2012 IEEE International Conference on Robotics and Automation
RiverCentre, Saint Paul, Minnesota, USA
May 14-18, 2012

A Modular Control System for Warehouse Automation - Algorithms and Simulations in USARSim

Damjan Miklić, Tamara Petrović, Mirko Čorić, Zvonimir Pišković, and Stjepan Bogdan

***Abstract*— In this paper, we present a control system for a fully autonomous material handling facility. The scenario we are considering is motivated by the 2011 IEEE Virtual Manufacturing Automation Challenge (VMAC). It consists of multiple autonomously guided vehicles (AGVs), transporting pallets of various goods between several input and output locations, through an unstructured warehouse environment. Only a map of the warehouse and a pallet delivery list are provided a priori. Pallets must be delivered to the output locations in the shortest time possible, while respecting the ordering of different pallet types specified by the delivery list. The presented control system handles all aspects of warehouse operation, from individual vehicle control to high-level mission planning and coordination. Delivery mission assignments are optimized using dynamic programming and simulated annealing techniques. Mission executions are coordinated using graph search methods and a modified version of the Banker's algorithm, to ensure safe, collision and deadlock-free system operation. System performance is evaluated on a virtual warehouse model, using the high fidelity USARSim simulator.**

I. INTRODUCTION

In recent years, Automated Guided Vehicles (AGVs) are starting to impact material handling and logistics in the same ways that robotic manipulators have been benefiting manufacturing processes. Modern distribution centers, ports and other large material handling facilities are reducing costs and increasing throughput by employing fleets of AGVs for transportation tasks. However, the existing large-scale solutions require significant infrastructural investments to ensure safe and reliable vehicle operation. Small and medium manufacturers require even more flexible and powerful control systems, which would enable AGVs to operate in unstructured environments with minimal interventions in the infrastructure, and even share their workspace with human workers [1]. Providing such levels of autonomy and safety to a multi-AGV system represents an challenging and exciting area of research.

Because of the significant complexity of modern manufacturing process, and nontrivial interactions between the discrete-event supervisory control and continuous-state control loops at the lower level, extensive testing must be performed in order to verify safe and correct behavior of an automated material handling system. Performing such testing or experimenting with new control designs on a real system is infeasible for economical and safety reasons. Another important consideration in the design process is the ability to measure and benchmark the performance of the control system. With these goals in mind, The National Institute of Standards and Technology (NIST), together with IEEE, is organizing the Virtual Manufacturing Automation (VMA) Competition, an annual event aimed at promoting scientific progress in the field of manufacturing automation. The competition is based on USARSim [2], a high-fidelity simulator of robots and environments built on top of the Unreal Tournament game engine. Competition rules act as a set of benchmarks for evaluating the performance of factory automation systems.

In this paper, we present our warehouse automation solution that was demonstrated at the 2011 IEEE VMAC. This solution builds on and extends our previous VMAC system, described in [3]. In our new system, we rely on the previously described algorithms for automated map processing, while all other system components have been completely redesigned and reimplemented. Instead of the monolithic architecture of last years' solution, the new system features a modular and scalable design, which allows us to exchange and experiment with different control and coordination algorithms. Low-level path planning and waypoint navigation controllers now leverage the power and built-in algorithms of the Robot Operating System (ROS) [4]. Delivery mission assignments are now done on-line, and are optimized using dynamic programming and simulated annealing techniques [5]. Mission executions are coordinated using graph search methods and a modified version of the Banker's algorithm[6], to ensure safe, collision and deadlock-free system operation. System evaluation is performed on the newer version of the USARSim simulator, featuring much more realistic physical models.

The paper is organized as follows. This year's competition scenario is described in Section II. An outline of the proposed system architecture is given in Section III. Section IV, Section V and Section VI provide details on the implemented planning and scheduling algorithms. Performance evaluation results are presented in Section VII, and Section VIII provides concluding remarks and directions for future work.

