6

A Multi-Agent Reinforcement Learning Approach for the Efficient Control of Mobile Robots

U. Dziomin[1], A. Kabysh[1], R. Stetter[2] and V. Golovko[1]

[1]Brest State Technical University, Belarus
[2]University of Ravensburg-Weingarten, Germany
Corresponding author: R. Stetter <stetter@hs-weingarten.de>

Abstract

This paper presents a multi-agent control architecture for the efficient control of a multi-wheeled mobile platform. Such control architecture is based on the decomposition of a platform into a holonic, homogenous, multi-agent system. The multi-agent system incorporates multiple Q-learning agents, which permits them to effectively control every wheel relative to other wheels. The learning process was divided into two steps: *module positioning* – where the agents learn to minimize the error of orientation, and c*ooperative movement* – where the agents learn to adjust the desired velocity in order to conform to the desired position in formation. From this decomposition, every module agent will have two control policies for forward and angular velocity, respectively. Experiments were carried out with a simulation model and the real robot. Our results indicate a successful application of the purposed control architecture both in the simulation and in real robot.

Keywords: control architecture, holonic homogenous multi-agent system, reinforcement learning, Q-Learning, efficient robot control.

6.1 Introduction

An efficient robot control is an important task for the applications of a mobile robot in production. The important control tasks are power consumption optimization and optimal trajectory planning. Control subsystems should

Advances in Intelligent Robotics and Collaborative Automation, 123–146.

provide energy consumption optimization in a robot control system. Four levels of robot power consumption optimization can be distinguished:

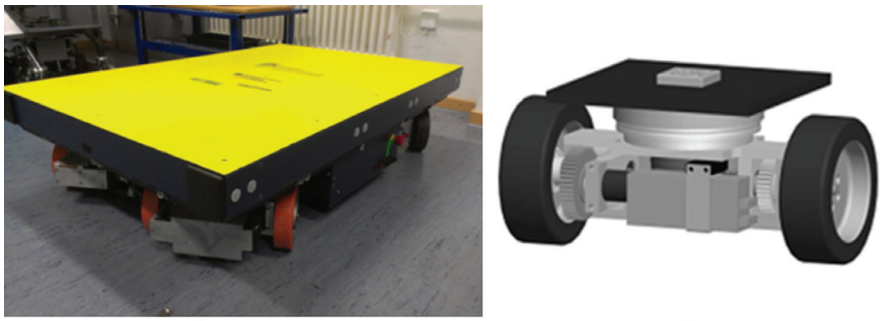
- *Motor power consumption optimization*. Those approaches based on energy-efficient technologies of motor development that produce substantial electricity saving and improve the life of the motor drive components [1, 2];
- *Efficient robot motion.* Commonly, this is a task of an inverse kinematics calculation. But the dynamic model is usually far more complex than the kinematic model [3]. Therefore, intellectual algorithms are relevant for the optimization of a robot motion [4];
- *Efficient path planning*. Such algorithms build a trajectory and divide it into different parts, which are reproduced by circles and straight lines. The robot control subsystem should provide movement along the trajectory parts. For example, Y. Mei and others show how to create an efficient trajectory using knowledge of the energy consumption of robot motions [5]. S. Ogunniyi and M. S. Tsoeu continue this work using reinforcement learning for path search [6];
- *Efficient robot exploration*. When a robot performs path planning between its current position and its next target in an uncertain environment, the goal is to reduce repeated coverage [7].

The transportation of cargo is an actual task in modern production. Multi-wheeled mobile platforms are increasingly being used in autonomous transportation of heavy components. One of these platforms is *a production mobile robot*, which was developed and assembled at the University of Ravensburg-Weingarten, Germany [3]. The robot is illustrated in Figure 6.1(a). The platform dimensions are 1200cm in length and 800cm in width. The maximum manufacturer's payload is 500kg, battery capacity is 52Ah, and all modules drive independently.

The platform is based on four vehicle steering modules [3]. The steering module (Figure 6.1(b)) consists of two wheels powered by separate motors and behaves like a differential drive.

In this paper, we explore the problems of *formation control* and *efficient motion control* of multiple autonomous vehicle modules in circular trajectory motion. The goal is to achieve a circular motion of a mobile platform around a virtual reference beacon with optimal forward and angular speeds.

