Task-level Collaborative Ad-hoc Autonomous Guided Vehicles (AGV) for Efficient Warehouse Logistics

Integrative Masters Project
in
Master of Computer Science

L. Schöb 13-915-343
C. Zweifel 15-614-159

17th of January 2025

Abstract

This paper investigates task-level collaboration among Automated Guided Vehicles (AGVs) to enhance efficiency in warehouse logistics. Traditional AGV systems primarily focus on individual task completion, with limited collaboration restricted to crash avoidance and path optimization. This study explores a novel approach, where AGVs dynamically divide and coordinate subtasks to optimize collective performance. Using experimental setups, we evaluate the performance of single-robot and collaborative modes across two distinct path environments. Results demonstrate that task-level collaboration reduces idle time and improves throughput in certain scenarios, particularly as task complexity and path length increase. It identifies challenges such as inefficiencies during freight handovers and system scalability with the used hardware. The findings provide insights into the design and deployment of collaborative AGV systems, paving the way for scalable solutions in task-saturated environments, where operational demands continue to rise alongside global logistics growth.

1 Introduction

The advent of automated guided vehicles (AGVs) has revolutionized warehouse logistics by automating item retrieval and transportation tasks, thereby enhancing operational efficiency. The first AGVs, introduced in 1950s, were rudimentary systems designed for specific industrial applications Wurman et al. (2008). Over the decades, AGV technology has evolved significantly, diversifying in size, functionality, and deployment strategies to meet the unique demands of different industries. For instance, large AGVs are often used by automotive manufacturers for transporting heavy components as shown by Swisslog (2025), whereas smaller fleets, such as those developed by Ferag (2025), excel in handling lightweight electronic goods.

Despite these advancements, most AGV systems today operate with limited collaborative capabilities. Their cooperation is largely restricted to crash avoidance, path optimization, or queuing protocols, ensuring minimal disruption in densely occupied environments (Cardarelli et al., 2017). While these systems are effective, their task completion strategies remain largely autonomous, leading to inefficiencies in complex, multi-task environments.

This paper explores a novel approach that has been employed by Frauenfelder (2024): task-level collaboration among AGVs. Unlike conventional systems, task-level collaborative AGVs distribute tasks dynamically among multiple vehicles, allowing them to share subtasks and optimize their collective performance. This capability enables the fleet to reduce idle time, balance workloads, and adapt to varying operational demands. For example, instead of a single robot completing an entire retrieval operation, the task could be subdivided: one robot retrieves the item, while another transports it to its destination.

The key research question underpinning this study is: Are task-level collaborative AGV systems inherently more efficient than conventional AGV systems that primarily focus on navigation and collision avoidance? To address this question, we conducted an extensive evaluation of AGV performance under two operational modes—single-robot and collaborative—across two distinct path environments. The study compares the execution times and task efficiencies of individual robots performing tasks independently against two collaborating robots that dynamically divide and coordinate subtasks.

By incorporating these experimental setups, this study highlights the influence of path environments on AGV performance, particularly in terms of how collaboration can mitigate bottlenecks and enhance throughput. The findings provide critical insights

into the conditions under which task-level collaboration becomes most advantageous, setting the stage for more scalable and adaptive deployment in warehouse logistics.

As global logistics demand continues to rise—with the market projected to expand at a CAGR of 7.2% from 2024 to 2030—developing innovative solutions for improving warehouse efficiency is more critical than ever (Research, 2025). Collaborative AGV systems represent a promising step toward achieving this goal, offering a scalable, adaptive solution for task-saturated environments.

Figure 1: *Large AGV agents deployed at AMAG to transport car tires*

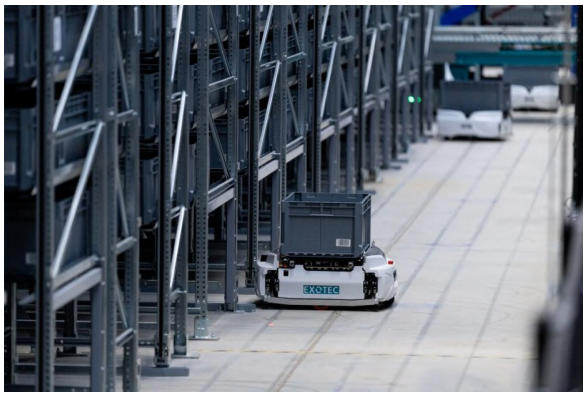


Figure 2: *Smaller AGV agents that transport electronic goods at Digitec Galaxus*

2 Background & Related Work

The development and deployment of task-level collaborative Automated Guided Vehicle (AGV) systems have garnered significant attention in recent years. This section provides an overview of foundational concepts, existing frameworks, and methodologies relevant to collaborative multi-AGV systems.