II. WAREHOUSE SCENARIO

The warehouse scenario that we are considering in this work is based on the *Mobility elemental test* of the 2011 IEEE Virtual Manufacturing Automation Competition [7]. It aims to recreate some of the challenges faced by small- and medium-sized manufacturers, when applying automation to legacy manufacturing facilities [1]. The warehouse setup features four Kiva-like AGVs, transporting pallets of goods through an unstructured environment. The vehicles can pick up pallets by driving underneath and engaging a vertical

All authors are with Faculty of Electrical Engineering and Computing, University of Zagreb, 10000 Zagreb, Croatia damjan.miklic@fer.hr

actuator that lifts the pallet off the ground. The physical dimensions of the AGVs are given in Table I.

TABLE I: AGV dimensions.

Length	Width	Height	Weight	Wheel radius	Spin speed
0.762m	0.635m	0.406m	150kg	0.1m	20rad/s

A sample warehouse layout, used at the 2011 VMAC, is shown in Fig. 1. It features three loading stations, where goods are ”produced” (spawned into the simulation). Each loading station, labeled *Loading-A* through *Loading-C*, produces a different type of goods. Goods need to be delivered to unloading stations, labeled *Truck-1* and *Truck-2*, representing delivery truck trailors. At the beginning of a simulation, the AGVs are spawned at the four docking stations. The environment is unstructured in the sense that it has no special markers, rails or similar objects to facilitate vehicle localization and navigation. A warehouse floorplan is provided a priori to the control software in the form of a *MapInfo* Interchange File (MIF) with an accompanying MID file.

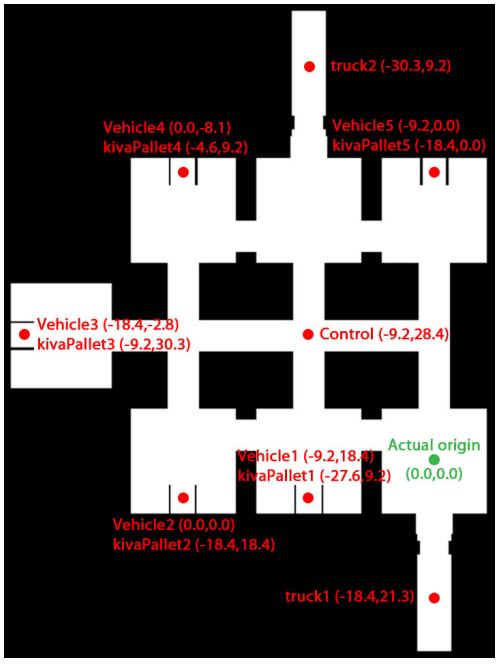


Fig. 1: A sample warehouse environment.

In the envisioned VMAC scenario, the loaded trucks need to make deliveries to several different locations. As every location requires a specific quantity of each type of goods, this places certain restrictions on the order in which goods should be loaded onto trucks. These restrictions are specified in XML-formatted *Truck order* files (one file per truck). The overall goal of the control software is to complete all the loading jobs in the shortest possible amount of time, while respecting the required loading order. The competition is carried out in several rounds, each round placing more complex loading order restrictions.

III. THE CONTROL ARCHITECTURE

Implementing a system that is capable of ensuring completely autonomous operation of a warehouse is a complex task. In order to manage this complexity, we have adopted a modular and layered approach, breaking the system down into several modules. Each module manages a well-defined segment of system operation and communicates to other modules through its interface. An interface consists of a set of messages that the module can generate and receive. This approach greatly facilitates the development, testing and performance evaluation of the system because individual modules can be developed and tested separately. Furthermore, any module can easily be replaced by a different implementation, as long as it has the same interface. Finally, this approach enables us to seamlessly transition between simulators with different levels of realism, and to eventually go from simulation to a real system, with only minor changes in the controller code.

