A safe autonomous stacker in human shared workspaces*

Luis Louro
centro ALGORITMI
University of Minho
Portugal
llouro@dei.uminho.pt

Duarte Teixeira
centro ALGORITMI
University of Minho
Portugal
a73822@alunos.uminho.pt

Tiago Malheiro
centro ALGORITMI
University of Minho
Portugal
tmalheiro@dei.uminho.pt

Luis Mesquita
centro ALGORITMI
University of Minho
Portugal
a78635@alunos.uminho.pt

Toni Machado
CM/LOI-Brg
Bosch Car Multimedia
Braga, Portugal
Toni.Machado@pt.bosch.com

Sergio Monteiro
centro ALGORITMI
University of Minho
Portugal
sergio@dei.uminho.pt

Wolfram Erlhagen
dept. Mathematics and Applications
University of Minho
Portugal
wolfram.erlhagen@math.uminho.pt

Estela Bicho**
centro ALGORITMI
University of Minho
Portugal
estela.bicho@dei.uminho.pt

***Abstract*—This paper proposes a solution for safe navigation of stacker vehicles in workspaces shared with people, with a focus on the docking manoeuvres for pallet picking and dropping. Behaviours for way-point and wall following are developed following the *attractor dynamics approach*. Then, these behaviours are orchestrated by state machines (that activate or deactivate them) depending on the specific task. Each of these states also defines different safe areas and maximum travel speeds, which is a requirement for safe operation. Results of real experiments are presented that show the standard operation and its robustness against perturbations (people in the way) and failure detection (missing pallets).**

***Index Terms*—Autonomous stacker, safe autonomous navigation and docking, workspaces shared with humans**

I. Introduction

This paper is about the development of safe autonomous stackers that are able to operate in environments shared with people. Up to this day, warehouse logistics is mostly supported on the use of stackers (or forklifts) to move goods around. Although being usually driven by humans, recent years have brought about examples of warehouses relying entirely on autonomous vehicles where the human intervention is minimal or absent [5], [19], providing structured environments where it is possible to benefit from the flexibility of operation, traceability of goods and direct integration with management tools that autonomous vehicles provide. If these vehicles are expected to operate in dynamic environments shared with people, such as production plants or assembly lines, then special care has to be taken to safety issues.

The development of an autonomous stacker poses many challenges. While most of them are shared with the typical autonomous robot (such as, for example, path and motion planning [16], localization and mapping [2]), other challenges are rather specific to the vehicle in question, such as the generation of precise motions that allow the vehicle to pick and drop pallets in specific localizations and orientations. For this purpose, in [10] it is used a path tracking strategy with an implementation based on screw theory, that minimizes the position and heading error regarding a generated B-spline curve that takes into account the docking location. Others, use model predictive controllers [11], sliding mode controllers [4], or even fuzzy logic controllers [13], as a solution to ensure this path tracking control (see, e.g. [18] for a comparison between model-based approaches and non model-based approaches).

Another specific problem to this type of vehicle, is that the stacker has to be able to detect pallets and to manoeuvre in order to pick it. This problem requires the use of local perception in order to make up for positioning errors of the vehicle and misplacement errors in the pallet. The solution to this problem typically relies on the use of laser scanners [7], 2D cameras [21], 3D cameras [20], a combination of the former [12], or even using infrared systems [17]. The pick phase is generally translated to a servo command problem, with visual servoing being a typical choice [1], [15]. Others still rely on pallet markers to navigate during the picking phase [7]. Yet, the safety aspect of the exploration of these vehicles is often overlooked in what regards its deployment in dynamic and human populated environments.

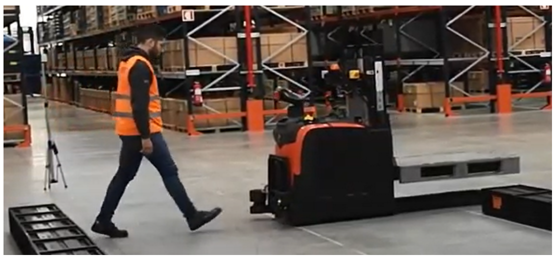
Fig. 1. The autonomous stacker in operation.

*This work has been supported by FCT Fundação para a Ciência e Tecnologia within the R&D Units Project Scope: UIDB/00319/2020.
**Corresponding author

The work presented here is part of a project that aims at developing solutions for endowing conventional human-driven stackers with the capacity to operate autonomously. This encompasses the addition of specific sensors, the development of custom electronics, firmware and interfacing software, and also all the high-level software, ranging from the generic sensor processing, localization and mapping up to the specific controllers to the selected tasks. In previous work, we have already presented the low-level architecture and interfacing but applied to a tugger vehicle [8]. Here, one extends that work by focusing on stacker specific tasks, i.e. on the behaviours that allow performing pick and drop actions and their orchestration in what regards safety in a human shared workspace. Task controllers are developed using a hybrid approach. On the one hand, elementary behaviours are formalized as nonlinear dynamical systems in terms of path velocity and heading direction, which are based on the *Attractor Dynamics Approach to Behavior Generation* [14], and build on our previous work [8], [9]. On the other hand, these behaviours are activated and deactivated by state machines whose states also define safety parameters for vehicle operation (protection areas around the vehicle and maximum velocity, for instance). The solution proposed here was implemented in a real stacker and tested in a realistic environment. Results demonstrating the ability to perform pick and drop operations, which may be challenged by perturbations (e.g. people in the way), are presented. The results also include links to videos showing the staker in operation.

