Proceedings of the 2010 Winter Simulation Conference
B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, eds.

AUTOMATED 3D-MOTION PLANNING FOR RAMPS AND STAIRS IN INTRA-LOGISTICS MATERIAL FLOW SIMULATIONS

Matthias Fischer
Hendrik Renken
Christoph Laroque
Wilhelm Dangelmaier

Heinz Nixdorf Institute, University of Paderborn
Fürstenallee 11
Paderborn, 33102, GERMANY

Guido Schaumann

McAfee GmbH
Vattmannstr. 3
Paderborn, 33100, GERMANY

ABSTRACT

Commercial software of material flow simulations has the ability to layout the simulated models. Arranged equipment, such as conveyors or machines, includes the need to model and determine motion paths for moving objects like forklifts or automatically guided vehicles, so that the simulation framework is able to navigate all vehicles across those motion paths. After analyzing first scenarios, the user often carries out layout changes in the simulation model, e.g. moving, adding or deleting equipment. However, those changes cause time consuming, additional modeling of the motion paths for the user. Our motion planning algorithm reduces these changes by automatically determining the motion paths for moving objects, depending on an actual model layout without colliding with other objects. The algorithm works on the basis of the virtual scene's 3D-data used for the simulation model's visualization. We demonstrate the technique with a multi-floor building example.

1 INTRODUCTION

In the global competition "change simply happens" and therefore sustainable planning and flexible adaptation of all processes close to production poses a continuous challenge for companies. Innovative products as well as their corresponding manufacturing processes are to be reviewed at regular intervals in order to improve their efficiency and productivity. One well established method in the area of designing and safeguarding those production processes is the material flow simulation, where multiple scenarios can be modeled, simulated and evaluated before the factory's actual construction. During the last years, in this area, the trend increasingly followed the idea of a "digital factory" (VDI 2006), which covers among other disciplines the simulation and 3D-visualization of the designed processes (Dangelmaier and Laroque 2007). More often, in this research focus, multiple disciplines and methods close to the "basic" simulation, which are also used for the design of production or logistic processes, are connected and combined to simulation systems, in order to improve the gain of knowledge for a specific domain problem. In the area of the material flow simulation, these enhancements might cover modeling and simulation of multiple production systems in a supply chain (Gan et al. 2000; Holweg and Bicheno 2002) or the simulation of special logistic cells, e.g. in the construction industry (Rossmann et al. 2009).

Since the results of a special simulation scenario should lead to a higher understanding of the underlying system behavior, the generated experience of simulation results leads to improved system layouts and parameterizations of the simulation model as well as the designed production processes. Nowadays, in all commercial tools used for the material flow simulation, layout changes lead to a high workload of changes regarding the modeling and design of intra-logistic transport networks, e.g., used for the planning and dimensioning of forklift capacities and availabilities. Here, the automated generation of routing and motion paths for automated guided vehicles or forklifts reduces the modeling work by a simulation expert significantly, since an initial layout of possible paths is generated automatically via motion planning algorithms, based on the underlying layout data (Mahajan et al. 2005). Here again, for a specific problem, in this case the planning and safeguarding of intra-logistics, additional methods enrich the material flow simulation method, meaning the discrete event-based simulation, in order to improve the achieved simulation results in their level of detail, the gain of knowledge of the simulation expert, and the reduction of modeling effort.

Typically, today's design and production of new products doesn't lead to a "green field" design or a totally new layout of a new production plant, but existing processes have to be adapted and layouts within an existing facility are changed in order to be able to clear space for new production equipment or replace existing production lines. In some cases, this leads to complex plant structures, where goods are produced at different ground levels, connecting different halls or levels via ramps. In some cases production and storage are separated on different floor levels. Using existing solutions of such a motion planning algorithm (Mahajan et al. 2005), the simulation expert might be confronted with the weaknesses of the existing approaches, because the calculation of possible routes is based on a 2D-reduction of the factory layout and needs a high amount of calculation time. With such an approach a dynamic adoption of the plant layout, such as the forklift objects with or without transported goods or the modeling of pallet inventory in storage areas or loading platforms cannot be realized in real-time.

Here, the presented work enlarges the existing 2D approach to an automated 3D-motion planning algorithm, where these weaknesses can be solved. The automated calculation of motion paths over different floor levels is realized as well as the possibility of a dynamic enlargement of the underlying factory layout or the objects used for the intra-logistic transports. This allows this technique's use, not only during a specific scenario's simulation within a simulation experiment, but also within an interactive design process during modeling and parameterization. Forklift types can be changed as well as layout routes, regarding multiple floor levels or the integration of dynamic objects like trucks on "inbound" or "outbound" loading platforms.

Therefore, in a next step, the related work in this area within the material flow simulation as well as existing motion planning algorithm is covered in the next section. Afterwards, the main research goals of the presented solution are introduced as well as the basic method and algorithm, their application to an example factory and integration within a material flow simulation framework. The paper closes with an additional outlook on upcoming work.

2 RELATED WORK - MOTION PLANNING IN SIMULATION ENVIRONMENTS

In the modeling of moving objects, different modeling approaches are imaginable. Assuming a material flow, where goods enter at a specific point A and are carried by a forklift to a specific point B, where they will be processed. The modeling of connections between point A and B can be accomplished differently. In common simulation tools like Siemens' PlantSimulation (2010), Delmia's Quest (2010), Incontrol's Enterprise Dynamics (2010) etc., the transport object is using the direct way of transfer between these two points. If the distance between these points changes, their carrying time varies as well. Nevertheless, the underlying factory layout is not respected, e.g. walls or machines. In most tools, the element's motion between two points has to be modeled by the modeler using "non-flexible" paths, in order to be able to take the factory layout in the forklift's paths into account. The more complex the intended model, the more alternative paths for moving elements have to be modeled. Moreover, this approach fails, if the underlying layout changes due to planning advantages. The simulation expert has to change each possible motion path of a moving object based on the new layout.