One solution to this problem, [8–10] is to calculate the kinematics of a one-wheeled robot for circle driving and then generalize it for multi-vehicle systems. This approach has shown promising modeling results.



a) b)

Figure 6.1 Production mobile robot: Production mobile platform (a); Driving module (b).

The disadvantage of this technique is its low flexibility and high computational complexity.

An alternative approach is to use the machine learning theory to obtain an optimal control policy. The problem of multi-agent control in robotics is usually considered as a problem of formation control, trajectory planning, distributed control and others. In this paper we use techniques from multi-agent systems theory and reinforcement learning to create the desired control policy.

The content of this paper is the following: Section 6.2 gives a short introduction to the theory of holonic, homogenous, multi-agent systems and reinforcement learning. Section 6.3 describes the steering of a mobile platform in detail. Section 6.4 describes the multi-agent decomposition of a mobile platform. Using this decomposition, we propose a multi-agent control architecture based on the model described in Section 6.2. Section 6.5 contains a detailed description of the multi-agent control architecture. The Conclusion highlights important aspects of the presented work.

6.2 Holonic Homogenous Multi-Agent Systems

A *multi-agent system* (MAS) consists of a collection of individual agents, where each agent displays a certain amount of autonomy about its actions and perception of domain, and communicates via message-passing with another agent [11, 12]. Agents act in organized structures which encapsulate the complexity of subsystems and therefore modularize its functionality. Organizations are social structures with means of conflict resolution through coordination mechanisms [13]. The overall *emergent behavior* of a

multi-agent system is composed of a combination of individual agent behaviors determined by autonomous computation within each agent, and by communication among agents [14]. The field of MAS is a part of distributed AI, where each agent has a distinct problem solver for a specific task [12, 14].

6.2.1 Holonic, Multi-Agent Systems

An agent (or MAS) that appears as a single entity to the outside world but is in fact composed of many sub-agents with the same inherent structure is called *holon*, and such sub-agents are called *holonic agents* [11, 14]. The transformation of a single entity into a set of interacting subagents is called *holonic decomposition*. Holonic decomposition is an isomorphic transformation. Gerber et al. [15] show that an environment containing multiple holonic agents can be isomorphically mapped as an environment in which exactly one agent is represented explicitly, and vice versa.

For the purposes of this paper and without the loss of generality, we use terms *holon* and *holonic multi-agent system* (Holonic MAS) interchangeably, meaning that a MAS contains *exactly one* holon. In the general case, a holonic, multi-agent system (called *holarhy*) is a self-organized, hierarchical structure composed of holons [14].

A holon is always represented as a single entity to the outside world. From the perspective of the environment, a holon behaves as an autonomous agent. Only a closer inspection reveals that a holon is constructed from a set of cooperating agents. It is possible to communicate with a holon simply by sending messages to them from the environment. The most challenging problem in this design is the distribution of individual and overall computation of the holonic MAS [15].

Although it is possible to organize holonic structures in a completely decentralized manner, it is more efficient to use an individual agent to represent a holon. Representatives are called the *head of the holon*; the other agents in the holon are called the *body* [11]. In some cases, one of the already existing agents is selected as the representative of the holon. In other cases, a new agent is explicitly introduced to represent the holon during its lifetime.

The head agent represents the shared intentions of the holon and negotiates these intentions with the agents in the holon's environment, as well as with the internal agents of the holon. Only the head agent communicates with the entities outside of the holon.

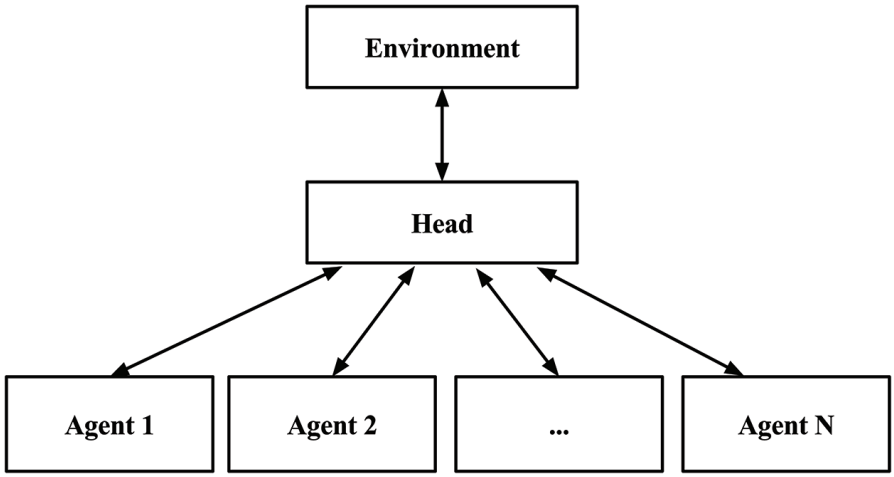


Figure 6.2 Organizational structure of a Holonic Multi-Agent System. Lines indicate the communication channels.