2.1 Evolution of Multi-AGV Collaboration

Frauenfelder (2024) highlights the potential of task-level collaboration in AGVs for improving warehouse logistics. The study investigates how dividing tasks into subtasks among robots could theoretically improve efficiency, but notes challenges in real-world implementation, such as inefficiencies in freight handovers.

Liu et al. (2024) present the MARTCO framework, which focuses on real-time path optimization for over 100 AGVs in dense warehouse environments. Their improved A* algorithm and priority rules exemplify how AGVs can collaborate dynamically to avoid conflicts, paving the way for large-scale applications.

2.2 Current Approaches in Collaborative Multi-AGV Systems

Several methods for multi-AGV collaboration are categorized into centralized and decentralized approaches. Centralized systems, such as Amazon Robotics' Kiva systems, rely on a central controller for assigning tasks and paths. However, decentralized systems allow AGVs to independently determine their paths, as proposed in Liu's MARTCO framework, which is computationally efficient for large-scale environments.

2.3 Research Gaps and Opportunities

Despite advancements in AGV collaboration, existing systems often focus on collision avoidance and path optimization rather than true task-level collaboration. Frauenfelder (2024) identifies this gap, emphasizing the need for AGVs to dynamically share and execute subtasks to improve warehouse throughput. Additionally, MARTCO demonstrates scalability but raises questions about its adaptability to scenarios requiring simultaneous task-sharing and real-time decision-making.

The following work focuses on the approach shown in Frauenfelder's master thesis.

3 Design

On the foundation of chapter 2 we have designed two paths to evaluate the baseline, so-called conventional approach, and the collaborative approach, where a task is split into subtask and worked in parallel by two AGVs. Furthermore, this chapter explores the scenario, where the AGV are operating in and illustrates the capabilities of the AGV.

3.1 Scenario & Environment

The prototype developed in this project focuses on the AGV's software but also accounts for other key elements: the operating environment, the WMS for

task scheduling, and the freight to be transported. Each component's main characteristics and capabilities are described in the following subsections.

Task The task given two the AGV system is to transport one piece of freight from one waypoint to another one. In our system, the waypoints are marked by colored, square markers.

Warehouse Management System In practice, a warehouse management system (WMS) creates tasks which are later fulfilled by the AGV system. In our prototype settings, the tasks are hardcoded in the AGV agent.

3.2 Hardware

This chapter describes our hardware, the JETANK by Weaveshare which we use as AGV agent. The JETANK is highly versatile robotics platform designed for advanced AI applications. Equipped with a 8MP camera, two tracks and one multi-axis arm, the JETANK is able to follow a line, identify waypoints and move freight packages around.

Figure 3: *JETANK*

Freight Wood cubes could be used to simulate freight.

3.3 Path Setups

Baseline Setup Figure 4 illustrates the baseline design for the JETANK robot evaluation. Waypoint 1 serves as the entry point where goods arrive, while waypoint 5 is the exit point where goods are dispatched. The path connecting these waypoints is designed to support both operational modes: Conventional, where a single robot transports goods along the entire path, and Collaborative, where tasks are divided into subtasks, enabling multiple robots to coordinate and share the workload efficiently. This setup ensures a fair comparison of task execution performance between the two approaches. Figure 1 shows the necessary actions each approach needs to accomplish the task.

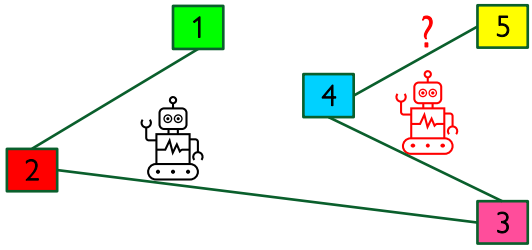


Figure 4: *Baseline setup*

Conventional	**Collaborative**	
Robot A	**Robot B1**	**Robot B2**
Drive	Drive	Drive
Pick up	Pick up	Idle
Transport	Transport	
Drop off	Drop off	
		Pick up
		Transport
		Drop off

Table 1: *Required Actions for single task execution*

Further experimental Setup In the further experimental design as depicted in Figure 5, two JETANK robots can work collaboratively by sharing tasks, leveraging parallelism as an obvious advantage since two distinct paths lead to the same destination. This design allows for efficient task distribution and reduced execution times through simultaneous task processing. However, this setup was primarily developed due to hardware limitations in waypoint recognition, which caused issues in more complex configurations. The simplified layout ensures clear waypoint detection, minimizing the risk of misinterpretation, and allowing the robots to operate reliably under controlled experimental conditions.