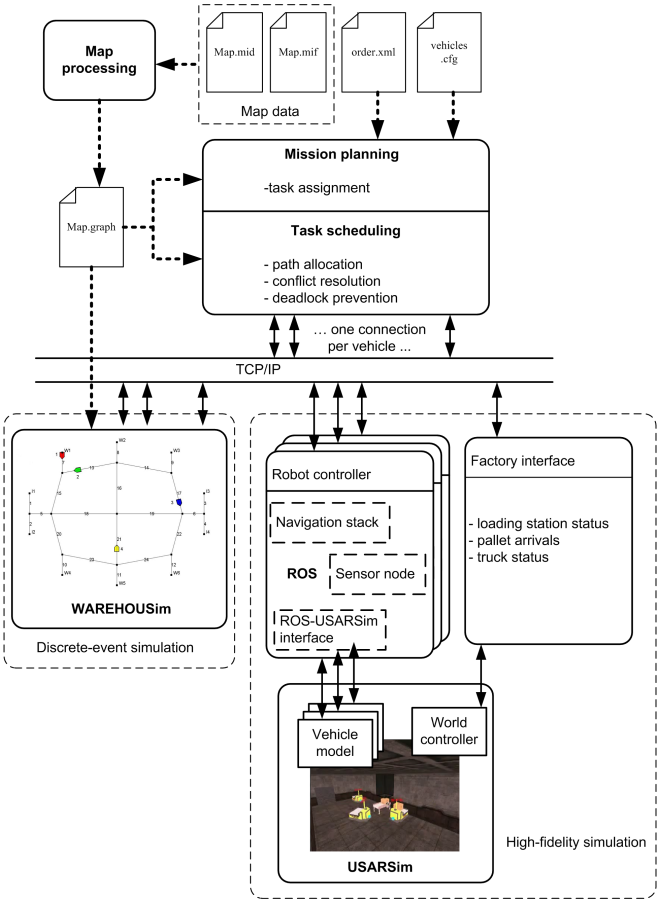


Fig. 2: Control architecture overview.

The overall architecture of our warehouse automation system is depicted in Figure 2. The main components that can be identified are *Map processing*, *Mission planning*, *Task scheduling*, *Vehicle motion control*, *Warehouse status* and simulation modules. Mission planning and task scheduling represent the core of the control system, as they implement all the high-level warehouse management logic, and are re-

sponsible for safe and efficient system operation. The mission planning module processes the XML-formatted Order file and assigns pallet delivery missions to available vehicles. The scheduling software allocates paths and ensures collision-free vehicle motion by assigning priorities in case of conflict. The scheduling module passes commands directly to motion controllers on-board the vehicles, through a message-based interface over the TCP/IP network. The interface consists of simple "GoTo"-type motion commands, so the scheduling software will work with any kind of vehicle, real or simulated, that can interpret the commands an execute these simple motions. Both the mission planning and task scheduling modules work with a graph-based representation of the warehouse environment. The warehouse graph is generated by the map processing module from the .MIF and .MID map files.

Motion controllers for individual vehicles are implemented using localization and navigation algorithms provided by ROS. In order to leverage the power of ROS, we have developed two software interface nodes. One node implements the communication protocol with the high-level warehouse control system, accepting motion commands and passing them on to the ROS navigation stack. The other node implements the USARSim communication protocol, accepting sensor data and passing motor velocity setpoints to the simulated vehicles. With this design, the entire warehouse management system could be transfered to real, physical robots, simply by replacing the USARSim node with a node that can read the sensors and pass setpoints to actual robotic hardware.

The high-level planning an scheduling software is the most complex part of the whole warehouse management system. It takes a lot of simulation iterations and test-debug-modify cycles to achieve error-free operation and optimize performance. For running a high number of simulations and evaluating only the performance of the planning and scheduling modules, it is unnecessary and cumbersome to use a high-fidelity simulator modeling physical interactions between the vehicles and the environment. Therefore, our system also features a lightweight discrete-event-based simulator called WAREHOUSim, for testing and evaluating the planning and scheduling logic. Because the simulator conforms to the communication protocol of the scheduling module, we can seamlessly transfer the tested and verified planning and scheduling algorithms to the USARSim system.