In order to reduce the simulation expert's workload for modeling each path in a complex model, layout-based motion planning algorithms may automatically find motion paths for moving elements (La-

Valle 2006). Moreover, they are able to adopt the identified paths, if the layout changes. Fischer et al. (2005) and Mahajan et al. (2005) proposed a combined motion planning algorithm within a material flow simulation. The system is able to disburden the modeler from modeling motion paths for the transport objects. Autonomous vehicles like forklifts, etc. can be used within the simulation system, only by defining the start and end position in the underlying factory layout. The paths themselves are computed automatically by the system. Constrained objects like packets moving over a conveyer belt can be animated by defining an object's paths during the design of the building block "conveyor belt". The implemented system supports the motion of autonomous and constrained objects.

The motion planning algorithm consists of three steps as shown in Fischer et al. (2005). In the preprocessing and after changes in the layout of the scene, an object specific phase is performed to create a 2-dimensional outline from a 3-dimensional model. The second phase creates a scene specific graph representing possible paths for the moving objects. During runtime, this graph is used to search for a path from the current position of an object to its destination.

Recently Torchelsen et al. (2010) presented an approach for multi-agent path planning on arbitrary 3D surfaces. They use hierarchical computation of geodesic distances to define scalar fields whose gradient smoothly guides the objects across the surface. Their method can deal with meshes of arbitrary genus and curvature.

3 OVERVIEW

Our research goal and contribution of this paper is a methodology to reduce these time consuming, manual changes. We present a motion planning algorithm, which automatically determines the motion paths for moving objects, depending on the actual model layout without colliding with other objects of the virtual factory. The novel and particular feature of our method is, that it works, not only for a factory with one ground level, but moreover also for production or logistics facilities with multiple levels.

The designed motion planning (MP) algorithm works in two steps: In the first step, we process the 3D scene data of the simulation model and compute a data structure used for the automatic computation of possible motion paths. Each time the user modifies the simulation model, the processing of the static scene data is recomputed and updated, since the 3D layout might have also changed. This preprocessing step – called *MP scene processing* – is executed between modeling and simulation (see Figure 1). The MP scene processing and computation of motion paths is described in Section 4.

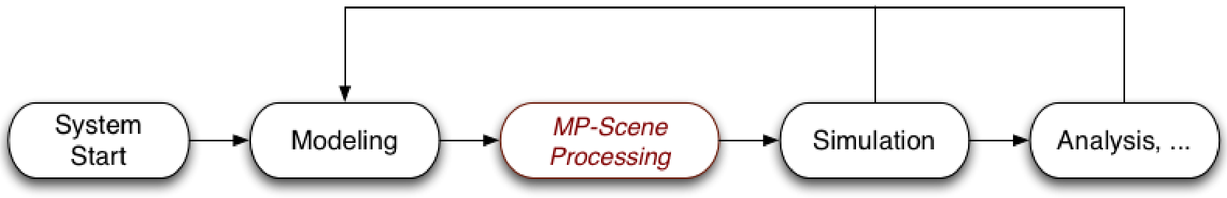

Figure 1: Workflow of modeling, simulation and analysis.

In the second step, the system computes motion paths for moving objects. They are computed on the fly during the simulation run, whenever the simulation needs to move objects, e.g. vehicles or avatars. Then the system dynamically computes a path for the object. The algorithm for this path computation is described in Section 4.3, its integration in the simulator, in Section 5. We evaluate our motion planning algorithm in Section 6.

We demonstrate the application of our motion planning algorithm by using an example of a multi-floor building. Our three-floor building is a facility, that produces some kind of goods (see **Figure 2**). Those goods are manufactured by robots, stored in a temporary storage, packed and bundled in a packing station. In the end, the finalized goods are delivered via trucks. We can show, that the algorithm handles 3D scenes of simulation models consisting of multiple floors, each connected with inclined ramps and stairs. Our algorithm automatically finds paths for the movement of vehicles across ramps and for the movement of persons' avatars across stairs. We report on:

- the movement of vehicles with start and end position on different floors,
- the collision prevention for overhanging objects,
- the computation of different kinds of driving surfaces used for the motion planning of paths,
- loading of trucks,
- and the computation of surfaces for moving packets on conveyor belts.

For demonstration of our motion planning algorithm in action with the simulation model shown in **Figure 2**, a movie is available (Fischer et al. 2010).

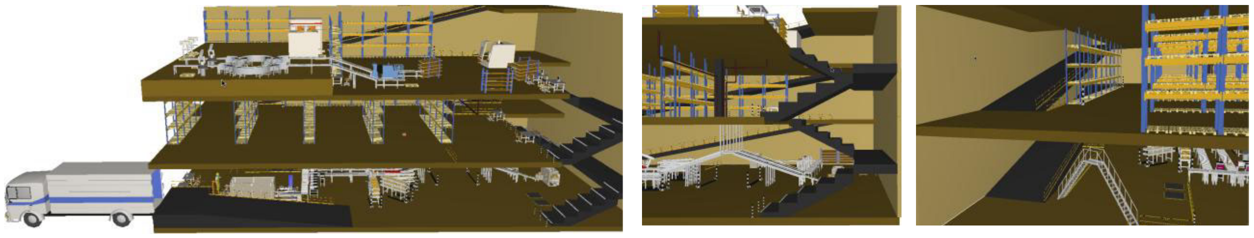

Figure 2: The left image shows the three levels of our multi-story building. The middle image shows a detailed view of the stairs and the right image shows a detailed view of two inclined ramps.

4 AN AUTOMATED 3D MOTION PLANNING FOR INTRA-LOGISTICS ENVIRONMENTS

In this section, details of the automatic MP scene processing algorithm are described. It starts with the automatic simplification of the complex layout data, since the exported CAD-data is typically too complex for further processing (see Figure 3). The idea is to convert the geometric objects from their continuous geometric representation into a set of voxels (Foley 1995), which at best approximates the continuous geometry (see Section 4.1). The second step is the computation of driving surfaces, which is a two-dimensional surface area within the 3D space. It includes all positions reachable for a specific moving vehicle (details in Section 4.2). The result of the MP scene processing is a data structure called *framed 3D rectangles*. It is used for the automatic computation of a motion planning query. For a motion planning query, the simulation framework only has to specify the start and destination position of the vehicles. The motion planning algorithm automatically finds a way, so that the simulation can use this data to compute transportation times (see Section 4.3).

