceed the vehicle's maximum clearance. For other modes of transportation, such container stacks form an impassable obstacle, and as such, must be taken into consideration while planning their path.

By using everything discussed in this chapter, the path planning design for terminal yard operations can begin. The environment free space can be expressed as the network of lanes and container areas accessible to the transportation equipment. The end configuration of the paths should be within these areas, and the start configuration is the vehicle's current position, which in addition to the free space, can be an "outside" vehicle storage yard not necessary for every path planning calculation. The containers and other yard equipment provide the dynamic obstacle space which the vehicle must avoid. The lanes, areas and their intersections form the basis for the mathematical representation of the environment. For the purposes of this thesis, a weighted graph is formed of these elements. The nodes of the graph are the intersections of lanes within the yard as well as the active container positions within the areas. Edges of the graph are formed as the

lengths of the lanes that fall between these intersections and their weights are the individual distances between two nodes. This graph enables the usage of sophisticated path-planning algorithms while still maintaining the lane-based solution necessary for this implementation. For multiagent path planning edge weights can be altered for blocking some routes if they intersect other equipment’s planned path.

5. SIMULATION AND MODELING

When planning terminal operations, simulations can prove crucial. By modelling and simulating the environment, operators can test different configurations and inputs without affecting the real system. Simulation models are used to evaluate dynamic processes within the container terminal to analyze key statistics. These include average productivity and waiting times as well as number of moves. The statistics are gathered to identify possible bottlenecks in the terminal's processes. Simulation is used when planning for a completely new terminal or while studying the performance of an existing one. Either way, critical logistic decisions can be tested in multiple configurations to optimize them before implementation. [10]

5.1 Simulation basics

The first step in understanding modelling and simulation, is to explain the concept of a system. System can be defined as a combination of components that act together to perform a function. This function would not be performed as is without all the components. Systems are used to describe physical objects and natural laws. [6] In the context of this thesis, the system is the combined functionality of the container terminal.

Systems can be classified as static and dynamic systems. In static systems, the output value is dependent only on the concurrent input value. In dynamic systems, on the other hand, the output depends on both the concurrent input value, as well as all past input values [6]. Most natural systems are dynamic, as they can be quantified with equations that use continuous variables evolving over time. [27]

Description of a system, often referred as a model, contains the necessary information of the system's technical process. Models duplicate the systems' behavior by using mathematical equations and measurable variables to describe it. A model is used to control the system as it indicates how it reacts to certain control actions or external occurrences. By simulation, the model can be evaluated numerically. Simulating the process provides data from the system's behavior, which can be used to analyze the effects of dynamic inputs without the actual system. This often is more cost-effective than testing with the actual system. [6]

5.1.1 Dynamic models

Modeling dynamic systems can be achieved in two ways. Either by combining basic physical equations and principles, or by using measurement data from the real system. Typically, modeling process utilizes both approaches. The physical approach for modeling uses balance equations for force, mass, torque, and energy [27]. Modeling with measurements from the actual system requires the system's variables to be defined and measurable. Data is gathered by measuring these variables over time. These variables include input and output data. Both sets of variables are measured and compared at any given time. The relation between the input and output variables produces an approximate model of the system. The model does not necessarily represent the system accurately, so it should be validated and verified for their purpose [6].

As systems may be quite large and complex, it is important for the modeler to decide the noteworthy features in them. Any features that are unimportant should be left out, and only the essential features added. This, of course, requires knowledge of the model's requirements, so the modeler can decide the essential features [21]. Some systems may have a lot of different variables, so the modeler should proceed with only the interesting variables regarding the exact modeling problem.

Dynamic models can be divided into several categories, but the three major categories are:

- Continuous time models.
- Sampled time models.
- Discrete event models.

Continuous systems are defined with linear or non-linear differential equations for force, energy, mass, or momentum balance. Many non-linear equations can be linearized to ease their usage. Sequential, or discrete systems are defined with linear or non-linear difference equations. Their output information is only acquired in certain discrete time-steps. Computer-supported control almost exclusively incorporates discrete description as computers work sequentially in time. Continuous time models can also be discretized by sampling the data. Choosing sampling time is also a part of the modeling process. Sequencing systems, also known as discreet event models, describe processes with separate events executed in sequence. Signals in sequencing systems are often binary on/off in nature. Sequencing systems are often found in industrial processes. [27] In the context of this thesis, the modeling is largely done to sequencing systems, so they are detailed more below.

Sequential systems can be further separated into two categories: combinatorial and sequencing networks. Combinatorial network has a true or false -output condition that depends on a multitude of input variables that must be fulfilled simultaneously. These kinds of systems have no memory, so the conditions are always checked at present time, which renders these systems static. They can be used for example as safety checks to permit a manual control action, only allowing the system to process if all input conditions all met. Sequencing networks, on the other hand, takes into consideration the present and past values of process states and inputs, and as such, are dynamic systems. The states of a system are conditions at a given time instant that describe the system's behavior at that time instant in a measurable way [6]. Only one state can be active at any given time in a system. The states are accessible by transitions that are triggered by events or user inputs. If the transitions between the states are dependent on logical conditions, it is called asynchronous. If the transitions are triggered by a clock pulse, the transitions are called synchronous. Industrial applications commonly favor the asynchronous transitions. [27]

5.1.2 State machines

Another way to represent discreet event systems (DES) is by using state machines. State machine is an abstract machine that represents the behavior of a DES with the use of a state transition diagram. In the case of a deterministic state machines, the transitions in the diagram are labeled distinguishably, so that transitions out of a state cannot share a label. An example of a deterministic state transition diagram is depicted in figure 13 [6].

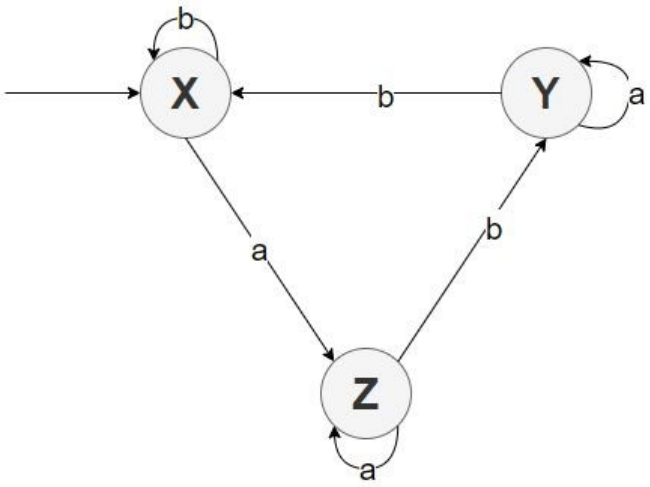


Figure 13. *State transition diagram of a deterministic system.*

The state diagram in figure 14 has three states of *X*, *Y* and *Z*, with two events *a* and *b*. The initial state is *X* as indicated by the empty transition. The transitions can be triggered

either spontaneously by the system itself, or by some external input. While the system is in a state, when a permitted event triggers, a transition is done. For the case of the example diagram, if the system is in state *X* and event *b* triggers, transition happens, but without a change in states. Only when event *a* triggers, is the state changed via a transition. The same goes for the rest of the states. State machines can contain many states with varying relations, depending on the modeled system. Both the states and transitions can contain functions that manipulate the systems intrinsic variables, if activated.

When a system includes both time-driven and event-driven dynamics, it is called a hybrid system. Many current systems such as aircrafts and chemical processes among others, make use of a hybrid implementation. Usually the control logic is event-driven, while the process may be a complex system with time-driven dynamics. [6].