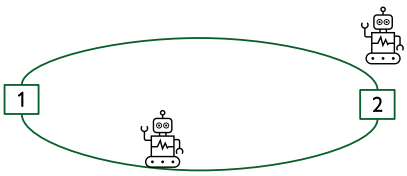


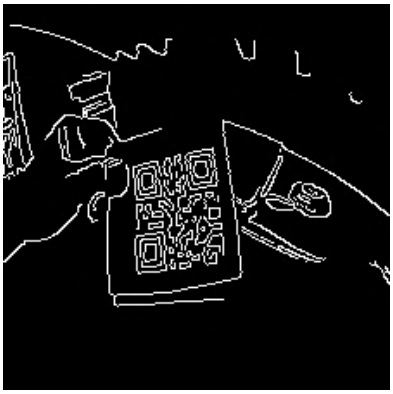
Figure 5: *Further experimental design*

4 Implementation

This chapter focuses on overcoming key challenges in the development process. Initially, QR code detection was explored but proved unsuccessful due to inconsistent recognition. As an alternative, ArUco markers were tested, but software compatibility issues prevented their effective implementation. Consequently, a third approach involving custom-designed color markers was pursued, which finally yielded positive results. With reliable marker detection in place, development progressed to refining the track design and establishing precise waypoint detection criteria, including defining specific triggers for waypoint recognition. Lastly, a more complex future track design was proposed but, due to its intricacy, could not be successfully implemented within the project's timeframe. This iterative process highlights the adaptive problem-solving efforts undertaken throughout the project.

(a) *color image*

(b) *grayscale image*



(c) *edges image*

Figure 6: *QR code*

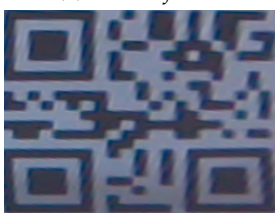
4.1 QR Code Detection

The points where package loading and unloading occur are indicated by QR codes. To detect and decode these codes, the QRCodeDetector class from the OpenCV framework is utilized. However, the limited camera capabilities of the JETANK reduce detection reliability, requiring improvements in the detection algorithm or image preprocessing. Figure 6 illustrates the QR code processing pipeline, showing the transition from the original image to grayscale conversion and finally to edge detection using cv2.Canny.

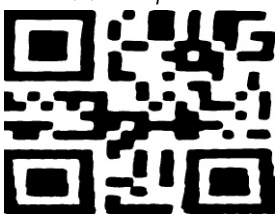
It is important to note that the current setup runs OpenCV version 4.1.1 on Jupyter Lab with Python 3.6.9. The authors compared this version to OpenCV 4.10.0.84, hoping for improved reliability in QR code recognition. However, after thorough testing and comparison, no significant difference was observed in the performance of the detectAndDecode method between the two versions.

Due to the QR code being rarely aligned perpendicularly to the camera, a transformation was applied to adjust the image orientation for better detection. Additionally, a binary image was created from the original 8-bit RGB image to enhance contrast. The transformation was performed using the four cor-

(a) *camera frame*



(b) *transposed*



(c) *binary image*

Figure 7: *QR code improvement*

ner points of the QR code rectangle returned by the detect method. Despite these efforts, passing the binary image to the decode method still frequently resulted in a decoded value of "None" instead of the expected result. Interestingly, using a standard QR code decoding application, such as the iPhone Camera App, successfully decoded the QR codes without any issues. Refer to Figure 7 for the detailed steps involved.

Given the challenges encountered with QR code detection, alternative approaches should be considered. In the author's opinion, two viable options remain: using **color patches instead of QR codes** or implementing **ArUco markers** as mentioned in Frauenfelder (2024).

4.2 ArUco markers

Adapting the code to recognize ArUco markers would necessitate updating both the OpenCV framework and the current Python distribution to ensure compatibility and access to the latest features. This update could enhance detection reliability and improve the overall performance of the system by leveraging more advanced image processing capabilities available in newer OpenCV versions. Efforts to update the system to support ArUco marker recognition ultimately failed. The primary reason was the extensive dependencies of the JetBot library, which made it challenging to adjust the system to newer versions of Python, Jupyter Server, and OpenCV. Due to these compatibility constraints, implementing ArUco marker detection could not be pursued further.