The remainder of the paper is structured as follows: section II describes the vehicle setup and the task constraints; an overview of the system architecture is presented next, in section III; the dynamics of the behaviors composing the motion controller for manoeuvres is presented in section IV; which is followed by the description of how one can use these dynamics to perform pick and drop manoeuvres, in section V; section VI presents several experimental results, and the paper finishes with conclusions and future work in section VII.

II. Stacker Vehicle and Task Constraints

The autonomous Stacker is based on a commercially available human-operated stacker machine, fitted with two coupled drive systems that control translation and steering (see Fig. 2). In order to enable autonomous operation, the vehicle required a reengineering of its interfaces, for which dedicated Electronic Control Units were developed. The vehicle's actuation is exposed on a ROS network using the ROS Control framework, to which the high-level controllers will interface. Most of this high-level control and interface has been previously developed for a tugger with similar internal architecture (see [8] for more details). The vehicle's indoor localization is also the same as in [8], i.e. based on an xsens Inertial Measurement Unit and an Eliko Real-Time Localization System. Furthermore, three safety laser scanners (Sick S300 LiDAR) were installed to ensure safe operation: one on each corner of the vehicles front, and the third one on the rear, under and in between the forks, to ensure a 360° vehicle surrounding coverage. Because this

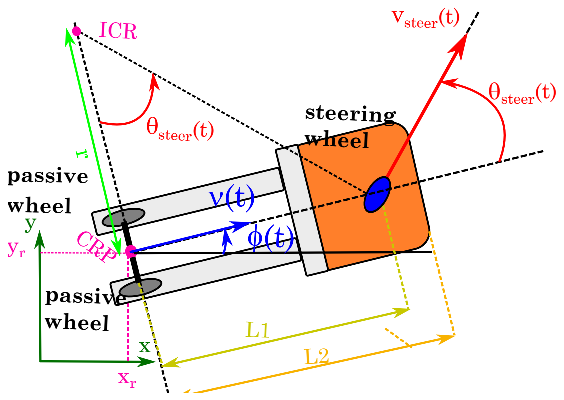

Fig. 2. Tricycle kinematic model for the Stacker vehicle, with actuation variables (θ_{steer} and v_{steer}) and behavioural variables (ϕ – heading direction and v – path velocity).

third scanner is occluded with the forks totally down, there is a requirement to keep the forks at the height of 30 cm, except when absolutely required to perform pallet loading or unloading (with previous scanner deactivation). It secures a hazardous area around the vehicle by triggering the emergency stop should it detect an object or person in the predefined protective fields. These particular areas were defined after risk analysis and are selected online depending on the vehicles path velocity and subtask being performed.

An Orbbec Astra 3D camera, pointing to the rear of the vehicles allows for pallet detection and estimation of its relative localization. Then the distance from the forks to the pallet and their relative misalignment are computed. This misalignment is defined by two values (Fig. 3): the angle, α, that the center of the vehicles forks makes with the central column of the pallet (0° when aligned), and the angle, β, that the center of the vehicles forks makes with the orientation of the pallet (90° when aligned). The vehicle is also fitted with an absolute wire traction encoder (SICK BCG08-L1KM03PP), connected to the stacker forks, which provides information on their current height, and a photoelectric sensor (Sick W2S-2) to detect the complete insertion of the forks in the pallet.

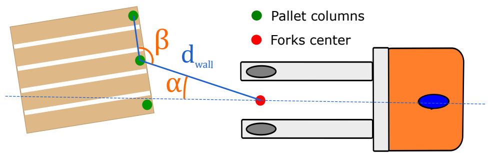

Fig. 3. Vision system tracks the values of pallet distance (d_{wall}), α and β angle.

This stacker is designed to perform pick (load pallet) and drop (unload pallet) tasks is an industrial facility where the environment is shared with other vehicles (either human driven or autonomous) and human workers. As such a special attention has to be addressed to safety issues. After a risk analysis on the tasks and the environment, the following constraints were defined: i) the vehicle should move only on its lane (right one); ii) the exception is in corners where, due to the vehicles

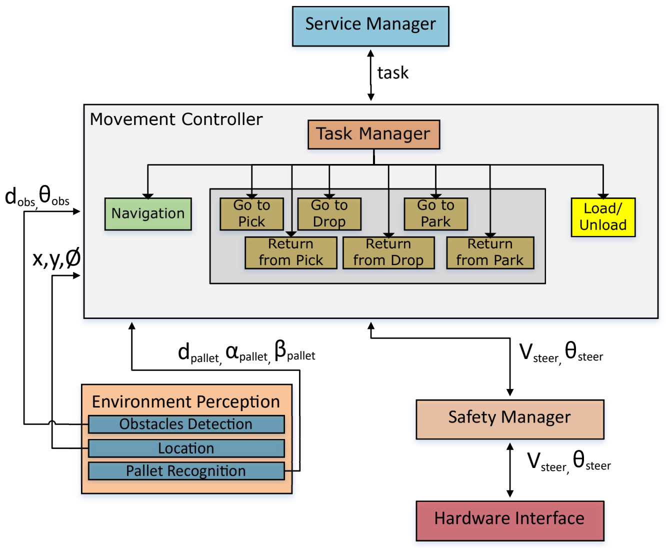

Fig. 4. Overview of the system architecture.

dimension, keeping lanes is a physical impossibility; iii) safety distances must be kept at all times; iv) raising or lowering the forks is only allowed when the vehicle is stopped.