5.2 Modelling process with waterfall model

The system in this thesis is modelled by utilizing a software development tool called waterfall model, see fig 14. The model consists of discrete phases, with the process flowing from top to bottom. The segments all have feedback loops between each segment to allow modifications due to new information that is gained in the process. [4]

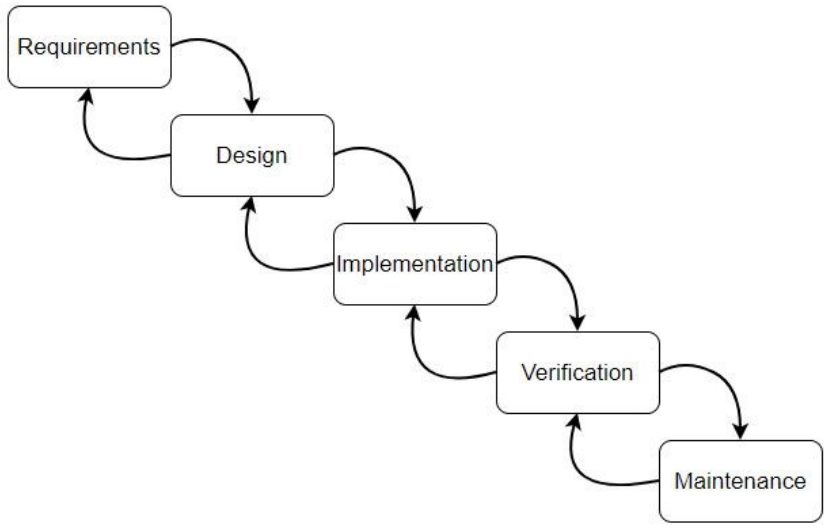


Figure 14. *Waterfall model. Adapted from [4]*

The first phase of the model involves mapping and analyzing the stakeholders' needs for the system. These lay the basis for the system in terms of functions that the system must perform, external systems it must be compatible with and the performance levels it should reach. The requirements are used to compile a requirements specification which is used as input for the next phase.

The result of the previous phase is used to define the architecture, interface, and data store choices for the system. The system design is done to satisfy the requirements specified in the first phase. The design choices determine the implementation solutions in the next phase.

The third phase is the actual implementation of the system as per the design specification of the previous step. During the implementation, unit testing is performed to determine the validity of the progress made as part of the whole system. The feedback loops of the model allow new design choices being introduced and implemented during the process.

The implemented system is verified in the next phase. This is done by testing that all the requirements for the system's functionalities and performance are satisfied. Three types of testing are usually done: unit testing, mentioned in the previous phase, system testing for the entire integrated system, and acceptance testing, which is done by or on behalf of the stakeholder. Any defects detected in this phase are logged and corrected via the feedback loops.

After the system is verified and deemed ready, it is installed, and the last phase begins. The maintenance phase consists of performance improving modifications done to the system. The modifications are due to change request made by the users or the defects discovered in the extensive use of the system.

5.3 Modelling methods for wheeled vehicles

A wheeled vehicle is a complex system that is composed of multiple subsystems that have different dynamics. Steering and suspension systems define characteristics in different motions. The driving environment provides random external inputs to the vehicle, with the addition of the complexity to measure interactions within tire-road interface. In the case of a manned vehicle, the human-vehicle interaction can affect the characteristics and dynamic behavior of the system. All of these have to be taken into account when compiling an accurate model of a wheeled vehicle. As for the scope of this thesis, we will focus more on the directional behavior and modeling of the vehicle. [2]

The basis for a vehicle model is the range of motions that it is able to perform. A vehicle can move according to six degrees of freedom (DoF). These are made up of three rotations and three translations around and along the vehicle's main axes. The DoF are shown in figure 15.

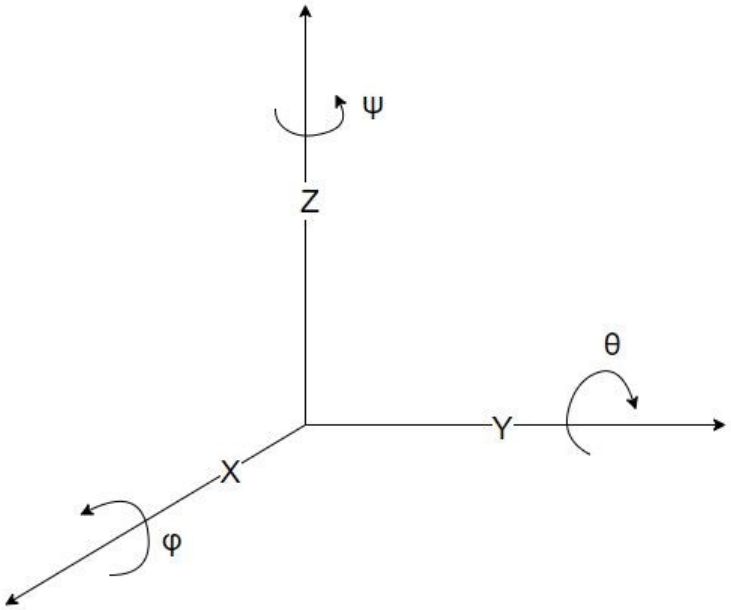


Figure 15. *The six DoF of a vehicle.*

Translation along *X* axis describes the vehicle's longitudinal motion, translation along the *Y* axis is called lateral movement and translation along *Z* axis indicates vertical motion. Rotation around *X* axis is called **rolling** rotation *φ (phi)*, while the rotation around *Y* axis is known as **pitch** *θ (theta)* and the final rotation around *Z* axis is called **yaw** *ψ (psi)*. In this thesis, the three rotations are henceforth called as yaw, pitch and roll for clarity's sake. We can identify major forces that affect the motion of a vehicle, one being the lateral force from the steering, and the other being the longitudinal traction force responsible for accelerating and decelerating the vehicle. These forces are dynamically connected but are often treated as separate [2].

Steering in a vehicle is done via a direction system. It is compiled combination of many components that are used to exert steering force to the directing wheel(s). The longitudinal force is exerted by the presence of a traction force in the tire-road interface. The friction present in the interface transforms the tire's tangential speed into the vehicle's longitudinal speed. These two forces can be used to approximately represent a dynamic four-wheeled vehicle by using a so-called bicycle model. Bicycle model combines the four wheels of a vehicle into two equivalent wheels. Figure 16 depicts a bicycle model of a four-wheeled vehicle. The local vehicle coordinate system is marked with red, while the global coordinate system is marked with blue.

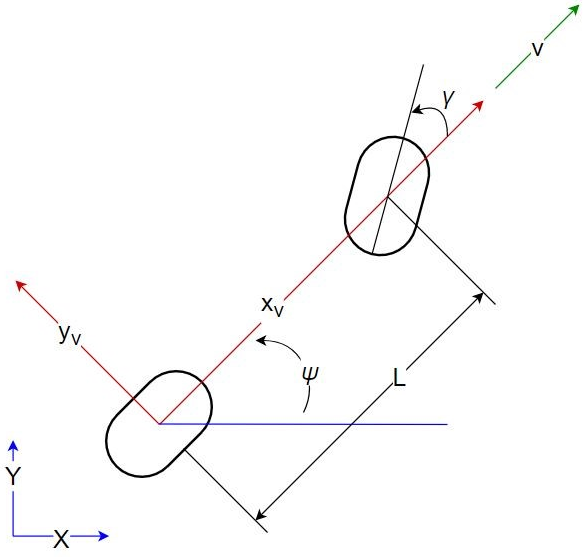


Figure 16. *Bicycle model of a four-wheeled vehicle*

This representation leaves the roll and pitch rotations out of the dynamics for the vehicle, but as such can represent a wheeled vehicle in an urban environment. The bicycle model allows us to simulate two DoF of the vehicle, namely the longitudinal translation along *X* axis and the yaw rotation around *Z* axis. These two are shown in the figure 15 as rear wheel speed *v* in direction x_v and the angle of yaw rotation *ψ*. The yaw angle is controlled by the steering angle *γ*. The speed of the vehicle in y_v is zero as the wheels cannot slip. The vehicle can be represented with three coordinates in the global coordinate system *q = (x, y, ψ)*. The velocity *v* can be calculated as

$$v_x = v, v_y = 0 \tag{1}$$

As the reference point of the vehicle in a turn follows a circular path, its angular velocity can be calculated by.

$$\dot{\psi} = \frac{v}{R} \tag{2}$$

From equation (2) the turning radius *R = L / tan(γ)*, can be calculated, where *L* is the wheelbase length. The steering angle *γ* is limited, and as such defines the minimum value of *R*. In the global coordinate system, the vehicle can be defined with the following equations.

$$\dot{x} = v \cos(\psi) \tag{3}$$

$$y = v \dot{\sin}(\psi) \tag{4}$$

$$\dot{\psi} = \frac{v}{L} \tan(\gamma) \tag{5}$$

The change rate of heading $\dot{\psi}$, also called the turning rate of the vehicle is proportional to the vehicle velocity *v*. This means that the vehicle's orientation cannot be changed while stationary. Although simple, this model is more than capable of simulating a wheeled container handling equipment as a part of a horizontal transportation system within a terminal environment.