4.3 Color markers

Rather than attempting to update the system to support ArUco markers, the decision was made to proceed with color detection. The risk of spending additional time on system configuration instead of focusing on actual task development was deemed too high, especially after already investing more than two weeks. Figure 8 shows the hue, saturation, and value (HSV) of the pixels inside the detected square. These HSV values will be used as reference points for detecting different waypoints along the path.

(a) *yellow*

(b) *green*

(c) *cyan*

(d) *blue*

(e) *magenta*

(f) *red*

(g) *track line*

(h) *track background*

Figure 8: *HSV values*

4.4 Controlling the robot on paths

To control the AGV in path transer learning was used by using pre-trained ResNet-18 model. After capturing training data sets in different environment and applying it to ResNet, the JETANK is able to follow paths or lines.

To get the best results, around 200 different pictures should be included in the dataset. Additionally, we trained with 50 to 70 epochs. Depending on the training set, we have test losses of 0.005827.

Listing 1: *Function used to evaluate*

```
model = models.resnet18(pretrained=True)

model.fc = torch.nn.Linear(512, 2)
device = torch.device('cuda')
model = model.to(device)
```

In the case for the JETANK it is important to note, that Cuda needs to be used to train the dataset, otherwise the powerful GPU of the JETANK can not be utilized. This setting can vary, if tested on a notebook or another AGV system.

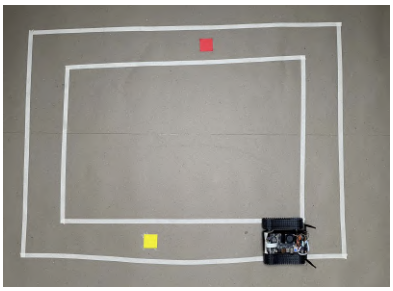
Listing 2: *Function used to evaluate*

```
def execute(change):
    global angle, angle_last,
        last_detection_time
    image = change['new']

    # Step 1: Check if within cooldown
        period
    current_time = time.time()
    if current_time -
        last_detection_time >
        detection_cooldown:
        color_detected = False
        _, color_detected = findColor(
            image, colorLower, colorUpper
            )
        if color_detected:
            # Update the last detection
                time to start the
                cooldown period
            last_detection_time =
                current_time

            tankTurn()

            # Return and do next cycle
            return
```

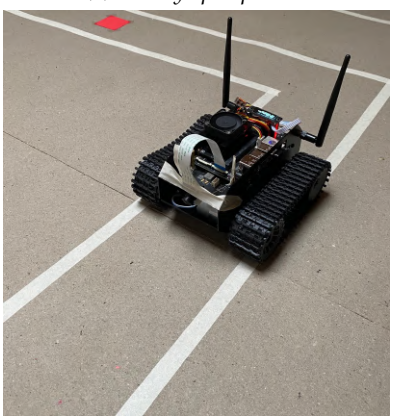
Combing this with color detection shown above, the JETANK start to execute the assigned task and measures the total completion time. When a dropoff point is reached, the JETANK does a tank turning, by moving 180 degrees that it can continue on the next task execution.

4.5 Experiments on the track

This section describes the track setup used for robot navigation. Figure 9 shows the baseline configuration, designed to guide the robot along a predefined path while detecting specific waypoints. For simplicity, two waypoints of different colors are used as targets. Data collection for training the model is conducted directly on the track. As shown in Figure 9c, the robot's desired position is marked with a green circle. Approximately 400 labeled images are prepared by manually clicking the intended robot positions using a controller, ensuring precise and relevant training data for waypoint detection.



(a) *bird eye perspective*



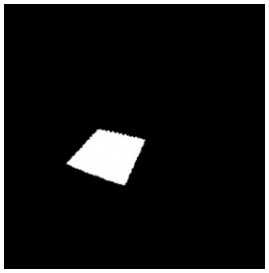
(b) *robot on track*

(c) *data collection process*

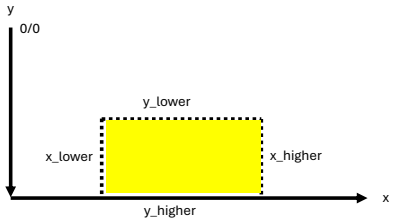
Figure 9: *Baseline track*

Waypoint detection Figures 10a and 10b show the robot's perspective of a yellow waypoint. The waypoint detection mechanism works based on a threshold function defined by specific coordinates: x_{lower}, x_{higher}, y_{lower}, and y_{higher}. This threshold area, symbolically illustrated in Figure 10c, defines a rectangular detection zone within the robot's camera view. The reference point $X/Y(0/0)$ is located at the top-left corner of the image. When a sufficient portion of the masked yellow waypoint image falls within this defined threshold area, the waypoint is considered "hit," allowing the robot to perform an action such as picking up or dropping off a package.