III. Overall System Architecture

The operation of the stacker is commanded by orders called services (e.g., "Load the existing pallet in the production area and transport it to the storage area"). These services are received by the vehicles' *Service Manager* (see Fig. 4 for a representation of the system architecture), and subsequently divided into a sequence of tasks (e.g., "Task 1 - Move to the production area and load the pallet.", "Task 2 - Transport the pallet to the storage area and unload it"), which are then sent to the *Task Manager*. The purpose of the *Task Manager* is to further split down the tasks into more elementary operations and manage their correct execution. These operations are: *Navigation*, which controls the movement of the vehicle between different areas of a plant and into the vicinity of a pick or drop location; *Go to Pick* should be only active near a pallet and allows the stacker to perform the entire approach/berthing operation to insert the forks into the pallet; *Return from Pick* makes the vehicle exit safely the loading zone, when the pallet is loaded on the forks; *Go to Drop* permits the stacker to approach the unloading zone and is terminated when the pallet is already at the place of unloading; *Return from Drop* allows the stacker to safely exit the unloading zone; *Go to Park* makes the approach to the parking zone, while the *Return from Park* operation ensures the safe exit from the parking zone; finally, the *Load/Unload* operations moves the forks up/down. Each of these operations have different risks and, as such, have different limitations in terms of velocities allowed and also protective areas.

The *Movement Controller* receives information from *Environmental Perception*, regarding the existence of obstacles in the surroundings, estimation of the vehicles' location and pallet detection and relative localization.

The *Safety Manager* analyses the speed values, steer wheel rotation angle and fork effort received from the *Movement Controller*, which are then sent to the vehicle hardware.

IV. The Dynamics of Manoeuvre Tasks

All of the tasks and operations mentioned in the previous section rely on the ability of the vehicle to head to a particular location, ensuring that there is no collisions with obstacles and persons, and stopping the movement at a safe distance. Additionally, the vehicle should maintain a specified distance (slightly larger than the minimum safe distance) to the wall(s). For this purpose one defines two basic behaviors: i) *move to target* via point and ii) *follow wall*. Dynamical systems theory is used here as a theoretical language and tool to design each of these behaviors, as well as of their integration. Specifically, to model the motion controller we use the heading direction, ϕ, and path velocity, v, as control/behavioral variables (see Fig. 2). The navigation behavior is generated by providing values in time to these variables, which will then be used to control vehicle's actuated wheel (c.f. section VI)(see [3], [14] for more details on the approach). The time course of the behavioral variables, $\phi(t)$ and $v(t)$, is obtained from fixed-point solutions of dynamical systems in the form of differential equations:

$$\begin{aligned}\frac{d\phi}{dt} &= F(\phi, parameters)\\ &= (c_{tar} * f_{tar}(\phi) + c_{follow} * f_{follow}(\phi)) \qquad (1)\\ \frac{dv}{dt} &= G(v, parameters)\\ &= c_{tar}(1 - c_{obs}) g_{tar}(v) +\\ &+ c_{follow}(1 - c_{obs}) g_{follow}(v) +\\ &+ c_{obs} * g_{obs}(v) \qquad (2)\end{aligned}$$

where the vector fields $F(\phi)$ and $G(v)$ consist of a number of contributions that express independent task constraints (move to target via-point, follow wall). In isolation, each contribution, f_i (*i=tar*, *follow*) or g_i (*i=tar*, *follow*, *obs*), creates an attractor at a desired value (e.g. direction of the next target via-point), with a specified strength and range of attraction. c_i (*i=tar*, *follow*, *obs*) are activation variables that determine the activation of each contribution in Eq. 1 and Eq. 2. c_{tar} and c_{follow} are activated depending on the operation selected by the *Task Manager* (c.f. section V). For safety purposes, c_{obs} is active whenever near obstructions making the path velocity being dependent only on $g_{obs}(v)$ and deactivating both $g_{tar}(v)$ and $g_{follow}(v)$. Parameters are tuned such that the motion is governed by a sequence of asymptotically stable states. This makes the control system robust against perturbations or noisy sensory information.

Next, we explain how the individual contributions to the vectors fields (1) and (2) are built.

A. Dynamics of heading direction

The heading direction dynamics (1) has two main components: 1) f_{tar} is responsible for orienting the vehicle into the direction of the next target via-point; 2) f_{follow} is responsible for keeping the vehicle parallel to the wall and at a desired distance from it. Rotation towards the direction at which the target point lies, has been already presented in previous works, it is modeled by an attractive force-let $f_{tar}(\phi(t)) = -\lambda_{tar}\sin(\phi(t)-\psi_{tar})$ (see [8] for more details).