6. HORIZONTAL TRANSPORTATION MODELING

This chapter describes the modelling process of HT operations thoroughly. Chapter 6.1 presents the software used in the modelling and simulation process. Chapter 6.2 describes requirements for the model and Chapter 6.3 the system design. Chapter 6.4 describe the implementation and process.

6.1 Software

The software used for modelling and simulation is MATLAB Simulink. It is a graphical programming environment for model-based design and multidomain simulation. It uses block diagrams to model algorithms and physical systems. The software allows the division of systems into coherent sub-systems, which enables the user to generate hierarchical models. [29]

Simulink's toolbox Stateflow provides tools to model and simulate decision logic by using flow charts and state machines. Combined with the modelling capabilities of Simulink, these tools enable the design of task scheduling, communication protocols and hybrid systems, among others. [31] Discrete event-driven systems, like control systems, can be modeled with Stateflow. Continuous time-driven dynamics, such as the kinematics of a vehicle, can be modeled with Simulink.

For the model visualization, a game-engine called Unity is used. Its primary use is in game development but can be used for engineering purposes. The visualization runs concurrently to the simulation, which means that it is displayed while the simulation is active.

6.2 Model requirements

Stakeholders' needs for a system, often called user requirements (UR), define what the users want the system to do. Based on these higher-level requirements, more defined functional specifications (FS) are drafted. Functional specifications are descriptions of functions required to satisfy user requirements. They are expressed in system implementation terms.

The system in this thesis is implemented as a part of Kalmar's terminal operations model, and their needs for the system outline the user requirements for the system. The user requirements as well as their more detailed descriptions are listed below, along with the corresponding functional specifications.

User requirements:

1. Simulate multi-vehicle traffic control in a terminal environment.
2. System is integrable to a larger scale terminal model.
3. Simulation must be able to run multiple times faster than real time.

These user requirements can be divided into sub-requirements to provide a more detailed description of the requirements as well as the goal of each requirement regarding the designed system.

UR1 – Terminal operation simulation

UR 1.1	Simulate container handling equipment.
Goal	Monitor traffic in terminal yard transportation.
Description	The kinematics of SC are modeled. This includes longitudinal movement in both directions and hoisting operations.
Output	Location timeseries of SCs and containers.

UR 1.2	Simulate a fleet manager.
Goal	Provide a control logic that imposes traffic rules in the terminal yard.
Description	A fleet manager for the terminal yard provides traffic regulations, such as lanes, for the terminal yard. Container handling operations are performed to abide to these rules. The path-planning is graph-based.
Output	Routes and localization data for the simulated vehicles.

UR 1.3	Enable multi-agent operation.
Goal	The system can perform multi-agent path planning operations.
Description	The system's fleet manager is capable of planning multiple routes simultaneously. Routes are inspected for collision detection.
Output	Collision detection and route optimization.

UR2 – System integrability

UR 2.1	Read and use data provided by terminal-scale model.
Goal	Utilize container and vehicle information from the upper-level system for simulation.
Description	The architectural choices of the HT system reflect those of the terminal model. This allows seamless transmission of information between the systems.
Output	Initialization data for the HT model.

UR 2.2	Generate data that is usable for the terminal-scale model.
Goal	Provide the upper system with accurate simulation information.
Description	The architectural choices of the terminal model are utilized to allow the HT simulation data to be streamed for visualization.
Output	HT simulation data for the terminal model.

UR 2.3	Use well known methods for simulation parameters.
Goal	Simulation is easy to operate.
Description	Using well known documentation methods for setting the simulation's parameters enables the easy configuration of different simulation scenarios.
Output	Simulation configuration parameters.

UR3 – System performance.

UR 3.1	Use efficient data structures and algorithms.
Goal	Minimize computational times.
Description	The data structures and algorithms of the simulation model are chosen in a way that minimizes route computation times.
Output	Multiple times faster than real-time simulation.

6.2.1 Functional specifications

The user requirements have been analyzed to produce functional specifications for the HT simulation model. The functionalities performed by the HT simulation model can be classified into 3 main functions:

1. Initialization.
2. Run HT operation.
3. Run visualization interface.

These functions can further be divided into subfunctions that describe the functionality in detail, with its required input and output arguments. These functions are implemented with information available from the terminal-scale model. The functionality of the terminal-scale model regarding the scope of this thesis is briefly explained in chapters 6.3 and 6.4.

FS 1: Initialization

FS 1.2	Generate lanes.
Description	The routes are planned according to traffic regulations in the terminal yard. A lane system is created to connect container areas.
Input	Container area locations
Output	Terminal yard lanes.

FS 1.1	Initialize container locations for HT operations.
Description	Container areas and locations are needed for lane generation as well as path-planning operations.
Input	Container locations from terminal-scale model.
Output	Container locations for HT operation.

FS 1.3	Create graph notation of the terminal yard.
Description	Routes are planned with a graph notation. The lane system information is gathered and manipulated to represent a graph.
Input	Lane and intersection information.
Output	Graph for path-planning.

FS 1.4	Create a job priority list.
Description	The simulation model handles a finite number of jobs for each simulation scenario. A job list is required to track and allocate jobs for each vehicle.
Input	Container locations, drop-off locations.
Output	Job list with pick and ground locations for each container.

The initialization functions prepare the simulation environment for operation. The inputs required are either given to them by the terminal model, or by the user directly in the HT model.

FS 2: Run HT operation.

FS 2.1	Simulate SC vehicle longitudinal movement.
Description	The simulation model should realistically portray the movement of a SC in a terminal environment. This includes movement in both directions along the longitudinal axis, along with steering.
Input	Acceleration and waypoint direction data.
Output	Location and orientation data for a moving vehicle.

FS 2.2	Simulate SC vehicle operation logic.
Description	Along with the kinematic model of a vehicle, an operational logic is required. The logic

	handles the route waypoint information and acceleration values.
Input	Route waypoint.
Output	Acceleration and waypoint direction data.

FS 2.3	Assign a job.
Description	Each vehicle in operation is assigned with a container pick and ground locations.
Input	Void
Output	Target coordinates for route calculation.

FS 2.4	Calculate routes for vehicles.
Description	Routes to target coordinates are calculated from each vehicle's current location. The paths are created as sequences of graph nodes.
Input	Target coordinate, current vehicle coordinate.
Output	Sequence of graph node coordinates as waypoints.

FS 2.5	Send paths to vehicles
Description	Planned route sequences are transmitted to corresponding vehicles as lists of coordinates.
Input	Void
Output	List of waypoints for vehicles.

The HT operation is performed by the HT manager by guiding the modeled CHEs. The calculation of routes is dependent on the initialization phase, as it provides many of the inputs of the operation functions.

FS 3: Run visualization interface.

FS 3.1	Gather data for visualization.
Description	The location and orientation data of the vehicles is gathered.
Input	Void
Output	Location and orientation data for all vehicles.

FS 3.2	Send visualization data to Unity.
Description	The gathered vehicle location data is transmitted to the visualization module via the HT manager.
Input	Vehicle location and orientation data.
Output	Data transmission to visualization module.

Visualization is performed by a provided visualization module. The modeled system tracks the location information of the modeled vehicles and relay it to the module. The module converts the information for Unity to utilize.

6.3 System design

The second phase in the systems engineering workflow, is system design. The purpose of this phase is to specify the software architecture, interface definitions, data structures, and functional logics in algorithmic detail [28].

6.3.1 Architecture and interfaces

The basic architecture of the implemented model is pictured in figure 17. The design is adapted from the large-scale terminal model. The top-most level is the operations simulation manager (OSM), followed by terminal operating system (TOS), horizontal transportation manager (HTM) and HTE. Visualization interface is embedded into the TOS model. These form the main hierarchical model.

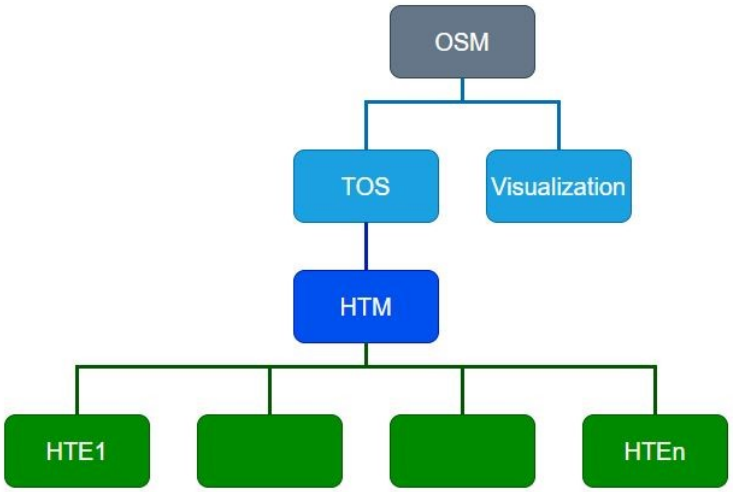


Figure 17. *Model architecture.*

OSM acts as an interface for the user to manipulate the simulation model. It reads a configuration XML document and converts the given simulation parameters into form compatible for Simulink. The XML document is user-configurable and can be altered to define multiple simulation scenarios. Yard areas, HTE and fixed objects are defined in OSM.