(a) *yellow waypoint*



(b) *yellow waypoint mask*



(c) *waypoint threshold area*

Figure 10: *Waypoint detection*

Future track Figure 11 shows an alternative track designed for future testing, featuring six waypoints and eight vertices. This track was originally intended for final performance evaluations due to its complexity and potential for testing advanced navigation and task coordination scenarios. However, due to technical challenges encountered during the project—particularly with waypoint detection reliability and system integration—testing on this track could not be completed within the project timeframe. Future work could focus on refining the detection algorithms and improving system stability to enable successful navigation and performance evaluation on this more complex track.

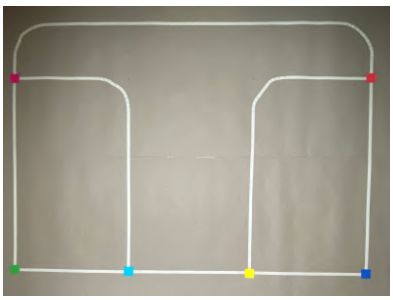


Figure 11: *Future track setup*

5 Evaluation & Results

The performance of JETANK robots was evaluated in two operational modes: Conventional, where a single robot independently completes a task, and collaborative, where a robot assigns subtasks to available robots in the environment. Task execution times were measured for both modes under identical conditions. The results highlight differences in efficiency, demonstrating the potential benefits of collaborative task distribution in reducing overall task completion time.

5.1 Baseline track results

In Table 2, the performance results for both conventional and collaborative modes are presented. Initially, the collaborative mode demonstrates a 75% slower execution time compared to the conventional mode when performing a single task. However, as the number of baseline runs increases, the performance gap narrows remarkably. After three runs, the collaborative mode is only 2% slower. This improvement highlights the adaptive efficiency of the collaborative mode, suggesting that it benefits from repeated task execution due to better load distribution and reduced idle times between AGVs. Consequently, the collaborative approach becomes increasingly viable as the system scales and tasks accumulate.

	Conventional	**Collaborative**
Single Execution	1x Baseline	75% slower
Consecutive Execution	2x Baseline	4.5% slower
Consecutive Execution	2x Baseline	2% slower

Table 2: *Baseline track results*

The trend of these results can be explained, when one considers the task execution of both approaches on a step-by-step basis.

Steps	**Conv.**	**Collaborative**	
	Robot A	**Robot B1**	**Robot B2**
Drive	0	0	2
Pick up	1	1	1
Transport	4	2	2
Drop off	1	1	1
Idle	0	0	2
Total steps	6	12 (8)	

Table 3: *Total of steps for single task execution*

As 4 out 12 steps are executed in parallel in the collaborative approach, the total number of steps for this approach is actually 8. If the same task is

repeatedly executed, where idle time will decrease, as a more optimal waypoint to exchange the freight is found.

5.2 Further experimental track results

Figure 12 illustrates the adjusted experimental track. Due to various software development challenges on the JETANK platform, the track complexity was reduced to ensure stable operation and reliable testing. The task involves picking up a parcel at one waypoint and delivering it to another, with waypoints marked by green or yellow colors on the track. This task can be performed either in the conventional mode, where a single robot completes the entire task, or in the collaborative mode, where multiple robots coordinate to share the workload. For detailed results in conventional mode with absolute values and all algorithm parameters, please refer to Table 4. A comparison of relative performance results between the conventional and collaborative modes is presented in Table 5, highlighting the efficiency gains achieved through task-sharing and coordination in the collaborative mode.

Run	Time [s]	speed	steering	steering kd
1	54.52	0.40	0.21	0.00
2	50.60	0.40	0.21	0.00
3	55.94	0.40	0.21	0.00
4	58.26	0.40	0.21	0.00

Table 4: *Further experimental absolute results (conventional mode)*

	Conventional	**Collaborative**
Single Execution	1x Experiment	n.A.
Consecutive Execution	2x Experiment	100% faster

Table 5: *Further experimental results (both modes)*

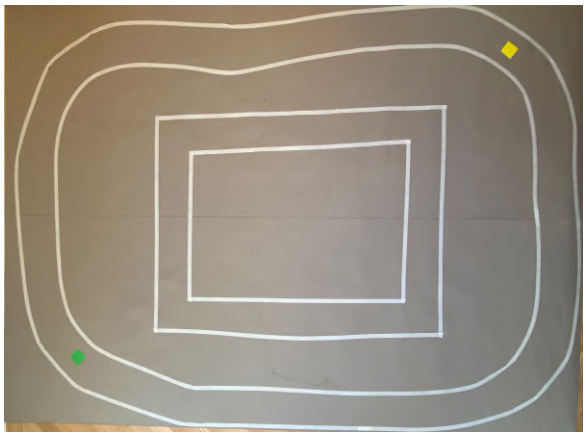


Figure 12: *Further experimental track*

6 Conclusion & Discussion

The performance of pick-up and drop-off operations is highly influenced by environmental factors and the mechanical characteristics of individual robots. The layout of the warehouse, including the floorplan and travel distances, significantly impacts the cooperation and efficiency of robotic systems.