In the *follow wall* behaviour the vehicle must navigate so as to follow the outline of a wall at a predefined fixed distance. The dynamic system describing this behaviour is obtained according to

$$\frac{d\phi}{dt} = f_{wall}(\phi) = c_{wall}\Big[f_{inwards}(\phi) + f_{outwards}(\phi)\Big] \quad (3)$$

where c_{wall} allows controlling the magnitude of the force of attraction or repulsion to the wall, and $f_{inwards}(\phi)$ and $f_{outwards}(\phi)$ are given by the $f_i(\phi) = \lambda_i \sin(\phi - \psi_i)$. When the vehicle is under the influence of the behaviour of following a wall on the right side, the contribution $f_{inwards}(\phi)$ creates an attractor in the direction $\psi_{inwards} = \psi_{wall} - \Delta\psi$, where ψ_{wall} is the estimate of the wall orientation that the vehicle is following, and $\Delta\psi$ is a fixed angle. The magnitude, $\lambda_{inwards}$, of the attractor increases with the distance to the wall, and can be obtained from

$$\lambda_{inwards} = \frac{1}{1 - e^{-\frac{d-d_{des}}{\mu}}} \quad (4)$$

with d being the minimum distance to the wall given by measuring the sectors on the right side of the vehicle, and d_{des} being the desired distance to the wall. Similarly, $f_{outwards}(\phi)$ creates an attractor in the direction $\psi_{outwards}$ given by $\psi_{outwards} = \psi_{wall} + \Delta\psi$ with decreasing magnitude to the wall, $\lambda_{outwards} = 1 - \lambda_{inwards}$. The superposition of both contributions create an attractor that points away from the wall, if the vehicle is too close, or towards the wall if it is more distant. When at the desired distance, the vehicle navigates parallel to the wall.

If the wall to be followed is on the left of the vehicle, then the only change to perform on the previous is to ensure that the attractors are $\psi_{inwards} = \psi_{wall} + \Delta\psi$ and $\psi_{outwards} = \psi_{wall} - \Delta\psi$.

This wall following behaviour applies both to real walls, which are sensed by the laser scanners, and to virtual walls, limiting stacker's forbidden zones, which are computed from a map and the stacker's current position.

B. Dynamics of path velocity

Each contribution in the path velocity dynamics, Eq. 2, is of the form $g_i(v(t)) = -\lambda_{v,i}\left(v(t) - v_{des,i}\right)$, which sets an attractor state at the desired value $v = v_{des,i}$ for the vehicle's path velocity, with relaxation rate $\lambda_{v,i}(> 0)$ (*i=tar*, *follow*, *obs*).

$g_{tar}(v(t))$ sets the cruise velocity for the vehicle and allows a smooth stop at the destination. $g_{obs}(v(t))$ reduces path velocity in the vicinity of obstructions, for safety reasons (check [8] for more details).

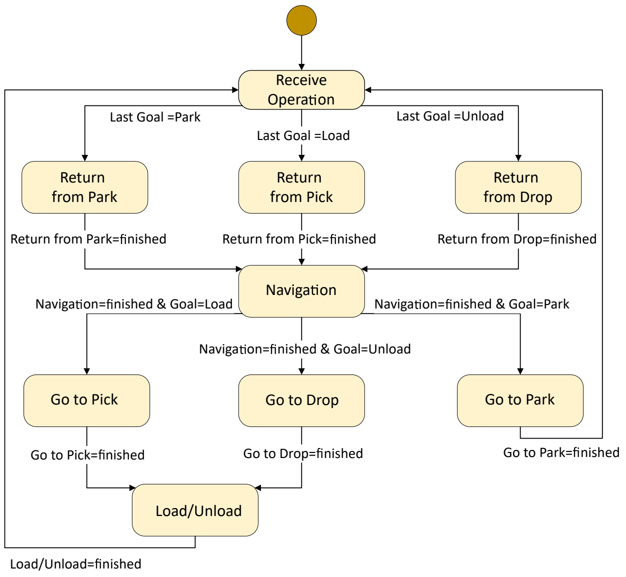

Fig. 5. Blocks diagram for the *Task Manager* component that manages operations' execution.

The attractor for the velocity dynamics of the *follow wall* behavior is

$$v_{des,wall} = \begin{cases} \frac{(1-d_{des})}{k^{v}_{follow}} v_{\max} & , d_{des} < 1 \\ 0 & , \text{otherwise} \end{cases} \quad (5)$$

with $d_{des} = |d_{wall} - d_{wall_{des}}|$ where d_{wall} is the measured distance to the wall and $d_{wall_{des}}$ is the desired distance to wall. When the vehicle is at the desired distance from the wall, then it navigates at the reference speed, otherwise the speed is decreased up to stopping the vehicle.

V. Behaviour Orchestration

As mentioned in section III, the *Task Manager* receives a task from the *Service Manager*, splits the task into elementary operations and then orchestrates and manages their execution for the given task.

Fig. 5 depicts the *Task Manager's* operations orchestrator. It starts by checking its state (in the form of the last executed operation, Last Goal). Then, it executes the complementary operation of the current state, e.g. if the vehicle is parked it executes a *Return from Park*. One should point out that although those *Return from...* operations are all very similar in terms of motor behaviour, there are important differences in terms of fork movement and cargo existence that also impose different safety procedures, that require the existence of three different operations. Next, the *Navigation* operation moves the stacker from point A to point B across a sequence of waypoints. At the destination another slow speed precise manoeuvre is performed. If the goal is to load a pallet, for instance, then the operations *Go to Pick* and *Load/Unload* are executed in sequence, with the former manoeuvring the stacker such that inserts the forks in the pallet and the later rising the forks to the desired height. Upon task completion, the *Task Manager* reports success to the *Service Manager* and waits for a new task. Additionally, the *Task Manager* should also be aware of information, coming from the *Safety Manager*, regarding safety violations and errors that occur in the stacker. When detected, the *Task Manager* interrupts any of the operations in execution and waits for a possible resolution. If the issue is solved within a time frame (e.g. a moving obstacle that "disapears"), it resumes the interrupted operation. Otherwise, it cancels the tasks and reports the problem to the *Service Manager*.