The second hierarchical layer is formed by TOS and visualization models. The visualization interface model carries out communications between Unity and Simulink using UDP. TOS handles the generation of containers and their assignation on container areas within the yard.

HTM handles the job creation, path planning and path assignment for the HTE on the yard. It also receives position data from each individual HTE and relays them for TOS for visualization. As part of the path planning, HTM gathers the container area data and

creates a lane system for the yard. This lane system is used to impose traffic laws within the yard, and all paths planned abide to these rules. The planned paths take into consideration obstacles within the area of operations such as containers and other equipment.

The lowest layer is the HTE, which in this thesis are SCs. These are the only models that represent physical entities. The SCs follow the paths created by the Horizontal Transportation Manager (HTM). They perform pick and ground jobs until no more jobs are provided for them by the HTM. The HTE model includes kinematics to represent realistic movement of the SCs. Location and orientation information for all vehicles is gathered and sent to the HTM concurrently.

6.3.2 Data structures and algorithms

To efficiently handle the large amount of data the model contains, careful planning for the data structures is required. Simulink supports the use of composite signals to transfer data between and within models. One such signal type is called a bus. Buses are name-based composite signals that support signal hierarchy. Buses are composed of elements, which retain their separate identities, and as such, can be extracted. The elements of a bus can be of any type, even another bus, which allows the use of nested buses. Buses reduce visual complexity in a model by combining various signals into one cohesive entity. Buses can be associated with a bus object, which specifies the architectural contents of the bus. These objects may contain information of the number and order of the elements, along with their datatypes and hierarchy. As they are used only to validate bus signals, they do not include data values. Any bus element linked to a bus object, that does not match the definition of that bus object, causes an error. [33,30]. The simulation model uses buses and bus object for most of the data transfer within and between sub-models.

The persistent information within the model is stored in a data dictionary. A data dictionary is a repository that stores design data, such as parameters and signals related to a model. Simulation data, such as inputs and outputs are not stored in the dictionary. [34]. Multiple models can be linked to the same data dictionary and can access the information stored in it. Container information from TOS is saved in a list, and jobs created from it are stored in multidimensional arrays. The yard's lane system is created with vectors representing each individual lane, and their intersection data is saved as buses. The path planning algorithms are using a graph-based notation and the calculated routes are sent to the vehicles as arrays. The graph nodes are stored as a list structures

for easy manipulation, but the actual route calculation is done with the use of an adjacency matrix.

Visualization and simulation parameter configuration utilizes Extensible Markup Language (XML) data type. XML is compatible with different systems due to its use of plain text format as data store.

6.4 Implementation

The next phase in the waterfall model is implementation. For the HT operations model, only HTM and HTE models pictured in figure 17 were modeled. The multiple HTE models of figure 16 are identical, although the architecture supports different types of equipment. OSM, TOS and Visualization models were given as a part of the terminal-scale model. This chapter briefly describes the function of the OSM, TOS and Visualization models. The implementation of HTM and HTE models is described in detail.

6.4.1 OSM, TOS and visualization

OSM reads the configuration parameters that the user defines in the XML document. The plain text information is transformed into data compatible for Simulink with the use of a MATLAB script. The document defines the number and type of HTE in the yard as well as their starting locations and the distinct yard areas. Although the yard has different container areas and area managers, such as STS and yard crane interchange areas, the corresponding equipment is not modeled or visualized. The yard managers are used to realistically portray the container pick and ground locations in the terminal operation. OSM generates any fixed objects on the yard, including yard and quay cranes' rails that must be taken into consideration when planning vehicle paths.

TOS is responsible for the generation of containers on the yard by assigning them on the areas OSM creates. The configuration parameters for containers contain the manager they are assigned to, the location of the container in that manager's area designated with lane, bay and tier, and any additional information, such as size.

Unity is used for visualization. During initialization of the simulation, Unity reads the information on fixed objects, containers, and equipment and creates them. Simulink and Unity communicate via UDP, with a single Simulink model responsible for this communication. The interface model receives necessary information from the other models and relays them for Unity. During container creation, TOS sends container information for visualization initialization. During HT operation, HTM gathers location and spreader data from the vehicles at every time step and sends them to the visualization

interface model. Figure 18 pictures two straddle carriers in operation on the terminal yard.

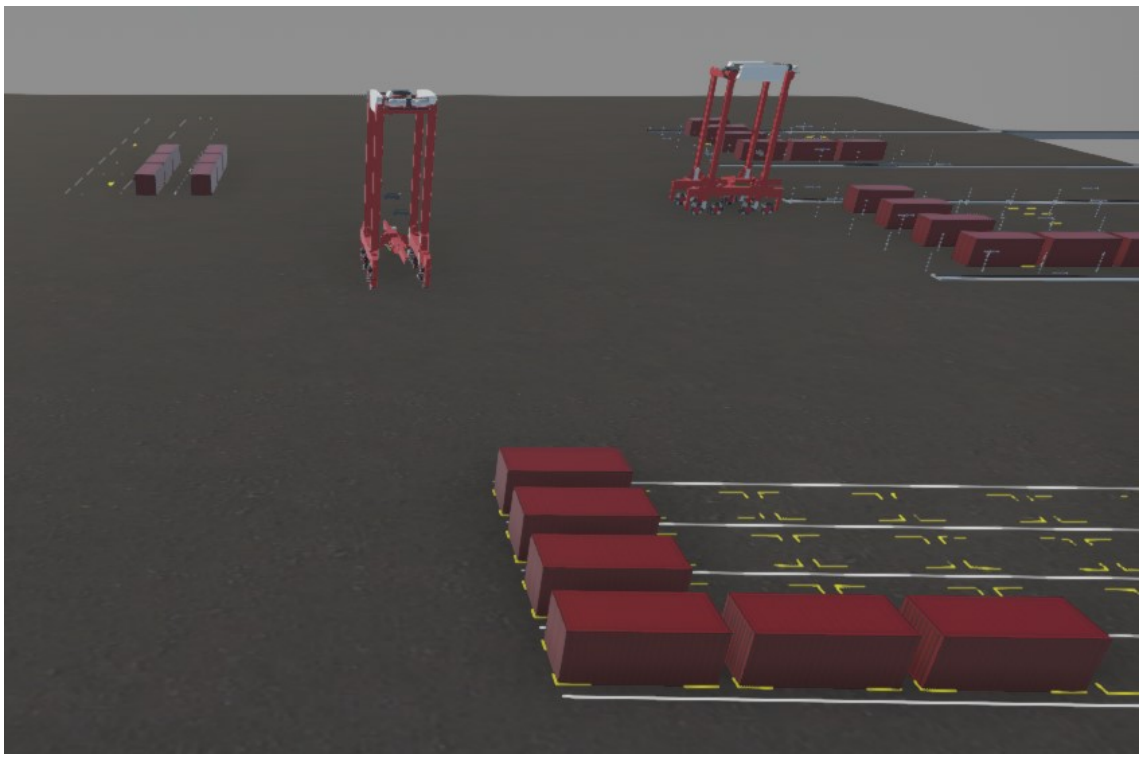
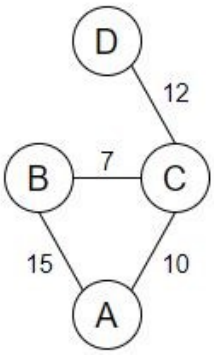


Figure 18. *Straddle carriers in operation.*

6.4.2 Horizontal transportation manager

During initialization of the simulation process, HTM prepares the container yard for path planning operations. OSM and TOS generate the frame for HT operations by creating the container locations and areas. HTM supplements this initialization by creating a network of lanes on the yard. The terminal yard needs to ensure safety of operations with traffic laws, which can be performed by the vehicles abiding to lanes. The lane network is created to connect different yard areas to each other. Lanes can be uni- or bidirectional, allowing for more detailed traffic rules should the need arise. In this thesis, the lanes are created as either vertical or horizontal line segments. The path planning operations of the HTM are graph-based, which means that the lane notation is not sufficient for the path calculation. The lanes' intersection points are calculated after lane initialization, and these values are saved. With the containers' positions and the lane intersections, the points of interests are formed for the graph. Intersections and container locations are the graph's nodes, and the rest of the generated lanes act as the graph's edges. Each node is given a name and coordinate data and is saved into a list of the nodes. The graph is declared as an adjacency matrix, to find adjacent nodes for each node, and the cost of travel between those nodes. This thesis uses the distance between

nodes as the cost, with no additional cost caused by terrain. An example of adjacency matrix is given in figure 19. The adjacency matrix is formed in from a graph by generating a NxN matrix, where N is the number of nodes in the graph. In this matrix the costs of the edges to adjacent nodes are saved on the corresponding row and column. In the example of Fig 19, Node A has two adjacent nodes: B and C. The cost of the edge from A to B is 15, and it is saved to the matrix in row A and column B. As the edge is bidirectional, the same cost is then saved to row B and column A. This process is done to each of the nodes in a graph. This form of data structure is very compact way of representing a graph.