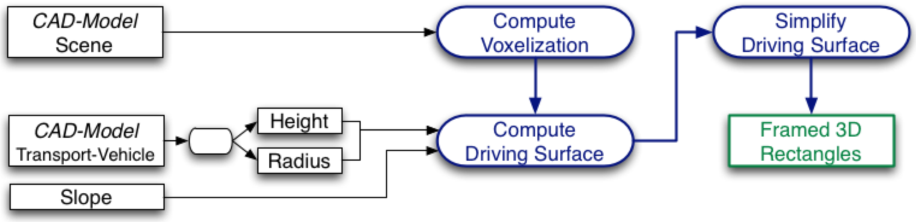

Figure 3: Structure of the MP scene processing

4.1 Simplification and Voxelization of the Scene Data

Our motion planning algorithm works on the simulation model's basic 3D-visualization data. In large simulation models such as production facilities, the exported 3D-layout data of the CAD-system is highly complex, so that it is impossible to render all objects in real-time. Nevertheless, the motion planning algorithm takes this large amount of data as the basic input for its computation. The complexity of the data causes problems for the motion planning algorithm too: firstly, the data structure that processes all data will become complex and storage as well as runtime problems will arise. Secondly, standard motion planning algorithms, e.g. trapezoidal methods, compute lots of trapezoids that generate tottering paths on a straight line (de Berg 2008).

In our approach, we solve this specific problem by the initial simplification of the complex data. The geometric objects are converted from their continuous geometric representation into a set of voxels, which at best approximate the continuous objects. We predefine in advance the size of the smallest box (cube) and generate a 3-dimensional raster of cubes for the entire scene (discretization). Afterwards, we store all cubes, met by the object's body. All empty cubes are thrown and the set of voxels is stored in an octree like data structure (de Berg 2008). Figure 4's left image shows the scene with all objects rendered with their continuous geometric representation, whereas the right image shows the voxelized data. We render only those voxels, which are met by an object's body, e.g. shelves, conveyors, floors and ceilings.

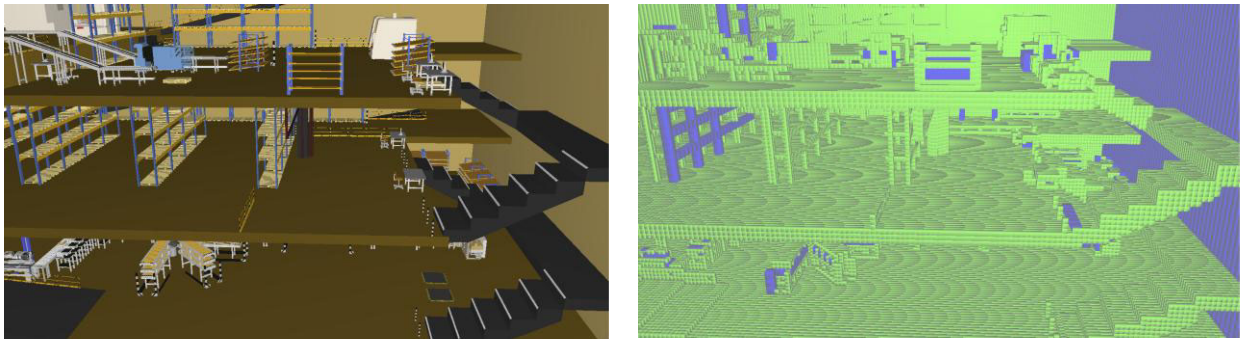

Figure 4: The 3D scene of the simulation model (left). The right image shows the voxelization of the same 3D scene.

4.2 Computation of Driving Surfaces

The second step of the *scene processing* is the computation of *driving surfaces*. A driving surface is a 2-dimensional surface, which connects all reachable points by a moving object. A vehicle placed at a specific position on the driving surface might reach all other positions of this driving surface. Moving objects can drive up inclined ramps, which do not exceed a certain angle of elevation (slope). Thus, an important parameter for the computation of an object's driving surface is the *slope* of the object. For each vehicle, the slope is defined by the modeler.

The algorithm, computing a driving surface works as follows: The modeler defines a single point of an object's driving surface, e.g. the parking place of a forklift. Based on this position, the algorithm searches all neighbored voxels having the same height as the "starting" voxel. Thereby, all neighbored points of a plan surface are identified. Moreover, the algorithm also searches for those neighbored voxels, that are still higher or lower as the "starting" voxel. Here, the slope between the neighbored voxels must not exceed the given slope parameter of the moving object.

The identified driving surface for an object is stored in an octree like data structure, similar to the one of the preceding voxelization. An example of a driving surface is shown in Figure 5's left image. It includes all three levels of the multi-floor building, whereas the single floors are connected via the inclined ramps. In order to reduce the storage of the data structure, the driving surface is decomposed into rectangles. Instead of storing all voxels of a rectangle, only the boundary of the rectangle is stored, whereas all voxels inside the rectangle are removed, so that the final data structure of the driving surface is simplified and yield a more efficient representation of the driving surface. This data structure is called *framed 3D rectangles* (an example is given in the right image of Figure 5). Each rectangle is rendered with a green boundary. The generated, framed 3D rectangles are used for the computation of specific motion paths during simulation runtime.

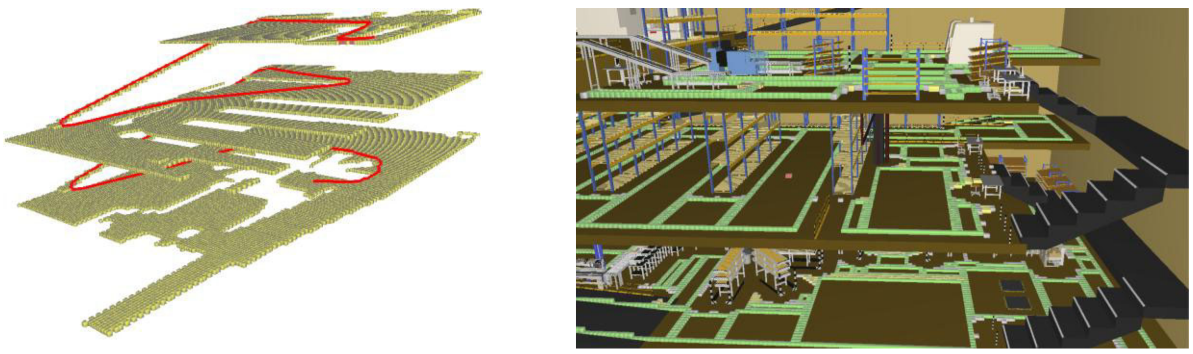
Figure 5: The computed driving surfaces of all three floors of the example facility (left). The right image shows the simplified driving surface composed out of lots of rectangles.