IV. Supervisory control system

Supervisory control is the highest level of warehouse control that processes pallet delivery demands on one hand and communicates with low level vehicle control on the other.

The supervisory control system consists of two modules: *mission planning* and *task scheduling*. Mission planning is the highest level algorithm that directly processes pallet delivery demands in form of .xml files, and afterwards dispatches missions (tasks) to vehicles in order to meet the given requirements. The task scheduling module is started after dispatching is done, and its role is to control execution of the assigned missions over time. This includes *i) routing* - deciding which of the possible alternative routes vehicle is going to take and *ii) scheduling* - coordination of movement for a group of vehicles by designing specific priority assignment policy.

Situations that the scheduling algorithm needs to predict and avoid are collisions and deadlocks. The goal of the supervisor is to ensure safe and accurate system functioning, while optimizing system performance in terms of total time and energy spent (distance traveled) for meeting pallet delivery demands.

In the text that follows we will introduce some basic terms that will be used throughout the paper. We start with modeling of a warehouse system using an undirected graph, which is the basis of our warehouse management system. In such a graph representation of a warehouse (Fig.3) arcs correspond to lanes along which vehicles move, whereas nodes correspond to lane intersections, input and output locations (sometimes referred to as loading and unloading stations), and vehicle parking stations. Each lane has a limited capacity and can hold only a limited number of vehicles at a time.

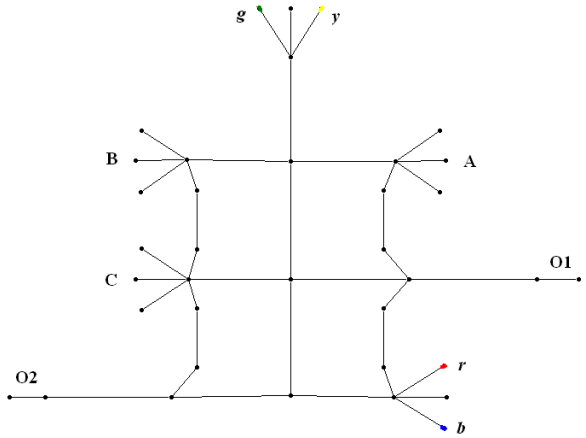


Fig. 3: Graph representation of a warehouse

We include input location nodes in set $I = \{i_1, i_2, \ldots, i_p\}$, and output location nodes in set $O = \{o_1, o_2, \ldots, o_q\}$. We assume that a certain type of pallets is produced at each input location. For the system in Fig.3, the set of input stations (or pallet types) is $I = \{A, B, C\}$, and output stations are $O = \{O_1, O_2\}$. We denote the set of idle vehicles at time t as $V_i = \{v_{i1}, v_{i2}, ..., v_{ij}\}$ and set of all vehicles as $V = \{v_1, v_2, \ldots, v_l\}$. Each vehicle v_i has its own parking station (arc) that we denote with $p(v_i)$.

V. Mission planning

We define a mission as an ordered pair $m_k = (i_i, o_j)$, where $i_i \in I$ corresponds to the input station that is a source of the pallet that needs to be transported to the output station $o_j \in O$.

For each output location, a pallet delivery list is provided that specifies the number of different pallets that need to be

transported to it. With all such delivery lists defined, mission planning algorithm determines the set of missions that need to be done, and we denote this set as $M = \{m_1, m_2, ..., m_n\}$.

For example if the pallet delivery list for warehouse in Fig. 3 is provided only for station O_2 and is given as $\{A, B, B, C, C\}$, the set of missions is then $M = \{(A, O_2), (B, O_2), (B, O_2), (C, O_2), (C, O_2)\}$.

The procedure for mission assignment is started at time t if *there is an idle vehicle, which does not perform any mission* ($V_i \neq \emptyset$) and *there is at least one non-assigned mission* ($M \neq \emptyset$).