Every elementary operation is designed as a state machine that can be halted at any time, by action of the *Task Manager* and in case a fault has been detected. As an example, one presents the internals of the *Go to Pick* operation, which is the most complex of all the *Return from...* and *Go to...* operations.

The *Go to Pick* operation starts when stacker is near the pallet location. Its purpose is to make the stacker insert the forks in the pallet pockets. Fig. 6 depicts its state machine together with a representation of each state. *Point of service* represents the expected location of the pallet to be picked. The first step aims at making the stacker be perpendicular to the pallet. For that purpose, it activates the follow wall behaviour (refer to Sec. IV) until it reaches target 1. Then the stacker moves backwards, executing target following, towards target 2 (in step 2). Next, the stacker aligns itself with the expected orientation of the pallet, by turning on the spot (step 3). After, the stacker moves backwards, in the pallet's direction, and stops at distance that ensures no operator entrapment between the pallet and the forks (step 4) and turns off the rear safety scanner and lowers the forks (step 5). Only at this point the stacker is able to use its pallet perception module, to correct the misalignments of the pair staker-pallet by acquiring local information (step 6). When aligned, the stacker moves straight backwards to guarantee full fork insertion.

Go to Drop and *Go to Park* operations are similar to *Go to Pick*, except for the states relating to the correction with misalignments with the pallet, which are non existent in these operations, and the time at which the forks move. *Return from...* operations have a very simple implementation. They are either a one or two step operation, with fork movement and move forward motion, respectively, dependent on the specific operation. The *Navigation* has been previously developed and was presented in [8].

VI. Experimental Results

The proposed solution was implemented in the real stacker vehicle, following the method reported in [8] (to which one refers for more details on this). Equation 1 gives directly the angular velocity ω, with the positioning system providing the stacker's heading direction, ϕ. Path velocity, v, is obtained applying the forward Euler method to Eq. 2. The values for the actuation variables are an outcome of computing the inverse kinematics model of the vehicle (e.g., [6]): $\theta_{steer} = \tan^{-1}(\omega L_1/v)$ and $v_{steer} = \sqrt{v^2+\omega^2 L_1^2}$, where L_1 (see 2).

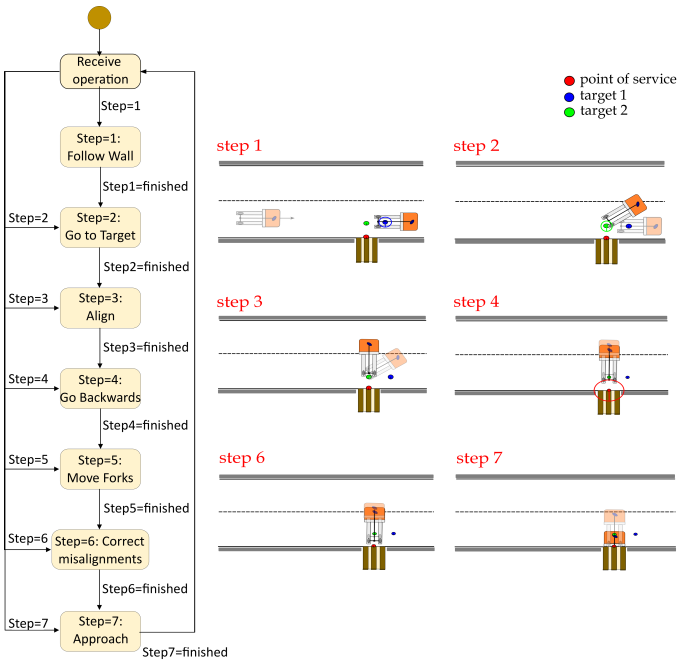

Fig. 6. *Go to Pick* state machine (on the left) and a representation of each state (on the right).

Parameters of the dynamics are tuned also to guarantee that $|\theta_{steer}| < \pi/2$ and $v_{steer} < V_{steer,max}$.

The experimental results shown here were collected in a warehouse with the layout depicted in Fig. 7. It consists of one park zone, two pick and two drop zones. It also has a crosswalk (that imposes speed limitations) and a door that narrows the corridor (so that only one lane traverses at the same time). The results also include links to videos showing the staker in operation in different task scenarios.

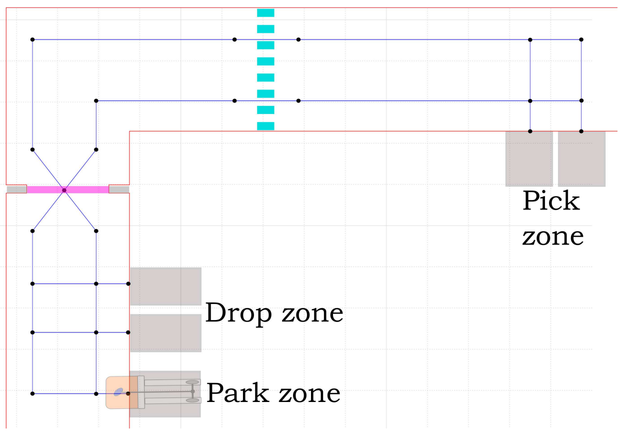

Fig. 7. Scenario layout where the experiments were conducted. The blue dots and represent the waypoints for the full service.

A. Experiment 1: full service

The full service is divided into three tasks: i) depart from parking and pick the assigned pallet; ii) unload the pallet at the assigned drop location; ii) return the stacker to the parking zone.