	A	B	C	D
A	0	15	10	∞
B	15	0	7	∞
C	10	7	0	12
D	∞	∞	12	0

Figure 19. *Example graph (left) and corresponding adjacency matrix (right).*

The following example shows the workflow of the HTM initialization process, from the creation of containers and areas to the creation of the graph. The example terminal has four different yard areas and some containers located within these areas. The initialization information from OSM and TOS is pictured in figure 20.

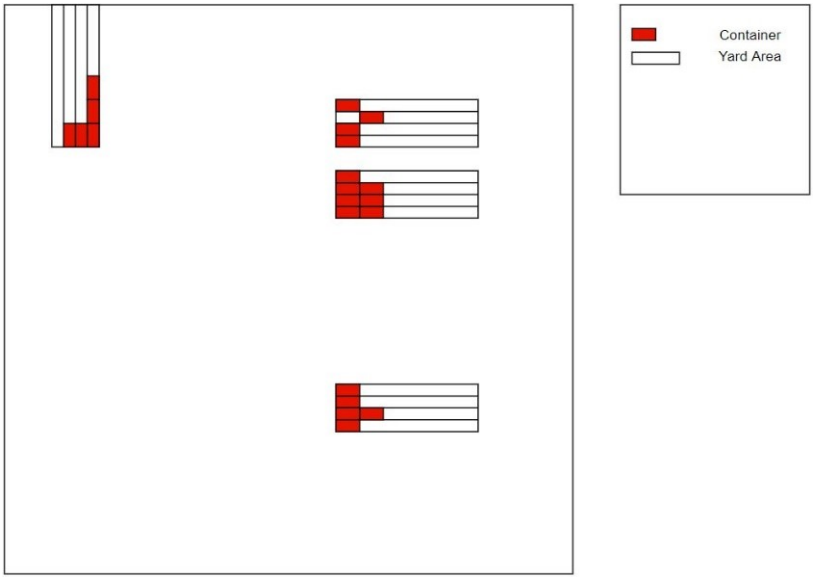


Figure 20. *Container and yard locations.*

HTM uses the coordinates of the yard areas to create a lane network connecting these areas. Yard areas themselves are composed of several lanes, so a corresponding number of lanes must be created. An example of a terminal lane layout consisting of vertical and horizontal lanes is presented in figure 21.

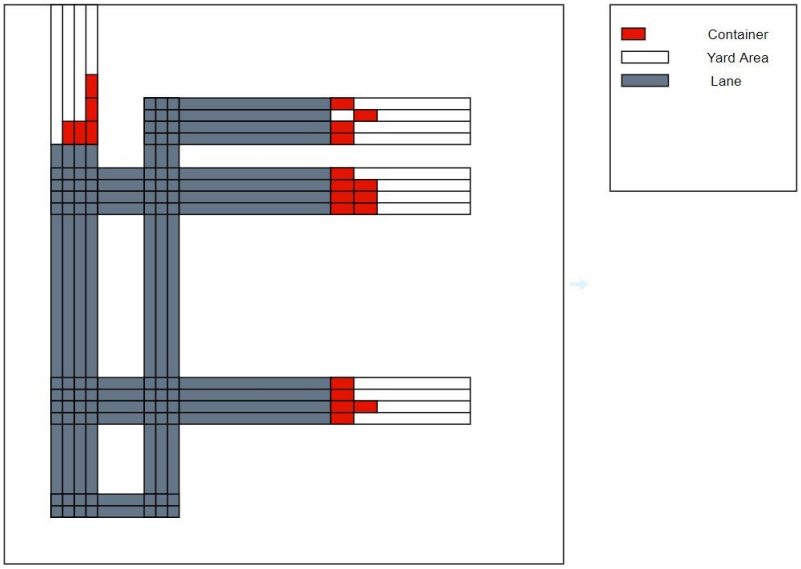


Figure 21. *Example of lane network on terminal yard.*

HTM then calculates the lanes' intersections to extract important information for graph generation. As all necessary points of interest are now gathered, HTM can begin the generation of the graph. Container locations and intersection points are recorded as nodes, and the lanes between them as edges. The lanes on the terminal yard are set as bidirectional edges, and no additional costs were implemented between each intersection node. Figure 22 pictures the lanes' intersection points and figure 23 shows the compiled graph notation of the example terminal.

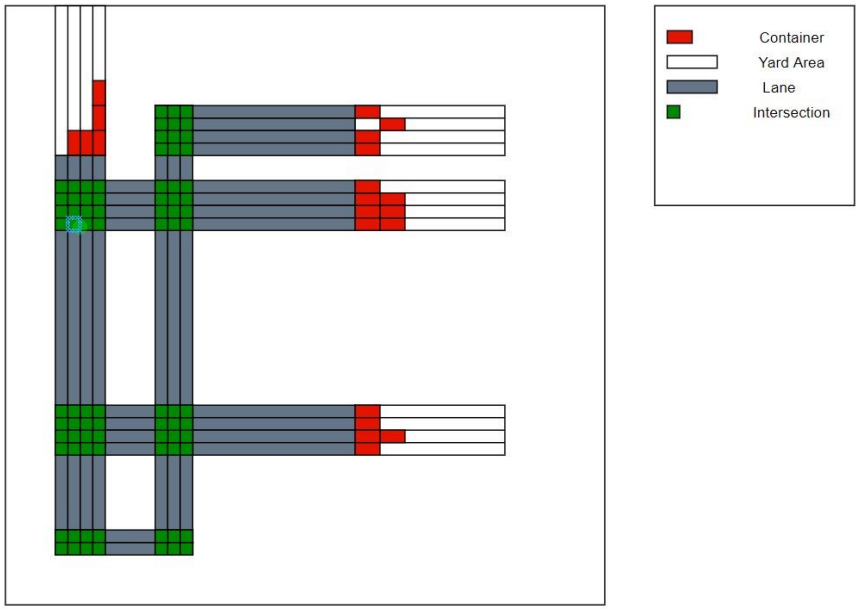


Figure 22. *Yard example with intersections*

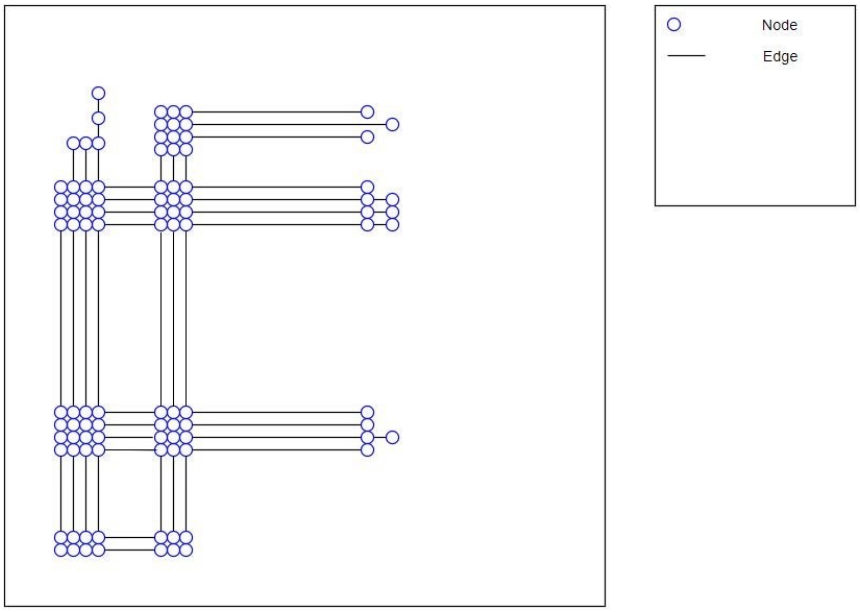


Figure 23. *Example yard graph notation.*

With the graph formed, the adjacency matrix like in Fig 19 can be calculated and all the necessary initialization for HTM operations is completed. After the initialization phase is done, the main operation of the HT manager begins.