Using an algorithm for vehicle dispatching, we need to determine a mapping of the set of idle vehicles V_i on the (part of) the active missions set M, in order to minimize the time needed to execute all active missions using the given resources (vehicles).

While developing the described solution for 2011 VMAC, we evaluated several mission assignment algorithms, including simple cyclic selection of input and output station for a vehicle, selection of an input/output location with the highest number of requested pallets etc. The best performance was achieved using the Munkres algorithm combined with simulated annealing. These algorithms are described in the text that follows.

A. Munkres algorithm

The first step towards solving the mission assignment problem is to calculate the performance cost of each mission in M by each vehicle in V. The cost $c_t(i, k)$ is calculated as the shortest distance that the vehicle v_i needs to cross, starting from its current position at time t, over the input location, to the output location of the corresponding active mission m_k. If the vehicle at time t executes a mission and is not idle, we take into account the path it needs to cross to finish its current mission.

The cost function that needs to be minimized in this optimization problem is given as:

$$C_t(x) = \sum_{v_i \in V} \sum_{m_j \in M} c_t(i, j) x_{v_i m_j}, \tag{1}$$

where:

$$x_{v_i m_j} = \begin{cases} 1 & \text{if vehicle } v_i \text{ is assigned to mission } m_j \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

A solution can be obtained using Munkres algorithm, one of most known combinatorial optimization algorithms for linear assignment problems in polynomial time [8]. Since our problem is not quadratic, we use the Munkres algorithm modified for rectangular problems [9].

The output from the algorithm is an optimal assignment strategy $X = \{\{v_i, m_j\} : v_i \in V, m_j \in M\}$, such that in optimal case mission m_i is assigned to vehicle v_j.

Having determined the strategy X, we calculate decision function $X_t \subset X$, which includes only those pairs $\{v_i, m_j\}$ where a mission is going to be assigned to a vehicle at time t. The decision function is such that a mission is assigned only to currently idle vehicles.

A special case of the assignment problem that was given in 2011 VMAC is a situation where a truck needs to deliver sets of pallets at several different stops. To simplify pallet unloading, one defines the order in which pallets need to be loaded onto the truck. Missions corresponding to the output location (truck) o_k can in that case be written as: $(M_{k1}, M_{k2}, ..., M_{kg})$ where all missions in set M_{k1} must deliver pallets before missions in set M_{k2}, etc.

In this case, the algorithm first adds the set M_{k1} to the active mission set M (which is to be assigned during the optimization procedure). After all missions in set M_{k1} deliver pallets, set M_{k2} is added to M etc. Having the active mission set M determined, the procedure is the same as for single stop demand.

B. Simulated annealing

The objective of Munkres algorithm (1) is to minimize the total distance traveled by the vehicles. However, shorter traveled distance does not necessary mean shorter time since vehicles, in order to avoid collisions, must in many cases stop and let other vehicles pass. More vehicles sharing common route parts means more waiting, and eventually a shorter route might become less favorable. Since it is infeasible to predict possible collision situations, to get a more accurate assignment strategy, we modify optimization problem to penalize route overlapping. As the optimization method, we use simulated annealing [5]. The starting point for the simulated annealing is the solution of the Munkres algorithm.

A new cost matrix, $C_{overlap}$, is then calculated so that the cost of a route increases for the total length of its overlap with other routes. In each step of the algorithm, a new neighborhood solution is chosen by: i) *assigning of a different task to a single vehicle* or ii) *swapping of tasks of two vehicles*. If the cost of the neighboring solution is lower than the current cost, the neighbor is accepted as the new best solution. If it is of higher cost, it is accepted with certain probability, which depends on variable $T(temperature)$, and decreases as the algorithm evolves depending on the value of α *cooling constant*. Algorithm terminates after a given number of iterations (I_{max}).