1) Task 1 - Load Pallet: Fig. 8 depicts the path travelled by the stacker during this task's execution, and Fig. 9 shows its path velocity. Different path colours represent different operations in execution. It starts with the vehicle leaving safely from the park zone, by executing operation *Return from Park*, depicted in blue. Next, the stacker navigates through the way-points towards the pick location (black path). Then the precise manoeuvre of inserting the forks in the pallets is executed by the activation of a *Go to Pick* (green path). The task ends when the pallet sensor detects a full insertion. The path's data is obtained directly from the localization module and referenced to the stacker's CRP (one can also see an overlay of the stacker's position in several different moments). It can be seen that the vehicle kept on its lane at all times, except when crossing the door (equivalent to a narrow passage) and around corners where, due to its large dimensions and the requirements to keep a minimum safe distance to lateral objects, it invaded the other lane. Also in this scenario, the stacker was only able to accelerate freely for a short distance, right after the cross-walk (between E and F time slots).

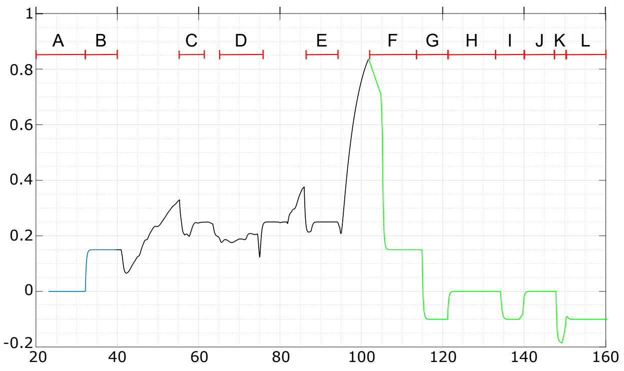

Fig. 9. Stacker's path velocity during the load task of full service, including *Return from Park* (blue line), *Navigation* (black line) and *Go to Pick* (green line) operations. $[A]$ - *Return from Park* step 1: move forks; $[B]$ - *Return from Park* step 2: return previous position; $[C]$ - vehicle through the door; $[D]$ - right turn; $[E]$ - crosswalk area; $[F]$ - *Go to Pick* step 1: follow wall; $[G]$ - *Go to Pick* step 2: go to target; $[H]$ - *Go to Pick* step 3: align; $[I]$ - *Go to Pick* step 4: go backwards; $[J]$ - *Go to Pick* step 5: move forks; $[K]$ - *Go to Pick* step 6: correct misalignment; $[L]$ - *Go to Pick* step 7: approach.

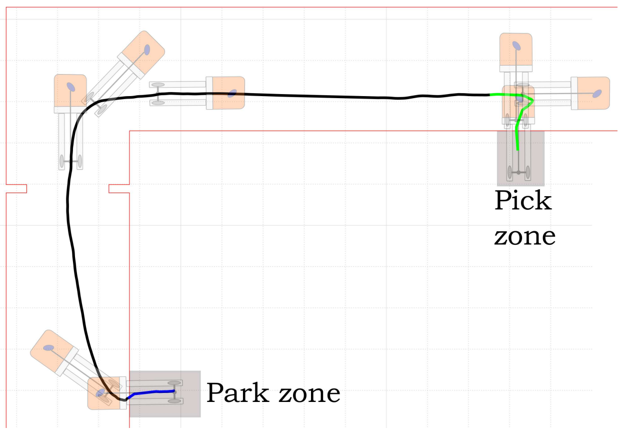

Fig. 8. Path travelled by the stacker during the load task of full service (https://youtu.be/9J4aLRy2-Mc).

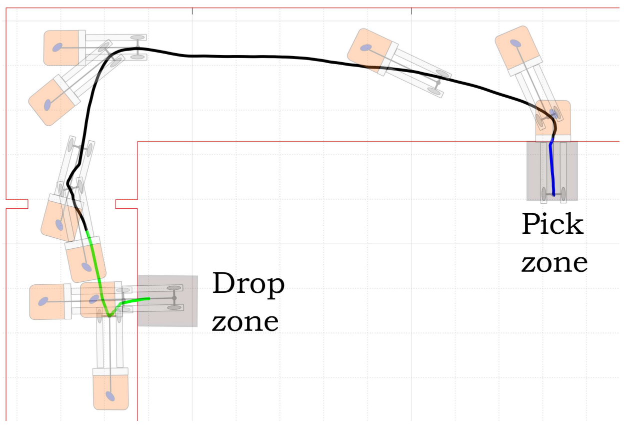

Fig. 10. Path travelled by the stacker during the unload task of full service (https://youtu.be/9J4aLRy2-Mc).

2) Task 2 - Unload Pallet: The stacker starts by executing a *Return from Pick*, which is then followed by a *Navigation* towards the drop zone. Upon arrival, the stacker activates the *Go to Drop* operation in order to unload the pallet. Fig. 10 plots the path travelled by the stacker.

3) Task 3 - Park: This task executes, in sequence, the operations *Return from drop*, *Navigate* and *Go to Park*, and the travelled path is represented in Fig. 11.