HTM has real time information of each vehicle's location, and the locations of given pick and ground targets. The manager then creates a job, including the container to pick and its corresponding ground locations, and assigns it to the vehicle closest to the first target of the job sequence. HTM then begins the path planning for the assigned vehicle. The target location becomes the path's goal, and the vehicle's current position its start. A* path planning algorithm was chosen for this thesis, and the heuristic for it is the

Manhattan distance, as the yard's lanes are vertical and horizontal only. The manager plans the path as presented in chapter 4.2. For a single vehicle, the process is straightforward, and the calculated route can be sent to the vehicle. In multi-agent implementation, however, additional care must be taken. As multiple vehicles cannot occupy the same location at the same time, a collision detection is necessary. The algorithm compares the vehicles' paths and calculates in which time-step they occupy which node. If two or more vehicles are going to be in the same position at the same time, priority should be given to one, and the other either finds alternative route, or waits until no collision is imminent.

The routes formed are stored as a list of coordinates in the desired driving order consisting of a maximum of 100 waypoints, as the routes in this thesis are all under 100 graph nodes. The route-list is multidimensional, so each vehicle can be given its route at any given time. The routes are sent to the vehicles one waypoint per time step, which are added to a waypoint buffer. This buffer is further explained in chapter 6.4.3. At each time step, HTM also receives location data from each active vehicle, this is used for path planning as discussed above, and visualization.

6.4.3 Straddle carrier

A straddle carrier was modeled for the HTE in this thesis. The model consists of two main sub-models: a simple cyclic operations logic and the kinematic model of a straddle carrier. The vehicle model includes calculation models in addition to the main vehicle kinematics model. These calculation models calculate the required inputs for vehicle operation from the control logic's outputs. The operations logic is formed by a buffer mentioned in the previous chapter, and a state machine with the main states of the operation. The buffer receives waypoint data from HTM at each time step and stores it, so the waypoints do not overwrite each other. The vehicle can reset the buffer if any new orders it gets have different job identification to its previous one. This allows the vehicles to terminate any previous orders if HTM so determines. The operational states determine the inputs given to the kinematic model. The operation logic has three outputs: acceleration, next waypoint, and direction. The direction input has two values: 1 for movement in the positive direction of the vehicle's longitudinal axis, and -1 for movement in the negative direction. This value determines the direction of acceleration necessary to traverse the vehicle to the desired coordinates. The directional input also determines the steering angle calculation for reverse driving. The acceleration output depends on the operational state: "driving" -state outputs nonzero acceleration, while "idle" or

“hoisting” -states output zero. The next waypoint input is a vector of coordinates in x,y -dimensions, and with the vehicle’s current location, the necessary steering angle can be calculated.

The kinematic model of the straddle carrier follows the principles stated in chapter 5.2.1 and is simplified to a bicycle model. The bicycle model is controlled by two inputs: velocity and steering angle. The other values required for the vehicle’s model, such as turning radius and the length of the wheelbase are constant. These were taken from a Kalmar Straddle Carrier technical data brochure for accurate representation of SC operation. The dimensions of the vehicle can be easily modified should the type of HTE be changed for simulation. Picking and grounding operation was modelled with a time delay, as they are not instantaneous. Secure locking and unlocking to the container are simulated to increase realism of the model. The vehicle drives according to the waypoints it extracts from the buffer, and when it detects that the waypoint is the final waypoint in that job, the necessary hoisting operation begins. The vehicle continues the hoisting operations until it receives no new waypoints from the HTM, in which case the vehicle enters an idle state. The vehicles themselves have no higher operating logic, so they will remain idle until more routes are sent to them.

7. VERIFICATION OF THE MODEL

The modeled system must be tested to verify its correspondence to the requirements. In the context of this thesis, the validity of the system is determined how well it fulfills the user requirements presented in chapter 6.2. The implemented model is compared to a real-world system found in literature to determine the realism of the models. The most time-consuming part of the simulation is the path planning operation. To determine if the model can be run multiple times faster than real-time, different lengths of paths are planned for both a single vehicle and multiple vehicles. Table 4 provides the technical specifications of the test computer.

Table 4: *Technical specifications of the test computer*

CPU	AMD Ryzen 5 3500U 2.1 GHz
Memory	8.0 Gb RAM
GPU	Radeon Vega Mobile Gfx
Operating system	Microsoft Windows 10 Home

In the simulation testing, the HTE are all SCs and identical in operation. The number of operational vehicles in the yard can be altered by simply copying and pasting the required models and interfaces. Complete collision avoidance is not possible with the implemented model. The HTM creates a space-time graph from the calculated routes to determine in which time-steps which vehicles are in which positions. This is done by calculating the expected velocities along the planned route to represent the movement of the vehicles. If the manager detects two or more vehicles occupying same node at the same time, it displays information about the collision. Although the simulation model cannot avoid collisions at this stage, the collision information is gathered for verification. The simulation model considers the driving in the yard, hoisting operations and collision detection. Delays caused by machine failures, human error, container weight or weather conditions are not considered.

In chapter 7.1, the customizable parameters of the simulation model are set to correspond values found in literature. In chapter 7.2 the simulation model is validated with studies found in literature. In chapter 7.3 multi-agent path-planning operation is simulated with varying number of vehicles and containers.

7.1 Parametrization

To get comparable simulation data, the model's parameters need to be tuned to represent corresponding real-life systems. The modeled SCs are modeled after Kalmar Straddle Carriers [14]. Table 5 presents the selected values.

Table 5: *Kinematic parameters of the simulated SCs*

Turning radius	6150	mm
Travel speed	8.3	m/s
Wheelbase length	9200	mm

Turning radius is the calculated average of inner and outer turning radius of an SC due to the use of the bicycle model. As explained in chapter 5.3, the wheelbase and the turning radius of the vehicle can be used to calculate the maximum steering angle. As the vehicles are bidirectional, the function for calculating the “negative” direction is implemented here. In the case of unidirectional HTEs, this function can be left out.

As the vehicles are controlled with an acceleration input, the differentiated velocity values must be limited. The travel speed is the maximum velocity the simulated vehicles can travel while driving in the yard. The maximum velocity is the same for both “positive” and “negative” directions. When approaching their assigned containers, the velocity of the simulated SCs is inverse to the distance between the SC and the container. This simulates the operator slowing down and carefully positioning the vehicle above the container.

The hoisting operation for SCs in real life consists of lowering the spreader, locking it to, and securing the container, and hoisting the container. Both picking and grounding operations have similar work cycle. Hoisting is simulated in the model with a time delay to compile all the actions done in a hoist cycle. According to literature, the time spent in one hoist cycle is approximately 20 seconds on average, for both picking and grounding containers [8]. This time is used as the delay in the hoisting function of the model.

7.2 Verification

After parametrization, the HT operations model is verified. The implemented model is compared to similar systems found in literature to determine its accuracy. The model is verified by analyzing both the work cycle of a single SC, and the path-planning capability of the HT manager.

The work cycle of a SC can be divided to travelling to a pick-up location, hoisting the target container, traveling to drop-off location, and grounding the container. This work cycle includes two separate jobs: picking and grounding. All other terminal yard cranes can be excluded from the work cycle as the modeled vehicles can perform every necessary operation.

The HT manager allocates these jobs for the SCs in operation and calculates the shortest paths to them. Path planning is done with A*-algorithm, with the heuristic in Manhattan distance. The terminal yard in the simulation is flat surface where the SCs can drive

according to their maximum velocity. The flat environment means that no other costs were added to the yard's graph edges, so the planned paths are chosen by distance traveled. The calculation time required for path planning for a single SC was consistently under a thousandth of a second, which bodes well for multiple times faster than real-time simulation.

