Junior Journal

OPEN ACCESS
ARTICLE

Junior Journal

Integration of a High-Level Control And Low-Level Control on a Low-Cost Mobile Robot Controller

Regular Paper

Himanshu Mongia[1*], Timon Höbert[2*], Raffael Tfirst[3*] and Andrej Gall[4*]

[1,2,3,4]Vienna Institute of Technology - Department of Information Technology, Austria
* Corresponding author E-mail: himanshu.mongia@outlook.com

Accepted 2013

Abstract The importance of mobile robots increases day by day in various fields of application. In order to gain the most benefit, the robots need to adapt to the existing environment while providing fast and high precision functionalities like sorting various components on the basis of a specific criterion.

Semantic enhanced multi-agent systems have proven to serve well for adaptation to an existing environment, especially when dealing with heterogeneous systems. However, these are not very suitable for fast and high precision hardware-specific functionalities when directly implemented on low-end industry controllers. Therefore, this work suggests the usage of an underlying Low-Level Control (LLC) that is responsible for handling the used equipment (e.g., the mobile robot's arm).

This approach takes the load off the multi-agent system and hence improves the overall system operation. By encapsulating the hardware specific functions, a flexible system architecture can be achieved. This enables the usage of an abstract artificial intelligence in form of a common interface for various hardware components.

Keywords High-Level Control, Low-Level Control, Mobile Robots, Industry Controllers, Multi-Agent System, Communication Interface, International Electrotechnical Commission 61499, Reconfigurable Systems, Semantics

1. Introduction

Since the robotics technology is increasingly used for general purpose and not only in industry it is necessary to make the robots more flexible, safe and self learning. Unlike in the industry, these environments are not static. Therefore, the robots have to be more adaptive and independent. [1]

This is the field of usage where mobile robots are the most suitable technology. They can offer flexible behaviour, which is capable of handling dynamic changes. [2] At the beginning, a single knowledge intensive multi agent system was used (High-Level Control (HLC)). This system enabled an ontology communication and a cooperation among multiple autonomous and heterogeneous units. [2] Each agent supervised one robot, that means, the agent had the knowledge about the skills of the robot. These agents are intended to represent the intelligence of the robot by planning, coordination and scheduling. In the old implementation they also had the unsuitable task of controlling the inputs and outputs of the robots' hardware. This led to an overall slow performance and long responses, as the agents are not optimized for hardware tasks. That is why a LLC was implemented on the robot. It was capable of doing the tasks of controlling the robot, which means all actuators and sensors. For this purpose the International Electrotechnical Commission (IEC) 61499 was chosen, because of the promoted re-usability, flexibility and rapid reconfiguration.

The target was to implement a communication interface between these two systems. The existing LLC functionality shall be extended through an additional HLC by the implementation of the interface, whereby the high performance of the LLC must not be compromised.

This paper is organized as follows: the architecture of HLC, LLC and the communication interface is introduced in Section 2; the implementation of the interface is presented in Section 3; a comparison between High- And Low-Level Control (HALLC) and HLC and conclusion are presented in Section 4 and Section 5 respectively.

2. Architecture

Both frameworks, HLC and LLC, are explained in the chapter below. In addition to this, the requirements for a communication interface between both frameworks are examined. Subsequently, different methods of communication are contemplated, so the most suitable one can be chosen.

2.1. High-Level Control (HLC)

The HLC is a planning system for complex problems, which are to be solved by using some form of artificial intelligence. The combination of different technologies like multi-agent systems, rule-based systems and ontologies seems to be a promising approach for the implementation of intelligent systems.

2.1.1. Multi-Agent Systems

The usage of agent-based systems (also known as multi-agent systems) can have various reasons. Some domains basically require it when different groups of interest act together.