We will clarify the mission assignment procedure for the situation given in Fig. 3 and active missions $M = \{(A, O_2), (B, O_2), (B, O_2), (C, O_2), (C, O_2)\}$. All vehicles are idle, that is, $V = V_i = \{b, r, g, y\}$. Since all vehicles are idle, cost is the shortest distance needed to execute a mission. The cost matrix is given as:

$$C^0 = \begin{bmatrix} & m_1 & m_2 & m_3 & m_4 & m_5 \\ b & 7806 & 7663 & 7663 & 5796 & 5796 \\ r & 7805 & 7662 & 7662 & 5795 & 5795 \\ g & 7407 & 5825 & 5825 & 5797 & 5797 \\ y & 7410 & 5827 & 5827 & 5800 & 5800 \end{bmatrix}$$

The output from the Munkres algorithm is the following mapping: $X = \{\{b, m_3\}, \{r, m_4\}, \{g, m_1\}, \{y, m_2\}\}$. In the next part, this solution is forwarded to the simulated annealing procedure with parameters: $T = 5800, \alpha = 0.6, I_{max} =$

10. Simulated annealing redispatches missions to penalize path overlapping. The result is the modified assignment: $X = \{\{b, m_2\}, \{r, m_1\}, \{g, m_4\}, \{y, m_3\}\}$.

VI. Task Scheduling

Having assigned missions to vehicles, the routing algorithm determines which route the vehicle is going to travel to successfully finish the given mission. Routing can in general be static (route does not change during mission execution) and dynamic (route is allowed to change). Our approach combines the properties of both approaches.

The basic idea behind the route assignment is that a vehicle can be given a certain route if and only if it is *safe*, that is, if all other missions can afterwards be finished without collisions and deadlocks. A trivial example for a non-safe route is one where a vehicle is blocking another vehicle, preventing it from finalizing its own mission. Route interactions can be very complex even for medium-scale systems, which makes avoidance of similar situations difficult.

A new route is assigned to a vehicle in two specific situations: i) *a new missions is assigned to a vehicle* ii) *a vehicle has picked up a pallet at the input station* i. The assigned route consists of two parts:

1) path from the current position to the next (input/output) station
2) path from the next station to the parking station

When a new mission is assigned to a vehicle (v), the route that is assigned first is the shortest possible. If this assignment is not verified to be safe, the algorithm tries to find a longer, but safe route. If there is not a single safe route, the vehicle remains in its idle node (if no route has been previously assigned to it), or continues to move along its (safe) route to the parking station, continuously checking whether a safe route to the given input station i exists. In the worst case, a route will be assigned in its parking station [10].

The scheduling algorithm, which acts as a centralized supervisor, is called each time some vehicle arrives in front of a node, which can correspond to a crossroad or a station. The algorithm knows the positions and routes for all vehicles and its role is to allow or deny vehicle transfer to the next arc on its route.

As we have said before, scheduling should verify that the system is always in a safe state, where there exist at least one scenario of vehicles movements such that all missions can be safely executed. All other states are either deadlock or are going to inevitably end up in a deadlock. The vehicle transfer should be denied if and only if it will lead the system into a deadlock. Unfortunately, the problem of state safety verification is in general NP-complete [11], so various polynomial algorithms that identify only necessary conditions for system safety are used today in practical applications.

Our approach, which is thoroughly described in [10], is based on Banker's algorithm [6]. According to the Banker's algorithm, a state is safe *if* missions can finish sequentially, one by one. Since this is a rather conservative strategy, we modified it to check not only if the requested transition is safe, but also if a vehicle can, by advancing on its route, get to a Banker's safe state. This modification results in increase in vehicle utilization for certain types of layouts.

VII. Performance Evaluation in USARSim

The developed warehouse management solution is evaluated in both WAREHOUSim simulator and USARSim simulation environment. System functionality and performance were analyzed for the warehouse layout shown in Fig. 1 for three different scenarios (S1, S2 and S3) given in Table II. Simulation included four Kiva-like vehicles. For each scenario, one pallet delivery list is provided for every truck (T1, T2). All scenarios have the same total number of packages, with a different number of stops that truck is required to make to deliver the packages to their final destinations. For example, in S2, truck T2 first needs to load one A package for Stop 3, then one A and two C packages (in any order) for Stop 2, and one A and one B package for Stop 1.