Relatively longer paths tend to benefit from task-level cooperation, as it reduces idle times and optimizes task distribution. Conversely, relatively shorter paths may experience a decline in performance due to delays introduced by the robotic arms during parcel handling.

However, additional challenges arise in more complex environments, particularly when a node or edge within the system becomes blocked. These issues highlight the importance of optimizing path planning and addressing potential bottlenecks to enhance overall system efficiency and reliability.

- **Parcel pickup** (robotic arm): Although parcel pickup was simulated rather than performed physically, it still required a considerable amount of time. As the path length increases, the AGV's actual movement time becomes longer, making the fixed pickup time less significant in comparison.
- **Path length**: The path length affects performance in the following way: the shorter the distance between two waypoints, the less efficient and slower the collaborative mode becomes. The challenge lies in determining the optimal path length where the collaborative mode outperforms the conventional approach.
- **Path environment**: The baseline track is defined by two sidelines. The AGV's speed depends on the ambient lighting conditions, with better illumination allowing faster movement. Additionally, sharp turns are more challenging to

detect than smooth curves.

- **AGV speed**: The AGV's task execution speed was set to 40%. When the speed was increased, tracking errors and deviations from the path rose exponentially. This indicates significant potential for improvement in speed optimization and error reduction.
- **AGV AI model performance** for line following (Controller speed): The trained model can be enhanced by collecting more data samples and conducting additional training rounds. Furthermore, physically constraining the AGVs to the track, rather than relying solely on camera-based guidance, would significantly improve stability.
- **Waypoint detection**: Currently, waypoints are indicated by colored markers, which is prone to errors and heavily affected by ambient brightness. In future developments, using ArUco markers would greatly enhance performance and reliability.

References

Cardarelli, E., Digani, V., Sabattini, L., Secchi, C., and Fantuzzi, C. (2017). Cooperative cloud robotics architecture for the coordination of multi-agv systems in industrial warehouses. *Mechatronics*, 45:1–13.

Ferag (2025). Neue logistiklösung bei digitec galaxus: Einführung des ferag skyfall taschensorters. Accessed: 2025-10-25.

Frauenfelder, T. (2024). Task-level collaboration between automated guided vehicles (agv) for efficient warehouse logistics.

Liu, M., Qiao, Y., and Wu, N. (2024). Efficient multi-agv real-time collaborative operation in large-scale intelligent warehouses. *IEEE Transactions on Intelligent Vehicles*. Accepted for publication, DOI: 10.1109/TIV.2024.3370763.

Research, G. V. (2025). Logistics market size, share & trends analysis report. Accessed: 2025-01-13.

Swisslog (2025). Case study: Logistics perfected in warehouse and distribution center for replacement automotive parts at amag, switzerland. Accessed: 2025-10-25.

Wurman, P. R., D'Andrea, R., and Mountz, M. (2008). Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI Magazine*, 29(1):9–19.

A How to setup a JETANK

A.1 General

The setup of the robot presented several challenges. The most significant issue was that the pre-installed image on the micro SD card provided with the device was faulty. Even after successfully flashing the image, the Jupyter Lab web server on the device could not be accessed. This issue caused significant delays in the setup process. Fortunately, one team member was able to provide spare SD cards from GoPro cameras, and flashing the image onto these cards resolved the issue.

This extra effort could have been avoided if the Waveshare company had opted not to include pre-installed SD cards in the delivery, considering the faulty state of the provided cards. The company has been informed of this issue via a support ticket to prevent similar problems in the future.

Additionally, it was observed that some instructions in the official wiki contained erroneous prompts. For correct setup instructions, refer to the valid prompts provided in Listing 3.

The assembly process begins with ensuring access to the correct links and manuals. Having accurate and up-to-date documentation is critical to avoid errors during assembly and to streamline the setup process. Reliable resources ensure that each step is clearly defined, reducing ambiguity and the likelihood of issues arising during the assembly and configuration stages.