As shown in **figure 1**, an agent then represents the goals and interests of a domain and has specific domain knowledge belonging to his task. The agent acts fully autonomously, can react to changes and influences his environment. That means, each agent has to handle with a dynamic environment. In order to communicate between domains, the underlying standard specifies interfaces which are used by agents for communication. [3]

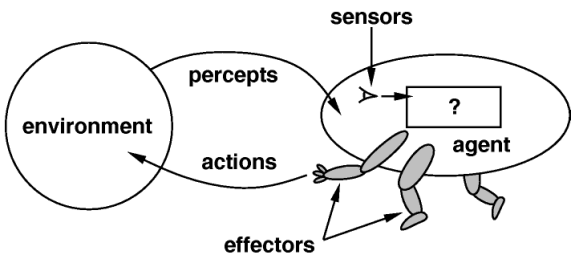

Figure 1. Tasks of an agent [4]

2.1.2. Rule-Based Systems

Rule-Based systems are a specific type of expert-systems which are a specific type of knowledge-based systems. The general purpose of expert-systems is it to map human knowledge into a knowledge base which is used to support the decision-making process. In rule-based systems, the knowledge base consists of persistent rules ("rule base") which are in the form shown in the **listing 1**. [5]

Listing 1. Syntax of a rule

```
IF <conditions> THEN <actions>
```

A rule consists of an IF and a THEN-part. The IF-part of a rule specifies a particular situation, which can be covered by the rule. The THEN-part of a rule specifies which operations have to be executed in case the IF-part equals true.

In addition to the knowledge base a rule-based system mostly consists of the following parts: [5]

- The working memory is transient and contains the underlying data ("fact base"), which is used by the rules.
- The inference engine applies rules to the fact base and uses special algorithms called "Forward Chaining" and "Backward Chaining".

2.1.3. Ontologies

An ontology is a hierarchical and semantical representation or mapping of a system (see **figure 2**). It contains – such as a Java class hierarchy – classes, their properties and relationships between classes. Furthermore, ontologies may contain instances of classes, which can be created and initialized. An ontology normally underlies specific rules which make it possible to find conclusions. A subset of ontologies is taxonomies, but they are much simpler because of representing only the hierarchy without relationships. [6]

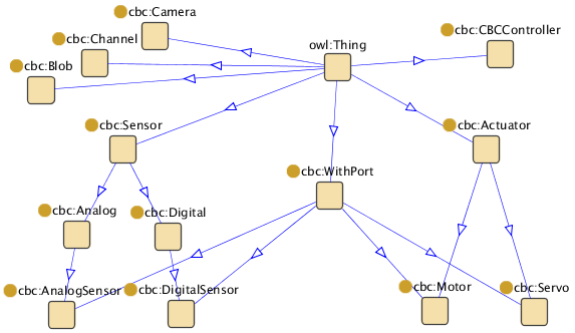

Figure 2. Structure of an ontology [7]

2.1.4. Architecture

The basic structure of the used Framework Disbotics Core is shown in **Figure 3**.

As already mentioned, the system is a combination of an agent-based system, a rule-based system and an ontology. There are multiple robot controllers shown in

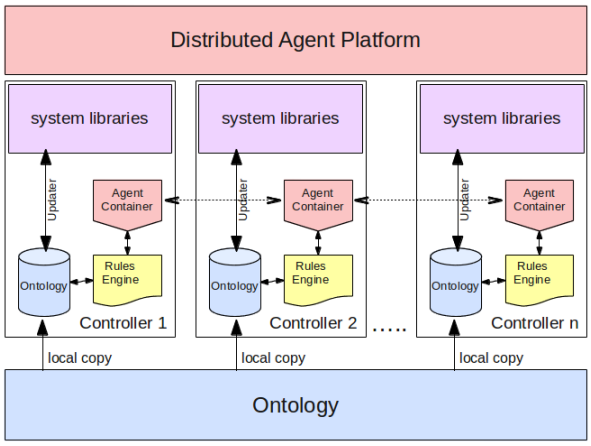

Figure 3. Basic structure of the HLC [7]

the figure. All of them possess a local copy of the shared ontology. On account of this each controller has access to it's own components (e.g., sensors, motors, servos) as well as the components of another controller. The big benefit of having a global ontology with components from all controllers is, that it is possible to write rules for components from other controllers. However, these components do not get updated automatically, so it is necessary to communicate with agents from other controllers and request their ontology values.