TABLE II: Delivery mission scenarios.

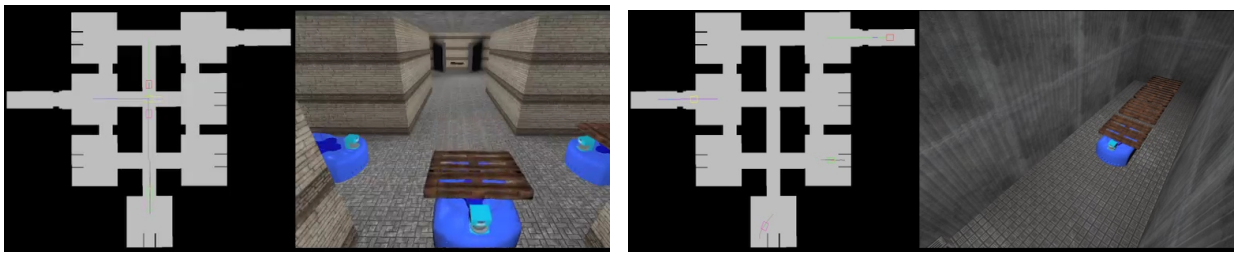
	Stop1	Stop2	Stop3	WHSim	USARSim	
T1	2A,3B,C	-	-	AABBCB	AABBCB	S1
T2	3A,B,2C	-	-	CBCACA	CBCAAC	
T1	2A,3B,C	-	-	AACBBB	AACBBB	S2
T2	A,B	A,2C	A	BA-CCA-A	BA-CCA-A	
T1	A,B,C	A,B	B	ACB-AB-B	ACB-AB-B	S3
T2	A,B	A,2C	A	ABC-CA-A	BAC-AC-A	

TABLE III: Algorithm performance metrics.

	Distance	Time	Pallets	Idle	Waiting	
S1	818m	6:51	12	0%	11.64%	
S2	793m	7:56	12	1.78%	14.74%	WHSim
S3	814m	7:33	12	3.43%	15.33%	
S1	814m	20:40	12	0%	20.64%	
S2	801m	21:35	12	3.18%	21.3%	USARSim
S3	788m	18:12	12	4.6%	14.08%	

Performance results are shown in Table III. For each scenario we logged the total distance traveled, total time needed to fulfill the order, total number of pallets that are delivered, the average time all vehicles spent idle in their parking station and the average time each vehicle spent waiting due to the scheduling policy restrictions.

Simulation results in both simulators show that the average idle time increases with scenario complexity (number of truck stops). That is due to the properties of mission planning algorithm, which ensures that any missions corresponding to a certain truck stop can not be started before missions for all subsequent stops are done. The total time needed to finalize the order is lowest for scenario with one stop, that is, no order requirements. An interesting observation is that for the third scenario, S3, the USARSim simulation achieved the best performance in comparison with other two, which is an unexpected result. This is partially due to the fact that for real vehicles, like the ones simulated in USARSim, the time required to complete a delivery depends not only



(a) Three autonomous vehicles, waiting for priority assignment by the scheduling algorithm.

(b) Pallet delivery at the unloading station (truck cargo area)

Fig. 4: Snapshots from simulation runs at VMAC 2011. The left side of each subfigure shows the warehouse floorplan with current robot positions and motion plans visualized in the *rviz* tool from ROS. The right side shows the USARSim world.

on route length but also on its structural characteristics, such as arc curvature, and vehicle's physical properties. As shown through the presented experiments, these effects are not observed in a discrete-event-based simulation, but they significantly impact the results of the high-fidelity simulation. This highlights the value of realistic, physics-based simulation for evaluating the performance of control algorithms.