4.3 Automatic Computation of Paths

The algorithm for the automatic computation of single motion paths is executed during a simulation run. Details concerning the simulation are described in Section 5; here only the basic algorithm is introduced. We use the computed driving surfaces stored in the framed 3D rectangle data structure already mentioned above, as a data structure.

The path finding algorithm is triggered by the simulation framework, which must only specify an object's or vehicle's start and destination position to be moved. Both positions must correspond to the computed driving surface. Starting from the rectangle that inherits the given start position, the algorithm searches the neighborhood with a wave algorithm (Chen 1997), a sophisticated variant of the A* algorithm (Dijkstra 1959). It stops when the destination position is reached. An example is given by the red line of Figure 5's left image. Here, the computed path connects a start position on Level 3 with a destination on level 1. The simulation uses the computed path to determine transportation times and the attached 3D rendering system uses the path to visualize the vehicle movement.

5 MATERIAL FLOW SIMULATION INTEGRATION

So far we have shown the different steps needed to be able to compute a path through a 3-dimensional, multi-floor scene with obstacles. In this chapter we will show, how the motion planning algorithm has been integrated into our material flow simulation. Like most other simulation tools, our tool works with building blocks. Complex systems can be created from a library of well defined blocks, which can be connected to define the material flow. The new motion planning block is modeled as a time consuming process that moves a material flow token from its input channel to its output channel.

In contrast to a simple, randomized time consuming process, the motion planning block uses the described path finding an algorithm to compute necessary time for the token's movement. The block defines a transport connection between two distinct blocks. Thus, there will be more than one motion planning block in one scene, defining different transportation routes.

5.1 Configuration

During the design of the motion planning block we wanted a very fine granular configuration control of how a transport object handles each connection. Therefore, we have two distinct sets of objects: moving objects and the connections. The configuration allows a *n-to-n* relation between those two sets. This means that a vehicle can handle several connections and that several vehicles can operate on one connection. To eliminate the possibility of having one vehicle operating on two connections simultaneously, we use a global scheduling algorithm to manage the processing of tokens by the vehicles.

5.2 At Runtime

In this paragraph we want to describe the motion planning building block's workflow during the simulation. For this, we have chosen an example, in which four moving objects (MO*1*, MO*2*, MO*3* and MO*6*) and one building block are defined. The block is associated with three of the four objects (see **Figure 6**).

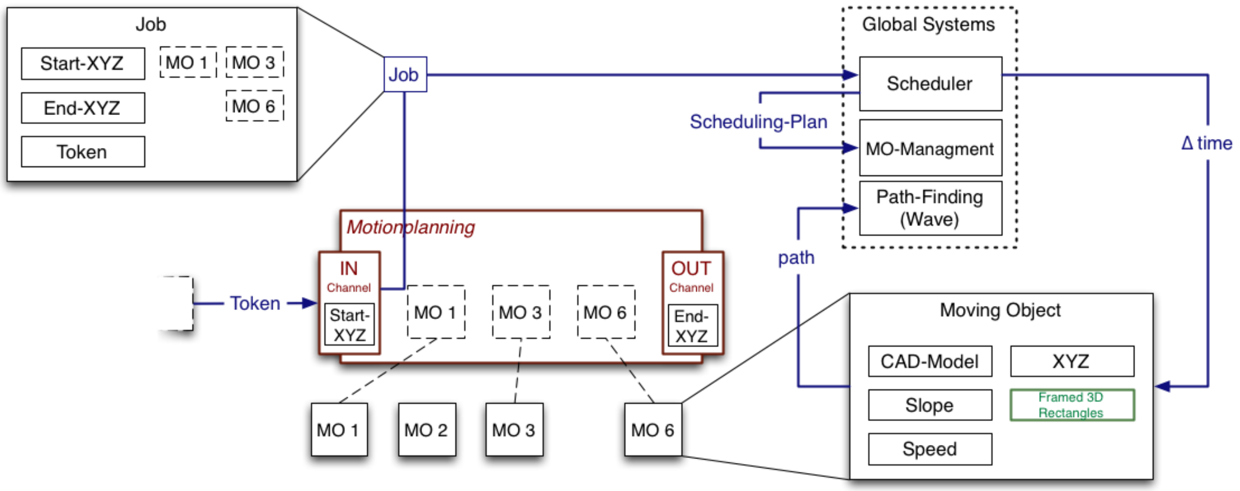

Figure 6: An example configuration. The motion planning block uses three moving objects to transport tokens. The global systems manage the different connections, jobs and transport objects.

At some time during the simulation, a token reaches the motion planning block through its input channel. The motion planning block creates the associated moving objects from the positions of the input- and output channel (named Start-XYZ and End-XYZ) and the token being processed a new job (see **Figure 6**) for the global scheduling mechanism utilized by all motion planning blocks within the simulation model. The scheduler has to solve a simple *job shop* problem with the already defined jobs, where the transport objects are the job processing machines. Since we do have different moving objects, the scheduler asks the object for the estimated processing time for a particular job. The object itself queries the wave path finding an algorithm (Chen 1997) with the start- and end positions from the job and its associated *framed 3D rectangle* data structure for path. The length of the polygon path multiplied with the moving speed of the transport object specifies the delivering time for the transported token. The scheduling algorithm uses the computed times to create a scheduling plan, which is then processed by a managing process (*MO-Management* in the above figure), advising the vehicles, which token when to process. Because of the scheduling plan, we avoid situations, where an object processes two connections simultaneously.