The updating process is performed between the ontology and the specific system-libraries of the controller; in case of the CBC Botball Controller (CBC), the CBC Botball Controller Java Virtual Machine (CBCJVM) is used for that. In this process, the own components mapped in the ontology are synchronized with the values from the system-libraries in order that rules can fire for them. Each controller has its own rules file, which contains the main behaviours of the robot defined in a rule based programming language, such as OPSJ. [8]

In this case, the OPSJ Rule Engine is also used for execution of the operations defined by the THEN-part of a rule. Consequently, there is no separation between the abstract artificial intelligence and concrete hardware-specific operations. So the same system is occupied with both tasks, thinking and doing, at the same time. Due to the required high update rate of the rules engine, the actual execution of operations is very much compromised in its' speed and efficiency. This inevitably leads to a worse accuracy and higher error rate of the robot.

2.2. Low-Level Control

The LLC describes the details and individual components of a system. This is required in order to implement it's core functionalities. Amongst assembler and machine code there are plenty of ways to use LLC, like using high-level programming languages (e.g., C or C++). This way the speed of execution stays the same while the language is more platform independent and abstract than assembler. [9]

The standard of programmable logic controllers has been published and prepared by the IEC. It is inevitable in the sector of automation engineering. The newest standard is called IEC 61499 and it is the object-oriented expansion of the IEC 61131. The main advantage of 61499 is the ability to control distributed systems. [10]

The IEC 61499 contains the standardization of different programming languages, instruction sets and concepts. [11] This way possible incompatibilities of Programmable Logic Controller (PLC)-Systems can be avoided. That guarantees the portability of central platforms. With the IEC 61499 modules, or to be precise function blocks **(figure 4)** can be created that provide the ability to control distributed systems. The programming language used to implement these function blocks is C++. [12]

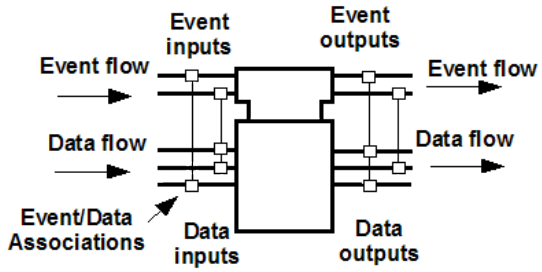

Figure 4. Overview of a function block [13]

The most important function blocks available are the *basic function blocks*. They contain the algorithms that are necessary to control the sequence of a program. *Service Interface Function Blocks* represent the interface to low-level services provided by the operating system or hardware of the embedded device. These can be elements for the graphical user interface (GUI) like a slider or a knob, communication services like client-server communication, or interfaces to hardware such as sensors or motors. Single function blocks can also be combined to a *composite function block*. [13]

Basic function blocks **(figure 5)** contain algorithms written in C++. They can be executed by sending an event to the corresponding block, which is able to communicate with its algorithm in the so-called Execution Control Chart (ECC). As soon as the algorithm ends, the event continues to the next function block. The lower interfaces of a function block are attribute in- and outputs, that can be used within the algorithm.

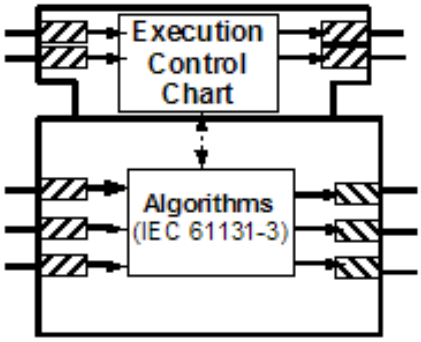

Figure 5. Structure of a basic function block [13]

In order to make use of those function blocks they have to be integrated into a system first. The system contains applications that contain function blocks in order to perform the desired actions.