Snapshots from an USARSim simulation run are shown in Fig. 4, and an edited video showing our software in action at IEEE VMAC 2011 is available for online viewing [7].

VIII. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a control architecture for autonomous warehouse management. The warehouse scenario has been motivated by the 2011 IEEE Virtual Manufacturing Automation Mobility Challenge. The described architecture incorporates all levels of autonomous warehouse management, from individual robot control to facility-wide planning and scheduling. The modular design approach facilitates system testing and reconfiguration. The performance of the control system has been evaluated in a discrete event-based simulator and in the high-fidelity USARSim simulator. Experiments have demonstrated the system's ability to enforce safe and correct multi-vehicle operation, preventing collisions and deadlocks and ensuring correct pallet deliveries throughout the warehouse environment. Experiments conducted in USARSim have also brought up some shortcomings of the implemented conservative scheduling strategy, which were not obvious in purely discrete event-based simulation. These results highlight the importance of a realistic simulation environment which models physical interactions.

In future research, we will investigate more efficient scheduling algorithms and integrate them more tightly with mission planning. Furthermore, we are planning to replace the current simulator with the new version of USARSim, based on the freely available Unreal Development Kit, which should provide a more stable simulation environment. Finally, we expect to further improve our warehouse management system trough more advanced scenarios and competition with other teams in future VMA Mobility Challenges.

ACKNOWLEDGMENTS

The work of S. Bogdan, T. Petrović and D. Miklić is supported by The Croatian Ministry of Science, Education and Sports, through grant #036-0363078-3016 (Task Planning & Scheduling in Robotic and Autonomous Systems). We would like to thank LARICS students Antonio Krnjak, Ivo Petković, Tonko Visković and Hrvoje Vugrinec for implementing the map processing algorithms, the WAREHOUSim application and ROS interface nodes used to perfom the described research.

REFERENCES

[1] S. Balakirsky, R. Madhavan, and C. Scrapper, "NIST/IEEE Virtual Manufacturing Automation Competition: from earliest beginnings to future directions," in *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*. ACM, 2008, pp. 214–219.

[2] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper, "USARSim: a robot simulator for research and education," in *2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 1400–1405.

[3] D. Miklic, , S. Bogdan, and L. Kalinovcic, "A control architecture for warehouse automation - Performance evaluation in USARSim," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2011, pp. 109 –114.

[4] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *Open-Source Software workshop of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2009.

[5] D. Kreher and D. Stinson, *Combinatorial Algorithms: Generation, Enumeration and Search*. CRC Press, 1999.

[6] E. Dijkstra, "The mathematics behind the Banker's algorithm," *Selected Writings on Computing: A Personal Perspective*, pp. 308–312, 1977.

[7] M. Coric, A. Krnjak, I. Petkovic, Z. Piskovic, T. A. Viskovic, H. Vugrinec, T. Petrovic, D. Miklic, and S. Bogdan, "Virtual Warehouse Management (IEEE VMAC 2011)," Online, June 2011, http://www.youtube.com/user/LaricsLab.

[8] J. Munkres, "Algorithms for the assignment and transportation problems," *SIAM Journal of Applied Mathematics*, vol. 5, pp. 32–38, 1957.

[9] F. Bourgeois and J.-C. Lassalle, "An extension of the munkres algorithm for the assignment problem to rectangular matrices," *Commun. ACM*, vol. 14, pp. 802–804, December 1971.

[10] K. L., P. T., B. V., and B. S., "Modified Banker's ALgorithm for Scheduling in Multi-AGV Systems," in *2011 IEEE Conference on AUtomation Science and Engineering*, 2007, Trieste, Italy.

[11] S. Reveliotis and E. Roszkowska, "On the complexity of maximally permissive deadlock avoidance in multi-vehicle traffic systems," *Automatic Control, IEEE Transactions on*, vol. 55, no. 7, pp. 1646 –1651, july 2010.