B. Experiment 2: Stacker operation cancelled due to absent pallet in loading zone

The second experiment validates the stacker's behaviour when the pallet is absent from the loading area. Fig. 12 shows some snapshots of this experiment. Already executing the *Go to Pick* operation and ready to receive information from the pallet detection module (step 6-7, of the state machine), if no

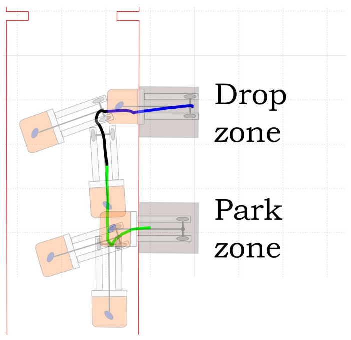

Fig. 11. Path travelled by the stacker during the parking task of full service (https://youtu.be/9J4aLRy2-Mc).

pallet is detected within a time frame, then the operation is halted and notice of failure is issued to the *Task Manager*, which on its turn informs the *Service Manager* that the task could not be completed. Then, the stacker waits for a new service.

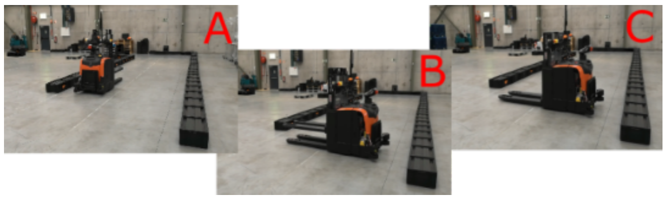

Fig. 12. Snapshots of experiment 2: Stacker operation cancelled due to absent pallet in loading zone. (https://youtu.be/8clcmjCcc3E).

C. Experiment 3: Stacker operation cancelled due to failure in pallet detection

The third experiment simulates a sustained failure in pallet detection while performing the loading task (see Fig. 13). After initial pallet detection (panel B) and the stacker started to align with it (panel C) an operator removes the pallet from the stacker sight (panel D) in such a way that it is not detected again. Again, after a time frame of missing detection, a fault is triggered and the operation is halted.

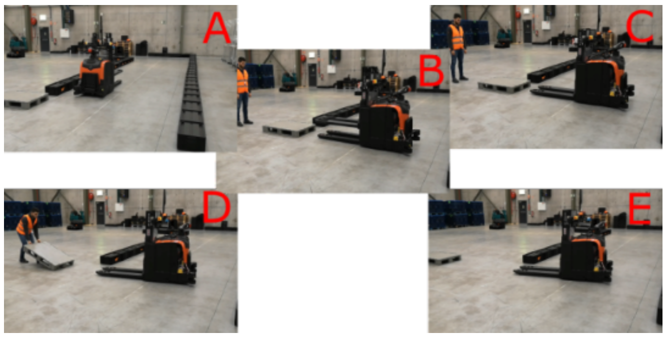

Fig. 13. Snapshots of experiment 3: Stacker operation canceled because a human operator takes out the pallet in the moment it loads the pallet. (https://youtu.be/s7E6KkD0ork).

D. Experiment 4: Stacker's safety of operation

This final experiment, with snapshots in Fig. 14, demonstrates the stacker's safety of operation. For this purpose it will be disturbed by human operators (panel A$\rightarrow$C and G$\rightarrow$L) and challenged by the appearance of sudden obstructions (panel D$\rightarrow$F). In all this situations, the stacker stops at a safe distance and only resumes the assigned task when the disturbance disappears (check the video with link on figure caption).

VII. Conclusion

This paper proposed a solution to endow stackers with autonomous operation in workspaces that are shared with other vehicles (both autonomous and human driven) and human workers. Behaviours for target acquisition and wall following, designed as dynamic systems of path velocity and

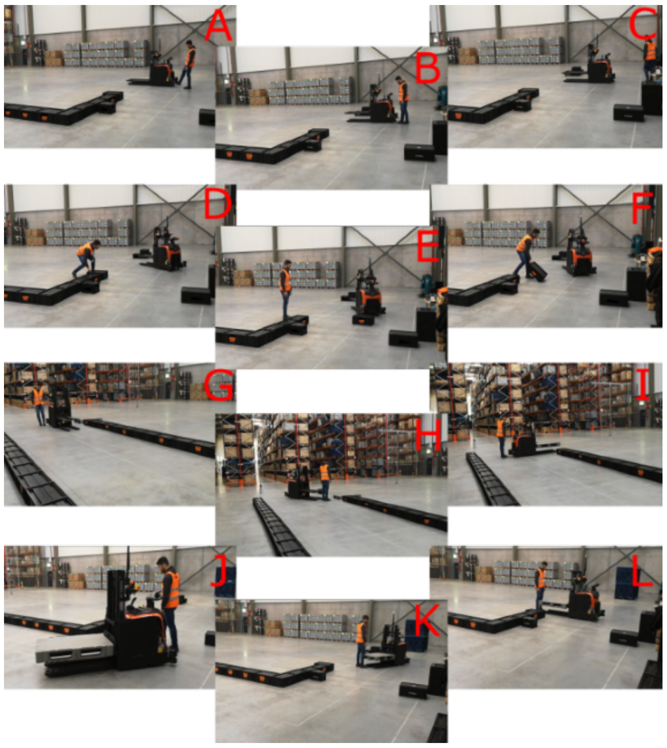

Fig. 14. Snapshots of experiment 4: Stacker operation disturbed by the presence of a human and by obstacles placed in its path. (https://youtu.be/ZvzDYxlPQfE).

heading direction, were the at the foundation. These are then orchestrated to generate more complex behaviour, such as exiting from parking locations or executing pallet picking and dropping manoeuvres, taking into account safety procedures. This means that the protection areas around the vehicle are dynamically changed according to the current actions in execution. Results showing the stacker performing full services and robustness against pallet detection have been presented. The results also include links to videos showing the staker in operation. Future work includes testing the proposed solution in more challenging scenarios, e.g. dynamic production plant characterized by the co-existence several stacker vehicles and larger number of human operators.