The simulation scenario for the work cycle analysis has a single SC working in a terminal yard with 4 distinct yard areas, with 6 containers in each area. The yard areas are in a configuration like the one depicted in figure 20. Jobs are created to realistically portray the transportation of containers from different cranes' areas to the stacking yard and vice versa. According to the simulations, the overall time to complete all jobs was approximately 3500 seconds. The validity of these numbers is hard to verify with studies found online, unless the referenced yard sizes are specified. The result however can be used as a baseline result for later simulation scenarios.

When analyzing terminal yard productivity, the performance of a single crane is often measured in moves per hour (mph) [5]. This refers to boxes or TEUs moved in an hour. A one move encompasses both picking and grounding jobs completed by the crane. The productivity of a straddle carrier depends on yard layouts, areas of operation, and other possible factors. The lowest approximation of productivity was 10 mph per straddle carrier [32], and the highest 36 mph per straddle carrier [8]. With these and approximate productivities from Hangga and Shinoda [9] and Legato and Mazza [20], the average productivity per straddle carrier is approximately 25 mph. With the baseline simulation's productivity of 24.7 mph, this simulation model seems to be realistic enough for more complex simulations.

7.3 Multi-agent path planning simulation

To study the impact of the number of SCs to the efficiency of the HT operations, simulations with varying number of vehicles and containers is required. Due to the lack of completely working collision detection, the performance of multiple vehicles is based on the elapsed time to complete a set of jobs, and how many collisions were detected during operation. The terminal yard in the simulation scenario became easily cramped when operated by multiple vehicles, the maximum number of SCs simulated was 16. Table 6 presents the simulation results.

Table 6: *Simulation results of HT operation with varying number of SCs.*

Number of SCs	**1**	**2**	**4**	**8**	**16**
Average productivity (mph)	24.2	23.3	22.5	21.9	17.7
Average number of collisions	0	27	66	117	330
Average path planning time (s)	0.0005	0,0013	0.0025	0.0071	0.0110

Simulations were conducted on the same yard with the same configuration of areas, with varying number of containers to get average values for verification. As table 6 shows, the average productivity of HT operations per crane is 24.2 mph when the number of cranes is one, while the productivity starts to decrease when additional vehicles are operating. The number of collisions detected in operation unsurprisingly increases with the number of operational SCs. The simulation results show that the number of collisions in multi-agent operations doubles with the number of active SCs in the yard. Although with 16 active SCs, the number of collisions almost triples. The average time for route planning increases with the number of SCs, as the algorithm needs to detect collisions by referencing each existing path when calculating new ones.

The decrease in productivity when operating with multiple vehicles can be due to the job assigning algorithm. The system in this thesis is far from optimal when operating multi-agent actions. Jobs consist of a pick, and a ground location, to which paths are planned in succession. When any vehicle is given a new job with a new pick location, the closest unmoved container is assigned. This works well enough for a single SC as the new jobs are reliably found nearby, but when adding operational vehicles, the nearest free container may not be the most efficient choice in the long term. The model gets the locations of the containers from the terminal-scale model, while the grounding locations for each container are generated by the HTM. The grounding locations are selected from a pool of possible locations and assigned to each container. As the jobs are assigned based on proximity, and the grounding locations are assigned from a pool, the distribution of the jobs in the yard, and among the SCs might be unbalanced, which in turn affects the per SC productivity.

The number of collisions also goes up, with the number of operating vehicles, as expected. The simulation scenario yard can prove to be quite cramped, as all vehicles are assigned the optimal route, regardless of obstacles. With a complete collision avoidance, the detected collisions could be avoided, either by rerouting vehicles, or by assigning wait time. Rerouting increases the time used for path planning, and with increasing number of vehicles, could be inefficient as more and more paths need to be rerouted. Adding wait times for operating vehicles with a priority system does not require much time for path planning, but it increases idle times when the number of vehicles increases, which decreases the overall productivity. The job assigning logic mentioned

above can also increase the number of potential collisions, as the assigned jobs are not necessarily optimal as more and more vehicles are active.

When the modeled HTM assigns a new job for a vehicle, it plans a path in its destination, for a single SC, the time is minimal, under a thousandth of a second. This, however, changes as vehicles are added to operation. The manager plans a path for the vehicle in question, but also compares that route to all existing routes for other vehicles. In this thesis, collisions are only detected, but the effect of calculation time is apparent with additional vehicles. As table 6 shows, the average time spent planning paths increases with the number of SCs. The change is quite manageable, with the path-planning time not falling under a hundredth of a second until 16 vehicles. As stated above, the full collision detection and avoidance could be done with two different approaches. Rerouting low priority routes with less optimal nodes would be the approach that affects the calculation time the most.

Based on the simulations, the implemented model can represent limited multi-agent path planning operations in a horizontal transportation setting. The produced estimates for productivity per straddle carrier are similar to those of studies found online. The lack of a complete collision detection and avoidance system and a more sophisticated job assigning algorithm, however, limits the capabilities of the model's use as a comprehensive productivity simulator for HT operations. The simulation model's performance was tested by comparing the total elapsed time (in real-time) and the last time-step of the simulation time provided by Simulink. With the test computer setup described in table 4, simulations can be run approximately 2 times faster than real-time. Without visualization, the simulations can be run 3 times faster than real-time. This leaves still much room for improvement in the used data structures and data manipulation.

8. CONCLUSIONS

In this thesis, a simulation model of HT operations was implemented with MATLAB Simulink. The model was visualized with Unity. The model was done to impose and analyze traffic control for terminal yard for different quantities of HTE. The model incorporates picking and grounding jobs in the terminal quayside and stacking yard. The model's equipment is modelled as straddle carriers capable of performing picking, grounding, and stacking operations without other yard cranes. The model was analyzed in terms of yard productivity and software performance. Productivity was evaluated by containers moved per hour and collisions detected, and software performance by time spent on path-planning.

How to design and implement a HT simulation model for traffic control in terminal environment?

The model was implemented according to a waterfall model often found in software development. Requirements for the model were gathered, and with them the functional specifications drafted (see chapter 6.2). Many requirements were dictated by a terminal-scale simulation model, into which the implemented HT simulation model can be integrated. This leads to the implemented model's architecture and features mimicking those of the terminal-scale model. The implemented model consists of two modules: a horizontal transportation manager, and horizontal transportation equipment. The manager is an event-driven control system, that handles job management, traffic rules and path-planning for the equipment. The HTE in this thesis are SCs, which are modeled with time-driven kinematics model that is controlled by event-driven operation logic. The number of SCs in operation, the number of containers, and the configuration of yard areas and interconnecting lanes is modifiable.

The implemented simulation model was verified to answer the gathered requirements. Verification was done by simulating one SC operation with a limited number of containers. Configurable operational and kinematic parameters of the SC were tuned to match values found in real-life systems to compare the implemented system's realism. The verification simulations' results show that the model performed reasonably realistically. The model also consistently calculated different lengths of paths in under a thousandth of a second, which was a good indicator for the simulator's capability for multiple times faster than real-time operation.

After the model was verified, the effect of altering the number of operational vehicles was studied. Simulations were performed with varying number of containers and SCs to evaluate average productivity, collision detection, and path-planning time. Operating the terminal yard with a low number of SCs lead to high per crane productivity, while keeping the number of collisions relatively low. Adding operational vehicles to the yard slightly lowered the average per crane performance and quickly increased the number of collisions. In simulations with 16 SCs on the yard, the average productivity decreased noticeably, and the vehicles collided even more frequently. The manager's path-planning capability was evaluated by analyzing the average time it took to plan different lengths of paths for multiple operational vehicles. The path-planning operation consisted of calculating the shortest path with A*-algorithm according to Manhattan distance, and the collision detection for all operating vehicles. The average path-planning time was consistently under a hundredth of a second for all number of SCs except 16. This shows that even without a complete collision detection and avoidance the calculation time accumulates when dealing with a large number of operating vehicles.

Which features are needed for the system to be used for realistic simulation?

Should the simulation model be used for realistic simulation, some features need to be incorporated. The kinematics of the modeled CHE need to be parametrized according to a real-life system, as is the case for the SCs in this thesis. However, the models of this thesis do not take into consideration dynamic conditions on the terminal yard. Instead, only the optimal circumstances are implemented, where no complications are caused by inclement weather, human error, or other hindrances. The operational logic of the HTM should also reflect existing systems for it to have realistically comparable results. The path-planning operation should be able to function with complete collision avoidance, without manual interference.