The most important facts of the IEC 61499 are the portability, the interoperability and the configurability **(figure 6)**. The first one implies that software tools are able to accept and correctly interpret software components and system configurations produced by other software tools. The interoperability enables embedded devices to operate together to perform the functions needed for distributed applications. At last software tools developed by multiple vendors can configure any device and its software components with the IEC 61499. [13]

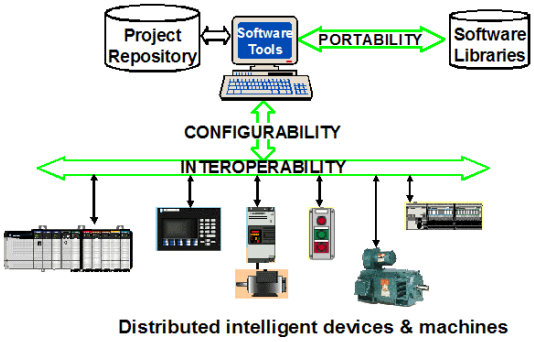

Figure 6. Functionality of IEC 61499 [13]

2.3. Interface High Level Control and Low Level Control

According to the principle of "Separation of concern" [14], the following requirements are imposed on a communication interface between the HLC and the LLC: [15]

- The interconnection of individual layers can only be realised as components (i.e., between LLC of a device and its agent assigned via physical decomposition).
- The primary target of the interface therefore is the vertical integration within the components.
- The individual layers should be as loosely coupled as possible. The functionality of the LLC is extended and can be influenced or more specifically reconfigured by the HLC. However, given a certain (static) starting configuration, the system shall be operational even without the third communication layer.
- The interface must be generic (i.e., it shall be independent from specific protocols, procedures, media or software platforms).
- Consequently, the LLC is neither aware of the presence of a HLC layer nor of its capabilities. The LLC simply provides an interface.
- Due to the real time nature of the LLC, it must not be burdened by the communication interface disproportionately (i.e., unnecessary overhead must be avoided).

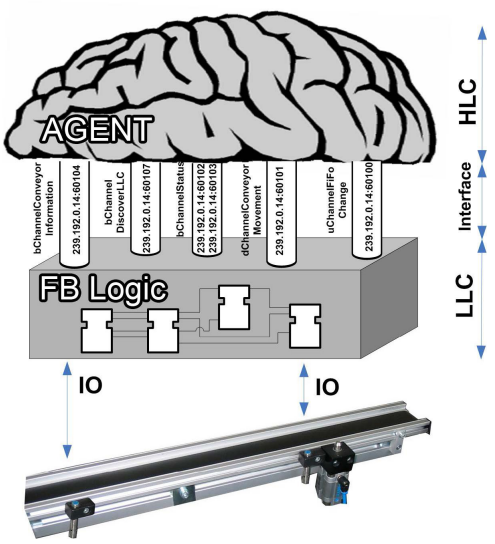

Figure 7. Architecture of the whole framework [14]

2.3.1. Methods and Problems

Currently, the following methods are used to implement such a communication interface between the HLC and the LLC:

- **Network-based**: Since both layers are capable of communicating over the network, using widely spread network protocols like User Datagram Protocol (UDP) over Ethernet or Transmission Control Protocol (TCP)/Internet Protocol (IP) is an obvious choice. For more sophisticated applications industrial Ethernet protocols might be used. [15]
- **Shared Memory**: Considering an ideal system (e.g., enough computational resources for the agent platform available) both HLC and LLC reside on the same controller. Local inter-process communication such as shared memory can be applied. [16]