- JETANK product page
- Assembly video from A-Z Basically one should just follow this video. However, please be aware that there are some mistakes. See below for the additional prompts that are needed.
- JETANK wiki
- Nvidia Jetson Nano Developer Kit
- NVIDIA AI IOT Jetbot source code

Listing 3: *Console prompts for JETANK setup*

```

Mandatory prompts:
ssh jetbot@<ip>
sudo nmcli device wifi connect <SSID> password <PASSWORD>
sudo apt-get update
sudo apt-get install python3-setuptools
git clone https://github.com/waveshare/JETANK.git
cd JETANK/
sudo chmod +x config.sh
sudo chmod +x install.sh
./config.sh jetbot

Before doing JETANK_1_servos tutorial (do in jupyter console)
cd JETANK/
sudo ./install.sh
python3 servoInt.py

Before doing JETANK_6_gampadCtrl tutorial :
sudo apt install joystick

For reference:
sudo netstat -tuln | grep :8888 #if you have to check tcp listings
sudo shutdown now
```

A.2 Increase partition size

Unfortunately, the SD card image partition is limited to 24GB upon loading the image, regardless of the physical SD card's capacity. This restricts the available storage space, which can pose challenges for projects requiring additional memory. To utilize the full capacity of the SD card, the partition size can be adjusted by following the steps outlined in Listing 4. These instructions will guide you through resizing the partition to maximize the available space.

Listing 4: *Increase partition size to maximum*

```
Enable the GUI (via ssh the repartitioning below did not work)
cd jetbot/scripts
./re_enable_gui.sh

Check file system
df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/mmcblk1p1   59G   21G   36G  37% /
....

cd ~/jetcard/scripts/archive
look for nvresizefs.sh -> change all values with mmcblkXpX and mmcblkX to the value
    of above (here: mmcblk1p1)

If you do not have jetcard repo:
git pull https://github.com/NVIDIA-AI-IOT/jetcard.git

Run partition script
./jetson_install_nvresizefs_service.sh
reboot your system and it should be fine. Check with df -h again

Disable GUI
sudo systemctl set-default multi-user.target
Alternative for enabling GUI (not to be done)
sudo systemctl set-default graphical.target
```

A.3 Access Code repository

The code for the project is hosted in a GitHub repository. Instructions for accessing the repository are provided in Listing 5. These steps will guide you through obtaining the code and setting it up for use in your environment.

Listing 5: *Access to Github repository*

```
Checkout repo (into root)
cd
git clone git@github.com:<user name>/IMP_AGV.git

Check SSH key already there (look for id_rsa.pub or id_ed25519.pub)
ls -al ~/.ssh

Create new one if none there
ssh-keygen -t ed25519 -C "<your email address>"

Add to eval agent
eval "$(ssh-agent -s)"
ssh-add ~/.ssh/id_ed25519

Copy to clipboard and add to your github account
cat ~/.ssh/id_ed25519.pub
Add the output like 'ssh-ed25519
    AAAAC3NzaC1lZDI1NTE5AAAAINz2rw6E6MRuxxxxxxxxxxxxxxxxX5zpK luzi.schoeb@student.
    unisg.ch' to your github account.

Check changes to be commited
git status

Add, Commit, Push file
git add <file> or git add . (for all files)
```

```
git commit -m "Meaningful message"
git push
```

A.4 Create virtual python environment

In some cases, it is preferable to create a virtual environment rather than modifying the software (e.g., Python) for the entire device. Using a virtual environment allows for isolated and customizable configurations without affecting the global setup. The steps to create and use a virtual environment are outlined in Listing 6, providing a flexible solution for managing dependencies and compatibility.

Listing 6: *Python virtual environment*

```
sudo apt-get install python3-venv

Create environment
python3 -m venv myenv

Activate environment
source myenv/bin/activate

Deactivate environment
deactivate

Remove environment
rm -rf myenv

Find environment
find ~ -type d -name "myenv"
```

A.5 Installation add ons for IMP AGV baseline code

To run main.py from the IMP_AGV_baseline code, certain additional dependencies and configurations need to be set up. These steps are critical to ensure the code runs smoothly. The required commands and instructions are provided in Listing 7. Follow these steps to install the necessary add-ons and prepare the environment for executing the baseline code.