Which features are needed for the system to be used as an optimization tool?

The simulation model should be highly configurable to be able to be used for HT in any terminal layout. As discussed above, the simulation model should be realistic enough to provide comparable results for each simulation scenario. For the model to be used for optimization purposes, the level of realism is dependent on each use-case for the model. Along with the implemented model's lack of collision avoidance, it also was flawed in another way. The job sequencing algorithm was not optimal, as it assigned containers to vehicles based on proximity alone. A more sophisticated algorithm for assigning jobs would have likely decreased the number of collisions on the yard, as the job sequence could be planned to distribute SCs more evenly across the yard areas. The simplicity of

the job sequencing algorithm was deliberate choice in this thesis, as more focus was given to the kinematic modeling of the SCs, and the HTM's path-planning capability.

Overall, the implemented simulator can realistically enough portray HT operations in a terminal yard setting with a few limitations. The model fulfills the set requirements with multi-agent pathfinding capabilities for realistically modeled SCs, using the same architectural choices as the terminal-scale model, and faster than real-time operation. The simulations, however, are not completely realistic and optimal as the model lacks some features required for a proper productivity simulations model. The implemented model can be upgraded to a more sophisticated, and more thorough model by implementing the missing features.

8.1 Future research

To enhance the model's usability as a proper optimization tool, some features must be implemented. First and foremost, an extensive collision avoidance system, that can detect collisions, and either reroute or impose idle times for the operating vehicles. The choice of collision avoidance would be dependent of the number of vehicles in the yard, as well as the yard's area and lane layout. Too many rerouting operations done to a large number of vehicles inflates the computation time, while making the vehicles wait could cause traffic jams, where every vehicle waits for every other vehicle to get out of the way. The second feature is a more sophisticated job sequencing algorithm, that works with the path-planning logic to optimally distribute the jobs among the operating vehicles to ensure minimal collisions and better productivity. A better algorithm could be achieved by utilizing optimization models found in literature. The additional features working in tandem would enable a more in-depth analysis for HT productivity simulation.

The model currently uses SCs as the HT equipment. Fully integrable model would need to incorporate other forms of HT to be used in any situation and with any configuration of terminal equipment. Handover operations for yard areas require different procedures for different combinations of CHE, and the model needs to be enhanced accordingly.

REFERENCES

[1] E.T.S. Alotaibi, H. Al-Rawi A complete multi-robot path-planning algorithm. Auton Agent Multi-Agent Syst 32, 693–740 (2018). https://doi.org/10.1007/s10458-018-9391-2

[2] H. Arioui, L. Nehaoua, Driving Simulation, John Wiley & Sons, Incorporated, 2013.

[3] R. Asariotis, M. Assaf, H. Benamara, J. Hoffmann, A. Premti, L. Rodríguez, M. Weller, F. Youssef, Review of Maritime Transport, United Nations Publications, United conference on trade and development, 2018, Available (accessed 12.06.2020).

[4] D. M. Buede, W. D. Miller, The Engineering Design of Systems: Models and Methods, John Wiley & Sons, Incorporated, 2016

[5] J.W. Böse, Handbook of terminal planning, Springer, Vol. 49, 2011, 433 p.

[6] C.G. Cassandras, S. Lafortune, Introduction to discrete event systems, Springer, 2008, 769 p.

[7] S. A. Fadzli, S. I. Abdulkadir, M. Makhtar and A. A. Jamal, "Robotic Indoor Path Planning Using Dijkstra's Algorithm with Multi-Layer Dictionaries," 2015 2nd International Conference on Information Science and Security (ICISS), 2015, pp. 1-4, doi: 10.1109/ICISSEC.2015.7371031.

[8] P. Hangga, T. Shinoda, Big Data Utilization and Analysis to Improve Operational Performance of Hybrid Straddle Carrier in Container Terminal, Asian Transport Studies, 2018-2019, Volume 5, Issue 4, Pages 679-693, Released September 01, 2019, https://doi.org/10.11175/eastsats.5.679

[9] P. Hangga, T. Shinoda, A Petri Net Model and its Simulation for Straddle Carrier Direct-System Operation in a Container Terminal, Applied Mechanics and Materials, 2017, vol. 862, pp. 202-207.

[10] S. Hartmann, Generating Scenarios for Simulation and Optimization of Container Terminal Logistics. OR Spectrum, 03, vol. 26, no. 2, 2004, pp. 171-192 Business Premium Collection; ProQuest Central; Technology Collection. ISSN 01716468. DOI http://dx.doi.org.libproxy.tuni.fi/10.1007/s00291-003-0150-6.

[11] Kalmarglobal.com, Automated guided vehicle, Available: https://www.kalmarglobal.com/equipment-services/automated-guided-vehicles/

[12] Kalmarglobal.com, Essential reachstacker, Available: https://www.kalmarglobal.com/equipment-services/reachstackers/essential-reachstacker/

[13] Kalmarglobal.com, Shuttle carrier, Available: https://www.kalmarglobal.com/equipment-services/shuttle-carriers/diesel-electric/

[14] Kalmarglobal.com, Straddle carrier, Available: https://www.kalmarglobal.com/equipment-services/straddle-carriers/diesel-electric/

[15] Kalmarglobal.com, Terminal tractors, Available: https://www.kalmarglobal.com/equipment-services/terminal-tractors/TL2-Essential-Terminal-Tractor/

[16] L.E. Kavraki, S.M LaValle, Springer Handbook of Robotics, Springer, 2nd Edition, 2016 pp 139 – 162.

[17] G. Klancar, A. Zdesar, S. Blazic, I. Skrjanc, Wheeled Mobile Robotics, Butterworth-Heinemann, 2017.

[18] D. Kress, S. Meiswinkel, E. Pesch, Straddle carrier routing at seaport container terminals in the presence of short term quay crane buffer areas, European Journal of Operational Research, Volume 279, Issue 3, 2019, Pages 732-750, https://doi.org/10.1016/j.ejor.2019.06.028.

[19] J. Latombe, Robot Motion Planning, Springer, 1991.

[20] P. Legato and R. M. Mazza, "A simulation model for designing straddle carrier-based container terminals," 2017 Winter Simulation Conference (WSC), Las Vegas, NV, USA, 2017, pp. 3138-3149.

[21] W.S. Levine, The Control Handbook, CRC Press, 1996, 1548 p

[22] MacGregor Container securing systems product catalogue, nd, Available: https://www.macgregor.com/globalassets/picturepark/imported-assets/65120.pdf

[23] Maritime container terminal, nd, accessed 19.05.2020, Available: https://www.container-transportation.com/container-terminal.html

[24] E. Martín, P. Morales-Fusco, S. Saurí, Comparing Manned and Automated Horizontal Handling Equipment at Container Terminals. A Productivity and Economical Analysis. Transportation Research Record Journal of the Transportation Research Board, 2013.

[25] F. Meisel, Seaside Operations Planning in Container Terminals, Springer, 2009

[26] T. Notteboom, The Time Factor in Liner Shipping Services, Palgrave Macmillan, Maritime Economics & Logistics, Vol. 8, Iss. 1, 2006, pp. 19-39.

[27] G. Olsson, G. Piani, Computer Systems for Automation and Control, Prentice Hall, 1992, 428 p

[28] T.Ruuska, Vaatimusmäärittely ketterässä ohjelmistokehityksessä, Master's Thesis, University of Jyväskylä, 2012, 78 p.

[29] Simulink, MathWorks, Web page, Available (accessed 10.06.2020): https://www.mathworks.com/help/simulink/

[30] Specify Bus Properties with Simulink.Bus Objects, MathWorks, Web page, Available (accessed 8.2.2021): https://se.mathworks.com/help/simulink/ug/when-to-use-bus-objects.html

[31] Stateflow, MathWorks, Web page, Available (accessed 10.06.2020): https://www.mathworks.com/products/stateflow.html.

[32] C. A. Thoresen, Port Designer's Handbook - Recommendations and Guidelines. ICE Publishing, 2003.

[33] Types of Composite Signals, MathWorks, Web page, Available (accessed 8.2.2021): https://it.mathworks.com/help/simulink/ug/composite-signal-techniques.html

[34] What Is a Data Dictionary, MathWorks, Web page, Available (accessed 8.2.2021): https://se.mathworks.com/help/simulink/ug/what-is-a-data-dictionary.html

[35] XML Tree, w3schools, Web page, Available (accessed 8.2.2021): https://www.w3schools.com/xml/xml_tree.asp