The option of network-based communication is preferred in practice, as standalone systems, meaning HLC and LLC on the same controller, can not be easily implemented because of the resource deficiency of current controllers. When working with network-based communication, the HLC is usually driven on a single, central high-performance server. Various controllers, which run the LLC locally, are connected via a network protocol with the central server. This structure raises the following problems:

- **Communication Cost**: The avoidable communication between the LLC of a single controller and the assigned agent covers multiple physical devices and therefore requires more resources and time. Especially the higher time exposure affects the real time feature of the LLC, often resulting in a worse precision.
- **Central Administration**: As the entire HLC and therefore all agents responsible for various connected controllers reside on a single central server, the entire system is very prone to failure. Misbehaviour or breakdown of the physical device directly affect the entire system.

3. Implementation of an Interface Between High- and Low Level Control

Existing interfaces of the HLC and LLC are used and enhanced for the communication interface.

The LLC provides various function blocks for communication, which send and receive data via the network. These function blocks can also communicate with other processes on the same device via the virtual loop-back interface (*localhost*).

The HLC is enhanced with a communication plug-in, which translates the received LLC specific data for the agent.

3.1. LLC Interface

The standard IEC-61499 provides communication interfaces via TCP and UDP. The TCP communication is always handled synchronously while the UDP communications happens asynchronously.

Synchronous communication means, that both communication partner synchronize their status, therefore wait for the other, when sending or receiving. A sent command therefore always requires a response, which is awaited. See **Figure 8**.

The program structure of the function blocks in IEC-61499 allows an easier management of asynchronous communication via UDP, as a synchronous procedure with waiting operations would require a more complex structure. See **Figure 9**.

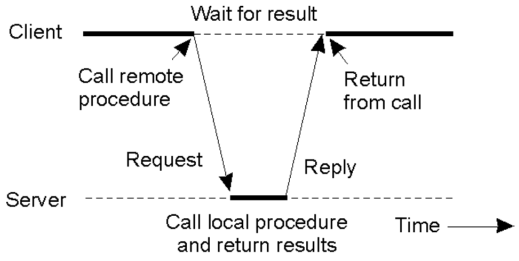

Figure 8. Synchronous Communication [17]

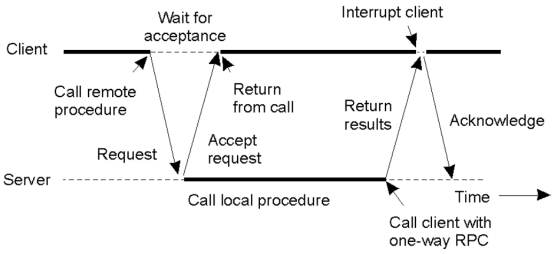

Figure 9. Asynchronous Communication [17]

The UDP communication operates on the Publish-Subscribe model: Various subscribers can subscribe to publishers or their messaging channels to receive notifications about new data. This is realized with publish-function blocks to send out new data, and subscribe-function blocks to receive. Therefore every function block needs a particular endpoint (communication channel), which consists of an address and a port.

3.2. HLC Interface

The HLC-Plugin executes the communication with the LLC with byte streams via the network interface. Communication commands can be executed in the THEN-part of a rule, which send the required data to subscribe blocks of the LLC. One thread per endpoint is responsible for receiving the response data of publish blocks and persisting these in the ontology for further usage (in the respective rules). A plug-in is the optimal choice to integrate the implementation into the existing HLC framework "Disbotics Core" [7], which is easily extensible due to its plug-in structure. See **Figure 10**.