Listing 7: *Installation add ons*

```
sudo apt-get update
sudo apt-get install python3-setuptools
pip install cython
pip install --upgrade pip setuptools wheel
pip install opencv-python opencv-contrib-python --only-binary=:all:


check if numpy and scipy are available
python -c "import scipy"
python -c "import numpy"

install if not available
pip uninstall numpy scipy -y
pip install --no-binary :all: numpy scipy

Now you can install according to the readme of IMP_AGV_baseline
pip install -r requirements.txt
...
```

A.6 UDP commands

In Listing 8, several UDP commands are defined, each containing three fields: type, message, and address. For the Task_Request command, the type field is set to TASK_REQUEST, while the message field contains a JSON object with parameters start_node and end_node, indicating the task's origin and destination. The address field is currently not utilized but remains part of the structure for future scalability.

After the piping operator, the actual UDP address and port details follow. In the task request example provided, the UDP command is sent to IP address 172.20.10.4 on port 5004, ensuring that task-related data is transmitted correctly over the network.

Listing 8: *UDP commands*

```
Task_Request
echo '{"type": "TASK_REQUEST", "message": {"start_node": "C", "end_node": "A"}, "
    address": "192.168.1.100"}' | nc -u 172.20.10.4 5004
```

A.7 Hardware

Additional hardware-related issues identified:

- **Adapter Compatibility**: The adapter for the Swiss plug is incompatible with current T13 power sockets and only functions with legacy installations such as T12 (SEV 1011:2009).
- **Power adapter** The power adapter exhibited malfunctions from the very beginning.
- **Micro SD Card Quality**: The provided micro SD card is insufficient in both capacity and reliability. It is recommended to use a high-quality micro SD card and re-flash the operating system image for optimal performance.
- **Motor Orientation Issue**: Due to the presence of two motors, there is a 50% probability that the direction of travel will be reversed. This issue was discovered only during the `JETANK_6_gamepadCtrl` tutorial, necessitating a complete disassembly of the robot to correct the wiring.
- **USB Flashing Speed**: It is recommended to use a USB 3.0 dongle for flashing, as the provided dongle is likely USB 2.0, significantly slowing down the flashing process.
- **Chassis Design Flaw**: Spare outlets in the chassis are poorly positioned, making it impossible to remove the micro SD card without fully disassembling the robot. This issue applies to other chassis outlets as well.
- **Cabling Improvements**: Improved cable management is necessary to facilitate smoother installation and maintenance.
- **Mounting Gear Quality**: The provided mounting gear, including screwdrivers and screws, is of low quality and lacks magnetic properties, making assembly challenging, especially in confined or hard-to-reach areas.

B Further links

- Code of master thesis project
- How digitec and galaxus use AGVs
- Digitec packaging machine
- Another project: Cleaning with Jetank
- Road-Following and Collision-Avoidance combined
- Improved QR Code detection
- How to upgrade python on Jetpack 4.6

C Contribution Statement

Both Luzi and Christoph have evenly contributed to this project.

D Aid registry

Aid	Usage / Application	Affected Areas
Chat GPT o1 - OpenAI	Providing LaTeX commands for writing shell and python code	Whole document
Chat GPT o1 - OpenAI	Drafting an abstract based on the rest of the document	Abstract
Chat GPT o1 - OpenAI	Drafting citations of various articles	Bibliography
LanguageTool	Grammar and spelling correction	Whole document

Table 6: *Writing Aids (Art. 57 AB)*

E Declaration of Authorship

We hereby declare:

- That we have written this thesis independently.
- That we have written the thesis using only the resources specified in the index.
- That all parts of the thesis produced with the help of external resources have been ap-propriately declared.
- That we have handled both input and output responsibly when using AI. We confirm that we have only used public data or data released with consent, and we have verified, declared, and comprehensibly referenced all results and/or other forms of AI assis-tance as required. We acknowledge that we are responsible for any incorrect content, violations of data protection law, copyright law, or scientific misconduct (e.g., plagia-rism), even if such violations occur unintentionally.
- That we have mentioned all sources used and cited them correctly in accordance with established academic citation rules.
- That we have acquired all necessary rights to any materials used, such as images or graphics, or that these materials were created by us.
- That the topic, the thesis, or any parts of it have not already been submitted for any oth-er coursework or examinations unless explicitly agreed upon with the faculty in ad-vance and noted as such in the thesis.
- That we are aware of the legal provisions regarding the publication and dissemination of parts or the entirety of this thesis and that we comply with them accordingly.
- That we are aware our thesis may be electronically checked for plagiarism and for third-party authorship of human or technical origin, and we grant the University of St. Gallen the necessary copyright permissions according to the Examination Regulations for administrative purposes.
- That we are aware the University will address any violation of this Declaration of Au-thorship with disciplinary and legal consequences, which may include expulsion from the University or the revocation of our title.

By submitting this paper, we confirm through our conclusive action that we are submitting the Declaration of Authorship, that we have read and understood it, and that the information provided is true.