6 RESULTS - NAVIGATION IN A MULTI-FLOOR BUILDING

We demonstrate the application of our motion planning algorithm using a multi-floor building example (see **Figure 2**). This example's material flow simulates the following production process. Goods are manufactured on the third floor by some welding robots. Forklifts carry the work pieces from the manufacturing of the third floor to an interim storage consisting of some shelves on the second floor. The work pieces on the second floor wait to be removed by trucks. All trucks arrive at the ground floor, which has a loading ramp. The ground floor shows the distribution area for the work pieces consisting of some conveyor belts, several packaging machines and a truck. If a truck arrives at the ground floor, the forklifts carries the goods from interim storage on the second floor to the ground floor. The work pieces are packed and commissioned at the packing station at the ground floor. The truck is parked in front of a loading ramp. This ramp can be used by forklifts for loading the truck.

Inclined ramps and *stairs* are types of surfaces connecting two levels (see **Figure 7**). Workers (rendered with avatars) use the stairs to reach the next floor and forklifts and other vehicles use the inclined ramp to move to the next floor. The material flow simulation program is capable of specifying start and destination of work pieces within the CAD model, e.g. “carry work piece X at Machine 2 on Floor 3 to the Conveyor belt 3 on the ground floor”. This information is available because the simulator knows where the tokens for work pieces are generated. However, the problem is to find an available route for the forklifts carrying the work pieces. Moreover, it remains difficult to find a path for a worker’s avatar, which leaves a floor.

In our system, the simulation program must only specify the start and the destination position of vehicles and workers: the start position and the destination is a three dimensional vector. The motion planning algorithm automatically finds a way such that the 3D rendering system can move the forklift, e.g., from the third floor to the ground floor. As described in Section 4 the motion planning generates the path from the 3D data of the virtual scene. The following subsections demonstrate the features of the motion planning algorithm using the factory example.

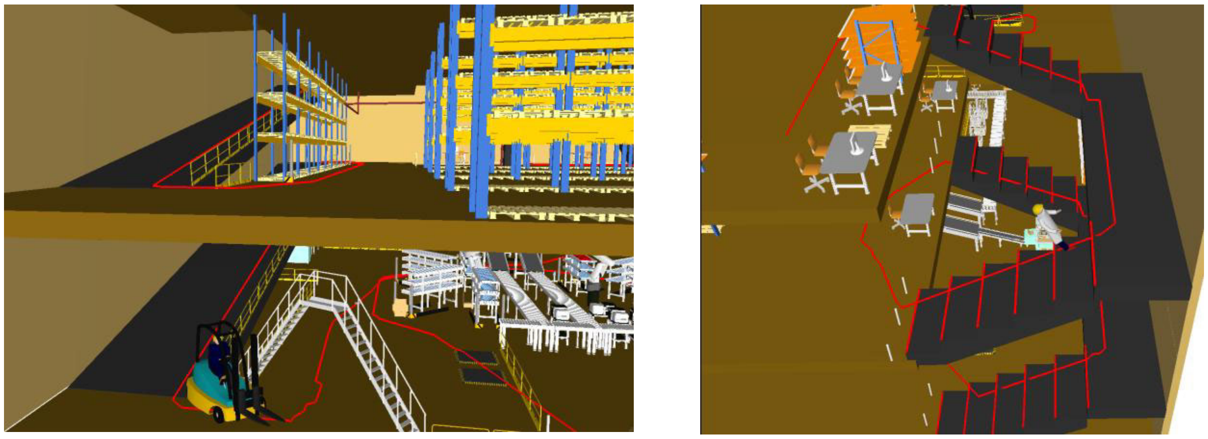
Figure 7: The left image shows a forklift driving across a ramp down to an lower level. The right image shows an avatar going upstairs.

6.1 Movement of Vehicles with Start and Destination on Different Floors

An example of the computed path is shown in the images of **Figure 7**: The red line shows the computed path. The 3D rendering system uses this path for the forklift’s movement. The computed path starts at floor three and runs along the ramp connecting floor three and floor two. It leads beside the shelves to the ramp connecting floor two and the ground floor. On the ground floor it runs around the conveyor belts and stops beside the ramp that is used for the loading and unloading of trucks (see Figure 9). The computed path is also shown on the driving surface image of Figure 5.

The path is computed automatically only specifying the start and destination position of the forklifts. Neither the floor’s number nor the obstacles’ positions have to be specified. Everything is computed on the basis of the 3D-scene’s triangles. It respects all the scene’s obstacles. This makes the application of the method simple and powerful.

We give two examples for the types of inclinations, that are possible. The first example is a ramp used for vehicles such as forklifts to drive from one floor to another floor. The slope of the inclination is a parameter used by the motion planning algorithm. This parameter can be chosen by the simulation expert. The slope value is a maximum value., i.e. the motion planning algorithm uses all inclined planes that have a slope less or equal to the maximum value. In **Figure 7**’s left image the two ramps have a slope, which is less than the specified maximum value and can therefore be used for the path. The second example is shown in the right image. These are some stairs connecting all three floors. A moving object, which shall use them must have a higher maximum slope than specified for the ramps.

6.2 Collision Prevention for Overhanging Objects

So far we have shown that the algorithm computes paths with respect to obstacles and allows the movement between floors across ramps or stairs. A further important feature is the collision prevention of overhanging objects. Overhanging objects exceed the height of vehicles that drive below them. The example scene has two kinds of overhanging objects: Figure 9's left image shows a step ladder and the third image shows pipes, which are located below the ceiling. The motion planning algorithm is capable of respecting the contour of the step ladder and the height between floor and pipes. During the computation of the driving surface, the algorithm checks for the space above a voxel. If the space is lower than the height of the bounding box, the voxel is excluded from the driving surface. Thus a collision with overhanging objects is avoided. For the path shown in **Figure 8**'s images, the algorithm uses the shape of a forklift fitting below the step ladder as well as the pipes.