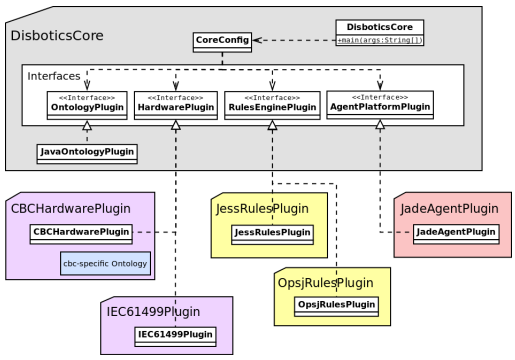

Figure 10. Architecture of HALLC [7])

4. Tests

Two different types of tests were constructed to compare the usage of HLC and HALLC in their characteristics like efficiency, speed and precision. In order to compare both technologies analogously, the algorithm of each test case was plotted out carefully, before implementing and adapting it for both technologies. Therefore, a fair comparison between both technologies can be assured.

4.1. Line-following

The line-following test case consists of a robot following a lap marked with a (black) line by using light sensors. As the robot is obliged to check his position, meaning update sensor values in a high frequency, this test case is supposed to highlight the performance difference of both technologies. Therefore, an algorithm was carefully planned and implemented in the same way in both technologies, these being HLC and HALLC. This assures a fair comparison of both technologies. Key figures like average lap time and deviation from the target line are measured and evaluated.

4.1.1. Evaluation

The following **figure 11** shows the average lap time of both technologies. The lap time differences between both

technologies as well as the consistency of average lap times of both technologies are to be noted.

On average, the HLC needs 18% more time than the HALLC per lap. The better performance of the HALLC is attributable to a shorter reaction delay of the framework, which causes a more precise and therefore faster tracking of the target line. The reaction delay is also tested in the following test case "Sorting".

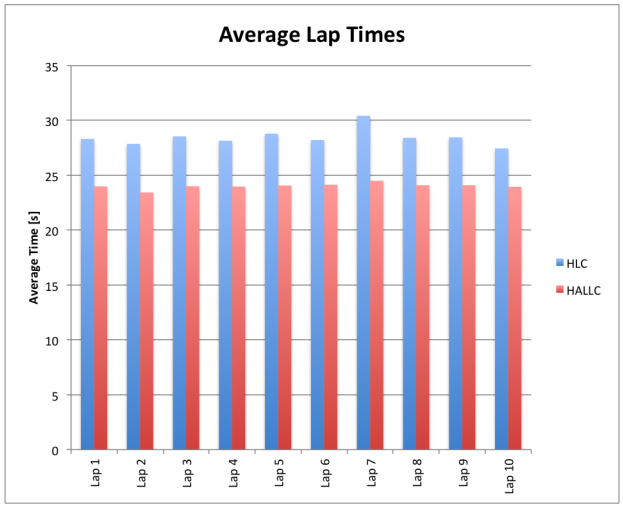

Figure 11. Average lap Times

The following **figure 12** shows the average deviation of the robot to the target line. The HALLC performs distinctly better than the HLC in this aspect. As already explained above, this improvement is accountable to the higher reaction capability of the used LLC, which is incorporated in the HALLC.