References

[1] M. M. Aref, R. Ghabcheloo, A. Kolu, M. Hyvönen, K. Huhtala, and J. Mattila, "Position-based visual servoing for pallet picking by an articulated-frame-steering hydraulic mobile machine," in *2013 6th IEEE Conference on Robotics, Automation and Mechatronics (RAM)*. IEEE, 2013, pp. 218–224.

[2] U. Behrje, M. Himstedt, and E. Maehle, "An autonomous forklift with 3d time-of-flight camera-based localization and navigation," in *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. IEEE, 2018, pp. 1739–1746.

[3] E. Bicho, *Dynamic Approach to Behavior-Based Robotics: design, specification, analysis, simulation and implementation*. Shaker, 2000.

[4] Z. Chen, J. Fu, X.-W. Tu, A.-L. Yang, and M.-R. Fei, "Real-time predictive sliding mode control method for agv with actuator delay," *Advances in Manufacturing*, vol. 7, no. 4, pp. 448–459, 2019.

[5] D. Giglio, "Task scheduling for multiple forklift agvs in distribution warehouses," in *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*. IEEE, 2014, pp. 1–6.

[6] Y. Kanayama, “Two dimensional wheeled vehicle kinematics,” in *Proc. of the 1994 IEEE Intl Conf on Robotics and Automation*. IEEE, 1994, pp. 3079–3084.

[7] D. Lecking, O. Wulf, and B. Wagner, “Variable pallet pick-up for automatic guided vehicles in industrial environments,” in *2006 IEEE Conference on Emerging Technologies and Factory Automation*. IEEE, 2006, pp. 1169–1174.

[8] L. Louro, T. Malheiro, P. Guimaraes, T. Machado, S. Monteiro, P. V. Silva, W. Erlhagen, and E. Bicho, “Motion control for autonomous tugger vehicles in dynamic factory floors shared with human operators,” in *IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society*, vol. 1. IEEE, 2019, pp. 5255–5262.

[9] T. Machado, T. Malheiro, S. Monteiro, W. Erlhagen, and E. Bicho, “Attractor dynamics approach to joint transportation by autonomous robots: theory, implementation and validation on the factory floor,” *Autonomous Robots*, vol. 43, no. 3, pp. 589–610, 2019.

[10] H. Martínez-Barberá and D. Herrero-Pérez, “Autonomous navigation of an automated guided vehicle in industrial environments,” *Robotics and Computer-Integrated Manufacturing*, vol. 26, no. 4, pp. 296–311, 2010.

[11] A. Mohammadi, I. Mareels, and D. Oetomo, “Model predictive motion control of autonomous forklift vehicles with dynamics balance constraint,” in *2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. IEEE, 2016, pp. 1–6.

[12] J. Nygards, T. Hogstrom, and A. Wernersson, “Docking to pallets with feedback from a sheet-of-light range camera,” in *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)(Cat. No. 00CH37113)*, vol. 3. IEEE, 2000, pp. 1853–1859.

[13] G. Rogachev, M. Patkin, and N. Rogachev, “Fuzzy optimization in the problem of forklifts path planning,” in *2018 International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon)*. IEEE, 2018, pp. 1–5.

[14] G. Schöner, M. Dose, and C. Engels, “Dynamics of behavior: Theory and applications for autonomous robot architectures,” *Robotics and Autonomous Systems*, vol. 16, pp. 213–245, 1995.

[15] M. Seelinger and J.-D. Yoder, “Automatic visual guidance of a forklift engaging a pallet,” *Robotics and Autonomous Systems*, vol. 54, no. 12, pp. 1026–1038, 2006.

[16] T. A. Tamba, B. Hong, and K.-S. Hong, “A path following control of an unmanned autonomous forklift,” *International Journal of Control, Automation and Systems*, vol. 7, no. 1, pp. 113–122, 2009.

[17] P. M. Vaz, R. Ferreira, V. Grossmann, and M. Ribeiro, “Docking of a mobile platform based on infrared sensors,” in *ISIE’97 Proceeding of the IEEE International Symposium on Industrial Electronics*, vol. 2. IEEE, 1997, pp. 735–740.

[18] J. Villagra and D. Herrero-Pérez, “A comparison of control techniques for robust docking maneuvers of an agv,” *IEEE Transactions on Control Systems Technology*, vol. 20, no. 4, pp. 1116–1123, 2011.

[19] P. R. Wurman, R. D’Andrea, and M. Mountz, “Coordinating hundreds of cooperative, autonomous vehicles in warehouses,” *AI magazine*, vol. 29, no. 1, pp. 9–9, 2008.

[20] J. Xiao, H. Lu, L. Zhang, and J. Zhang, “Pallet recognition and localization using an rgb-d camera,” *International Journal of Advanced Robotic Systems*, vol. 14, no. 6, p. 1729881417737799, 2017.

[21] J. Yu, P. Lou, X. Qian, and X. Wu, “An intelligent real-time monocular vision-based agv system for accurate lane detecting,” in *2008 ISECS International Colloquium on Computing, Communication, Control, and Management*, vol. 2. IEEE, 2008, pp. 28–33.