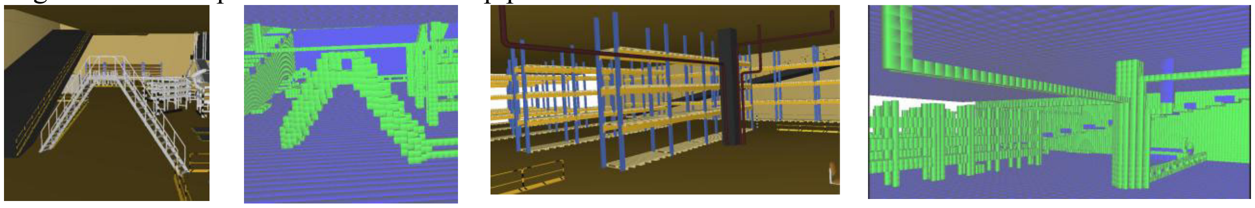

Figure 8: The left two images show a step ladder, the right two images show overhanging objects.

6.3 Dynamics and Movement for Loading Trucks

The motion planning algorithm supports the movement and insertion/deletion of objects during the simulation run. Insertion and deletion of objects is necessary if the simulation expert modifies the simulation model, e.g. by adding a new conveyor belt. Modification of the simulation model causes the modification of the virtual scene's 3D data. The motion planning perceives this modification and updates its data structures accordingly. Only the modified 3D data of the scene is necessary for the modification of the data structure.

The object's movement is an important feature with at least two applications: firstly it allows the movements of objects, e.g. forklifts carrying packets, and avatars. Secondly, it supports the handling of complicated situations such as the loading/unloading of a truck: This is explained with the help of Figure 9.

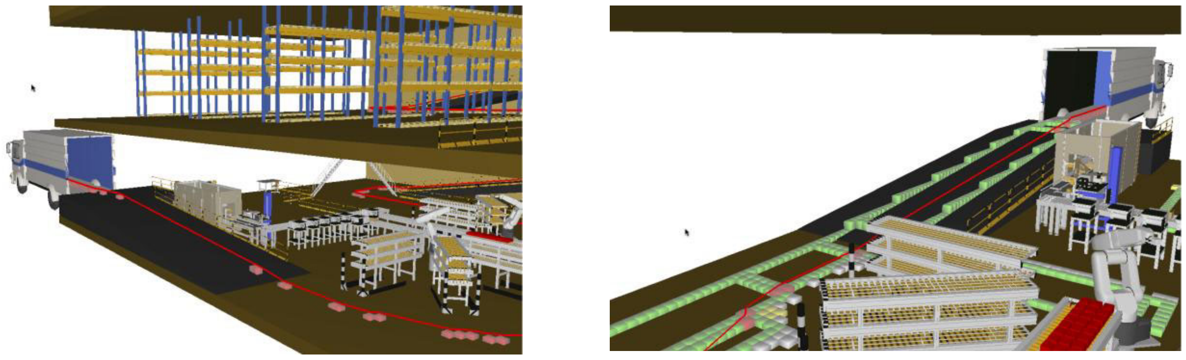

Figure 9: The images show a motion path starting from level three, crossing the ramps to level one, arriving at the truck.

The factory's ground floor shows the loading ramp used for loading the work pieces into trucks. Forklifts carry the work pieces from the conveyor belts and drive onto the ramp to load the truck. Implementing an animation for forklifts that drive just at the end of the loading ramp is easy. However, to animate a forklift that drives onto the load bed of the truck is more difficult, since the truck is also a dynamic object and additional data such as the size of the truck's load bed is necessary.

For our system this application is easy to achieve: The simulation system moves the truck to the end of the loading ramp. The motion planning algorithm receives the modified data from the corresponding 3D scene and computes the driving surface for the whole scene. The load bed of the truck will automatically be part of the new driving surface. We can see this in the images of Figure 9: the load bed of the truck is a green boarded rectangle and is thereby part of the driving surface. So it is possible for the forklifts to drive into the truck on the load bed and deposit the carried goods (see **Figure 10**). Arbitrary movement and modification of parts of the driving surface are possible. The motion planning automatically respects all modifications without knowing details of the modified parts of the 3D scene.

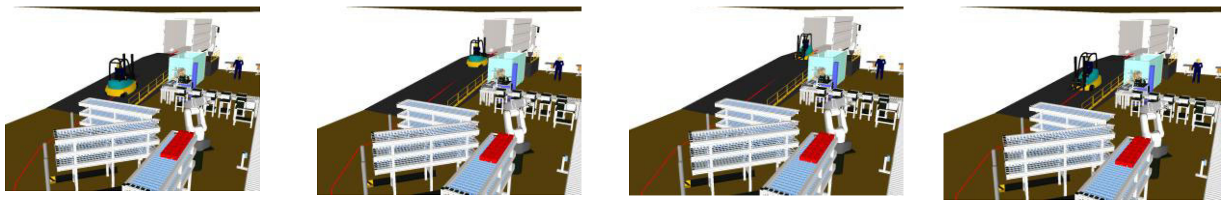

Figure 10: The four figures show a fork lift driving into the truck's loading space across the ramp and reversing again.

7 RESULTS - SURFACES FOR MOVING PACKETS ON CONVEYOR BELTS

As shown in Section 4.3, the motion planning algorithm computes several driving surfaces depending on the architecture of the building. We also use this feature of our motion planning algorithm to automatically compute "driving surfaces" on conveyor belts, too. Driving surfaces are needed for the movement of packets on conveyor belts (animation), which can be a complicated task, shown in this example. The work pieces' manufacturing is on the third floor (see **Figure 11**). Our example shows two conveyor belts with a circular crossing. At the conveyor belts' starting point there are two welding robots and at one's end there is one blue packaging machine. The problem with this equipment is the intermediate story of the third floor. The welding robots stand on a higher level as the blue packaging machine. Both conveyor belts bridge the elevation. Therefore parts of the belts are inclined.

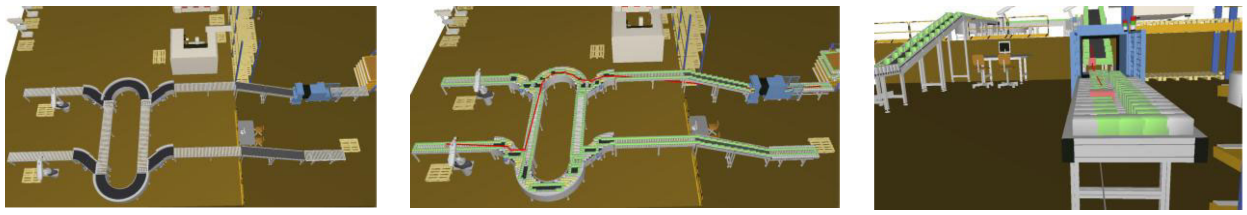

Figure 11: The left image shows a conveyor belt, the middle image shows the conveyor belt's driving surface and the right image parts of the driving surface passing through a packing machine.