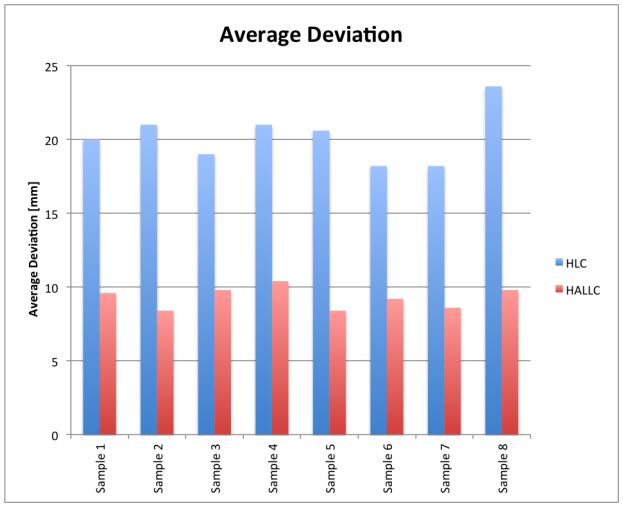

Figure 12. Average Deviation

4.2. Sorting

It is important to know how the HALLC implementation compares to the HLC in different situations, that is where the sorting test case comes into play. The objective is to have the robot gather LEGO DUPLO-structures and sort these by size, differentiating between a simple LEGO DUPLO-block and a structure consisting of two LEGO DUPLO-blocks. Thereby it is important that the robot can correctly recognize the given variant and act accordingly by sorting the LEGO DUPLO-structure into the suitable container. In order to allow statistical comparison the following data has to be measured:

- the performance or to be precise the time of execution
- the reaction time: the difference of time between an arisen event and the robot's realization

4.2.1. Evaluation

The average sorting time of all blocks varies in each test case as seen in **figure 13**. Nevertheless all total times of the HALLC are approximately equivalent to the ones of the HLC. The direct comparison however shows that the LLC has performed better in the various runs.

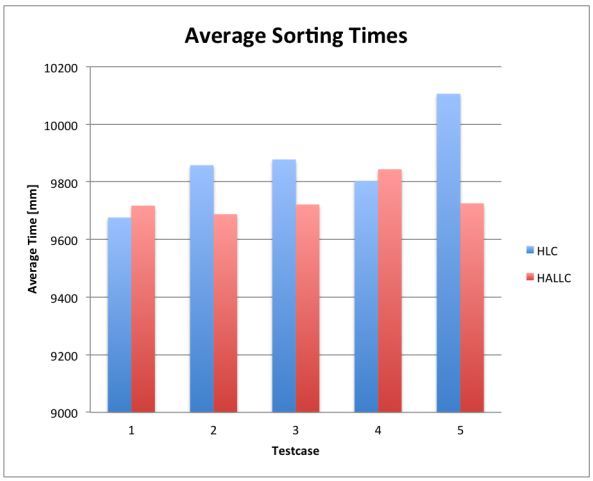

Figure 13. Average sorting Times

The average reaction delay shows clearer results in **figure 14**. It is way shorter with the HALLC implementation. This data suggests that the LLC offers better precision and high reaction speed within this test case.

5. Conclusion

At the moment the control of mobile robots is being put into practice by the LLC, a fast, hardware-specific control or by the HLC, an abstract control responsible for artificial intelligence. A combination of both variants has been applied in the past. However, usually a single HLC is executed on a central sever due to the required high resources. The HLC contains agents for several LLCs which actually run on mobile controllers.
By creating a combination of the HLC and LLC on the same controller as suggested, autonomous stand-alone systems can be created, which bring great advantages. The objective is to create a system with high performance, called OHALLC High- And Low-Level Control (OHALLC), that has the ability to control, plan and perform commands separately. This way the whole

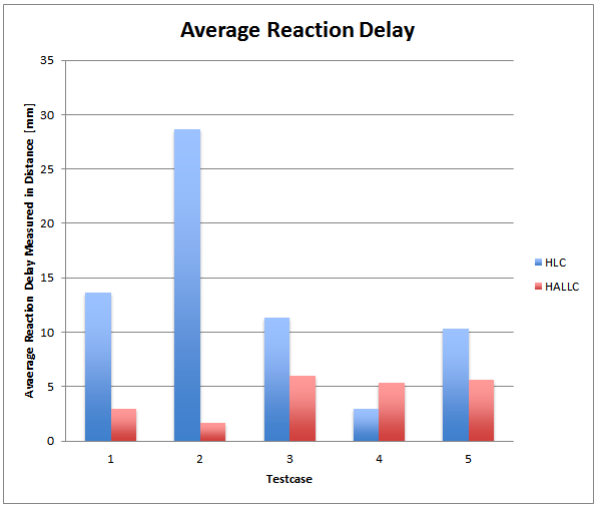

Figure 14. Average Delay Distance

product benefits from the aspects of both technologies, the fast reaction and the complex planning.