The problem of the construction is the automatic time estimation and animation for the transport of a work piece across the conveyor belt. The simulation needs a correct path length across the conveyor belts surface, while the 3D rendering system needs exact positions of the conveyor belts surface for the animation. This problem may become crucial as the modeler may modify the simulation model and the factory layout. Our motion planning algorithm can easily overcome this problem. We only need a single position on one part of a conveyor belt. The motion planning algorithm automatically computes its corresponding driving surface including all positions reachable from the starting position.

The second and third images of **Figure 11** show the computed surface. It consists of green bordered rectangles. All images show a red line that is a path computed by the motion planning algorithm to move a work piece across the conveyor belt. The right images of **Figure 11** show that the algorithm respects paths leading through the interior of the packaging machine (green rectangles).

8 RESULTS – BENCHMARKS

As stated in Section 3, we integrated this motion planning framework into the modeling workflow (see Figure 1). Therefore, a fast scene processing is crucial. In this section, we present some early benchmark results obtained with the known example scene. Our test machine was a Intel Core 2 Duo 2,4 GHz machine with two gigabytes of RAM. All algorithms are implemented in Java. The corresponding 3D models consist in total of 615.344 triangles and the motion planning algorithms are integrated into a small simulation model. A forklift carries finished goods from the top floor into the truck located at the ground floor, while a worker uses the stairs to get finished goods from the top floor when rush orders arrive.

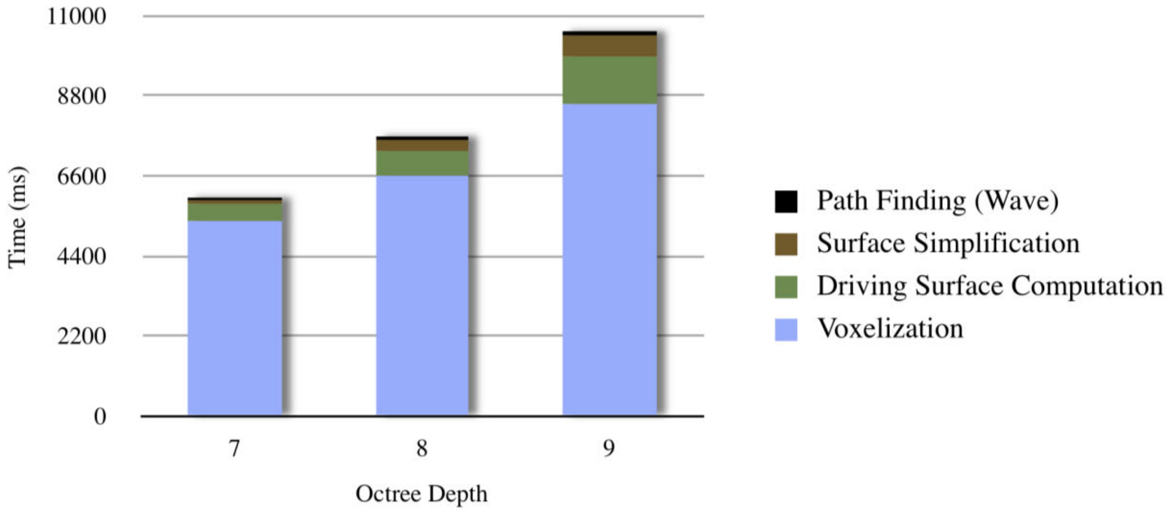

Figure 12: Benchmark results for the test scene, for different octree depths.

Figure 12 shows the accumulated times (blue shaded) for the algorithms outlined in Figure 3 and in red the necessary time of the path finding algorithm during runtime. On the X axis, three different octree depths (the granularity) are outlined. On the Y axis the time consumed (in milliseconds) for the specific steps. With an octree depth less than seven the granularity is too coarse and some overhanging objects are identified as obstacles preventing a path from the top to the ground floor. A surface computation with an octree depth higher than nine, the octree does no longer fit into the computer's memory.

Most of the scene processing time is consumed by the voxelization process, while at runtime the path finding query with 30 to 50 milliseconds is reasonable fast. Due to the usage of java as a programming language and the lack of efficiency of our code, we believe that further optimization will allow us to process smaller layout updates in less than five seconds (e.g. by only updating affected regions).

9 CONCLUSION

So far we have shown an automated motion planning framework, integrated into the scene modeling workflow from our material flow simulation framework. It provides a fast path finding algorithm through three dimensional, multi-story buildings and is also usable for token movements on conveyor belts. By mirroring the *UP*-Vector we are also able to support ceiling cranes, a feature that we could not show here due to shortage of space.

We plan to extend the method to support one-way streets and collision detection. The first method helps to control the paths used by the moving objects (e.g. distinct entrances and exits) and helps modeling paths on conveyor belts (which often only support one way) more accurately. The latter method brings more realism to the scene if more than one moving objects are defined in the scene with overlapping driving surfaces. Also it can be used for packages on conveyor belts not to get ahead of each other.

A shortage of our method is the memory consumption during the scene processing. The octree needs a lot of memory, especially due to the implementation as Java objects. To support even larger scenes with higher precision (deeper octrees) we want to re-implement the scene processing algorithms as *out-of-core* processes. To reduce the time consumption especially for large scenes, we plan to enhance the update process by updating only affected regions. Due to the local nature of the problem and the geometric splitting provided by the octree this can be easily accomplished.

The presented automated motion planning framework can be applied to simulation software used for layout planning of plants and planning of manufacturing processes. As shown in this paper, the field of application is not only restricted to automated guiding of vehicles that are driven by humans, e.g. a forklift. All moving objects are possible such as automatic guided vehicles (AGV). Also the guidance of humans, e.g. workers, across stairs is possible. Other examples are any kind of transport systems, e.g., conveyor belts for packets and moving walkways for humans are conceivable. The main roadblock to bring this method to full application is the accurate processing of the used 3D data of the virtual scene. For simulation programs, which use their own proprietary 3D models, this is easily possible. However, importing and processing 3D data of arbitrary file formats can cause problems due to the general problems in converting 3D file formats.