In order to compare the benefits of HLC and HALLC, two test cases were chosen to test both technologies within various scenarios. According to the results the performance of the HALLC significantly exceeds the one of the HLC, when various low level operations have to be executed in the background like in the line-following test case. As the HALLC separates the planning process from execution, the agents can continue to plan without distraction, even when frequent updates are being executed in background. That is why the measured time of execution and the deviation are way smaller.
The sorting test case shows that the performance improvement is dependent to the given situations. As the test case does not require a high update frequency, the performance difference between both technologies is not substantial. However the measured delay is smaller with the HALLC, since the updating process is executed by the LLC, which is not busy with any other planning tasks like the HLC.

6. References

[1] Ozan Caldiran, Kadir Haspalamutgil, Abdullah Ok, Can Palaz, Esra Erdem, and Volkan Patoglu. Bridging the gap between high-level reasoning and low-level control. In Esra Erdem, Fangzhen Lin, and Torsten Schaub, editors, *Logic Programming and Nonmonotonic Reasoning*, volume 5753 of *Lecture Notes in Computer Science*, pages 342–354. Springer Berlin / Heidelberg, 2009.

[2] G. Koppensteiner, R. Hametner, R. Paris, A.M. Passani, and M. Merdan. Knowledge driven mobile robots applied in the disassembly domain. In *Automation, Robotics and Applications (ICARA), 2011 5th International Conference on*, pages 52 –56, dec. 2011.

[3] P. Stone and M. Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, 2000.

[4] Stuart Jonathan Russell, Peter Norvig, John F Canny, Jitendra M Malik, and Douglas D Edwards. *Artificial intelligence: a modern approach*, volume 2. Prentice hall Englewood Cliffs, NJ, 1995.

[5] Jocelyn Ireson-Paine. Lecture 3: Expert systems. Online, 1996.

[6] w3.org. Owl web ontology language. Online.

[7] KOZA Clemens; KROFITSCH Christoph; PARG Manuel; PRVULOVIC Antonio. *DisboticsCore UserGuide*. Practical Robotics Institute Austria, 1.4.0 edition, January 2012.

[8] CL Forgy. Opsj 4.1 manual. *Production Systems Technologies Inc*, 2001.

[9] Jeremy Condit; Matthew Harren; Zachary Anderson; David Gay; George C. Necula. *Programming Languages and Systems*. Springer Berlin Heidelberg, 2007.

[10] IEC. Iec homepage. available at: http://www.iec.ch/about/?ref=menu, 2013.

[11] KW-Software GmbH. Technology for automation leaders. available at: https://www.kw-software.com/de/iec-61131-control, 2013.

[12] Robert W. Lewis. *Programming industrial control systems using IEC 1131-3*. Institution of Electrical Engineers, 1998.

[13] James H. Christensen. Holobloc website. available at: http://www.holobloc.com/papers/iec61499/overview.htm, 2012.

[14] M. Merdan, W. Lepuschitz, I. Hegny, and G. Koppensteiner. Application of a communication interface between agents and the low level control. In *Autonomous Robots and Agents, 2009. ICARA 2009. 4th International Conference on*, pages 628–633, Feb.

[15] I. Hegny, O. Hummer, A. Zoitl, G. Koppensteiner, and M. Merdan. Integrating software agents and iec 61499 realtime control for reconfigurable distributed manufacturing systems. In *Industrial Embedded Systems, 2008. SIES 2008. International Symposium on*, pages 249–252, June.

[16] O. J. Lopez and J. M. Lastra. A real-time interface for agent-based control. *IEEE 2nd International Symposium on Industrial Embedded Systems (SIES'07)*, 2007.

[17] Maarten van Steen Andrew S. Tanenbaum. *Distributed Systems: Principles and Paradigms*. Prentice-Hall, 2002.