ACKNOWLEDGEMENTS

This work is partially supported by DFG grant DA 155/31-1, ME872/11-1, FI 1491/1-1"Synchronisierte Analyse und 3D-Visualisierung paralleler Ablaufsimulationen in interaktiv erstellten Ausprägungen" (AVIPASIA).

REFERENCES

Barnes, M.R.. 1997. An introduction to QUEST. In *Proceedings 1997 Winter Simulation Conference*, eds. S.G. Henderson, B. Biller, M. Hsieh, J. Shortle, J. D. Tew, R. R. Barton, 619-623. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Chen, D. Z., R.J. Szczerba, and Uhran Jr. 1997. A framed-quadtree approach for determining Euclidean shortest paths in a 2-D environment. *IEEE Trans. Robotics Automat.* 13(5):668–681.

Dangelmaier, W., and C. Laroque. 2007. Immersive 3D-Ablaufsimulation von richtungsoffenen Materialflussmodellen zur integrierten Planung und Absicherung von Fertigungssystemen. In *Leobener Logistik Cases - Management komplexer Materialflüsse mittels Simulation*, DUV Verlage, 16:253-268.

Dijkstra, E. W. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271.

de Berg, M., O. Cheong, M. van Kreveld, and M. Overmars. 2008. *Computational Geometry: Algorithms and Applications*, Springer, Berlin.

Delmia's Quest. 2010. Available via <http://www.3ds.com/de/products/delmia/portfolio/delmia-v5/all-products> [accessed July 16, 2010].

Fischer, M., B. Mueck, K. Mahajan, M. Kortenjan, C. Laroque, and W. Dangelmaier. 2005. Multi-user support and motion planning of humans and humans driven vehicles in interactive 3D material flow simulations. In *Proceedings of the 2005 Winter Simulation Conference*, eds. M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, 1921-1930. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Fischer, M., H. Renken, G. Schaumann, C. Laroque, and W. Dangelmaier. 2010. Movie for demonstration: Automated 3D-Motion Planning for Ramps and Stairs in Intra-Logistics Material Flow Simulations. Available via <http://www.youtube.com/user/hniupb> [accessed July 16, 2010].

Foley, J. D., A.van Dam, S. K. Feiner, and J. F. Hughes. 1995. *Computer Graphics: Principles and Practice*, Addison-Wesley.

Gan, B.P., L. Liu, S. Jain, S.J. Turner, W. Cai, and W.-J. Hsu. 2000. Distributed supply chain simulation across enterprise boundaries. In *Proceedings of the 2000 Winter Simulation Conference*, eds. J. A.

Joines, R. R. Barton, K. Kang, and P. A. Fishwick, 1245-1251. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Holweg, M., and J. Bicheno. 2002. Supply chain simulation - a tool for education, enhancement and endeavour. *International Journal of Production Economics*, Elsevier, 78(2):163-175.

LaValle, Steven M. 2006. *Planning Algorithms*, Cambridge University Press.

Incontrol's Enterprise Dynamics. 2010. Available via <http://www.incontrolsim.com/index.php/Enterprise-Dynamics> [accessed July 16, 2010].

Mahajan, K., C. Laroque, W. Dangelmaier, C. Soltenborn, M. Kortenjan, and D. Kuntze. 2005. d³FACT insight: A motion planning algorithm for material flow simulations in virtual environments. In *Proceedings of Simulation and Visualization (SimVis 2005)*, 1:115-126, SCS European Publishing House.

Rossmann, J., M. Schluse, T.J. Jung, and M. Rast. 2009. Close to reality simulation of bulk solids using a kind of 3D cellular automaton. In *Proceedings of the ASME 2009 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE)*.

Siemens' PlantSimulation. 2010. Available via <http://www.plm.automation.siemens.com/en_us/products/tecnomatix/plant_design/plant_simulation.shtml> [accessed July 16, 2010].

Torchelsen, R.P., L.F. Scheidegger, G.N. Oliveira, R. Bastos, and J.L.D. Comba. 2010. Real-time multi-agent path planning on arbitrary surfaces. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, 47 - 54.

VDI-Richtlinie 4499: Digitale Fabrik - Grundlagen. VDI Verlag Düsseldorf, Düsseldorf. 2006.

AUTHOR BIOGRAPHIES

MATTHIAS FISCHER studied computer science at the University of Paderborn, Germany. Since 1995 he has been a research assistant at the Heinz Nixdorf Institute. In 2005, he received a Ph.D. for his work on distributed virtual environments. His research interests are computer graphics, real-time rendering algorithms, and distributed computing. His e-mail address is <mafi@upb.de>.

HENDRIK RENKEN studied computer science at the University of Paderborn, Germany. Since late 2007 he is a research assistant at the Heinz Nixdorf Institute. His research interests are material flow simulation models. His e-mail address is <hendrik.renken@hni.upb.de>.

CHRISTOPH LAROQUE studied business computing at the University of Paderborn, Germany. Since 2003 he has been a Ph.D. student at the graduate school of dynamic intelligent systems and, in 2007, received his Ph.D for his work on multi-user simulation. Since then he is team leader of the "simulation & digital factory" at the chair of Business Computing, esp. CIM. He is mainly interested in material flow simulation models and the "digital factory". His e-mail address is <laro@hni.upb.de>.

GUIDO SCHAUMANN studied computer science at the University of Paderborn, Germany. Since 2009 he works for McAfee as a Software Engineer. His e-mail address is <guido_schaumann@mcafee.com>.

WILHELM DANGELMAIER studied Mechanical Engineering at the University of Stuttgart, Germany. In 1981, he became director and head of the Department for Corporate Planning and Control at the Fraunhofer Institute for Manufacturing. In 1991, Dr. Dangelmaier became Professor for Business Computing at the Heinz Nixdorf Institute. In 1996, he founded the Fraunhofer Center for Applied Logistics (Fraunhofer ALB). His e-mail address is <dangelmaier@hni.upb.de>.