The organizational structure of a holonic, multi-agent system is depicted in Figure 6.2.

When agents join the holon, they surrender some of their autonomy to the head agent. The binding force that keeps the head and body in a holon together can be called a *commitments* [16]. It should be explicitly noted that agents are not directly controlled by the head agent. The agents remain autonomous entities within the holon, but they align their individual behavior with the goals of holon.

6.2.2 Homogenous, Multi-Agent Systems

For the purposes of this paper, we will consider the case when *all body* agents are *homogenous*. In a general, multi-agent scenario with homogeneous agents, there are several different agents with an identical structure (sensors, effectors, domain knowledge, and decision functions) [17]. The only differences among agents are their sensory inputs and the actual actions they take, as they are situated differently in the world [18]. Having different effector outputs is a necessary condition for MAS; if the agents all act together as a unit, then they are essentially a single agent. In order to realize this difference in output, homogeneous agents must have different sensor input as well. Otherwise, they will act identically.

Thus, the formal definition of *holonic, homogenous, multi-agent system* (H^2MAS) is a tuple $\mathcal{H} = \langle \mathcal{B}, h, \mathcal{C} \rangle$**:**

- $\mathcal{B} = \{M_1, M_2, \ldots, M_n\}$– is the set of homogenous *body* agents. Each agent is described by a tuple $M_i = \langle \mathrm{s}, \alpha, \pi_i \rangle$, where:
- s – is the set of possible agent states, where $s_i \in \mathrm{s}$ is the *i*-th agent current state;
- α– is the set of possible agent actions, where $a_i \in \alpha$ current action of the *i*-th agent;
- $\pi : \mathrm{s} \rightarrow \alpha$ is the behavior policy (decision function) which maps it's state to actions;
- h – is the *head agent* representing the holon to the environment and responsible for coordinating the actions inside the holon:
- $\mathcal{S} = \mathrm{s}^{\times n} = \{(s_1, s_2, \ldots, s_n) \,|s_i \in \mathrm{s} \text{ for all } \; 1 \leq i \leq n\}$ – is a *joint state* of the holon;
- $\mathcal{A} = \alpha^{\times n} = \{(\, a_1, a_2, \ldots, a_n) \,|a_i \in \alpha \text{ for all } \; 1 \leq i \leq n\}$ is a *joint action* of the holon;
- $\pi : \mathcal{S} \rightarrow \mathcal{A}$ – global;
- $\mathcal{C}$ – is the *commitment* that defines the agreement to be inside the holon.

The learning of multi-agent systems composed of homogenous agents has a few important properties which affect the usage of such systems.

6.2.3 Approach to Commitment and Coordination in H^2 MAS

The holon is realised exclusively through cooperation among the constituent agents. The head agent is required to co-ordinate the work of the body agents to achieve the desired global behavior of H^2MAS by combining individual behaviors, resolving collisions, etc. In this way, a head agent serves as a *coordination strategy* among agents. The head is aware of the goals of the holon, and has access to important environmental information which allows it to act as a central point of coordination for body agents.

Since a body agent has some degree of autonomy, it may perform an unexpected action, which can lead to uncoordinated behavior within the Holon. The head agent can observe the states and actions of all subordinate agents and can fix undesired behavior using simple coordination rule: if the current behavior of the holon M_i is inconsistent with the head agent's vision, then it sends a correction message to M_i. This action by the head is known as an influence on the body. When the *body* M_i succumbs to the influence, this is called making a *commitment* to the Holon.

6.2.4 Learning to Coordinate Through Interaction

The basic idea of the selected approach for coordination is to use influences between the head and the body to determine the sequence of correct actions to coordinate behavior within the holon. The core design question is how to determine such influences in terms of received messages and how received messages affect changes of individual policies.

To answer this question we postulate that interacting agents should constantly learn optimal coordination from scratch. To achieve this, we can use *influence-based, multi-agent reinforcement learning* [18–20]. In this approach, agents learn to coordinate using reinforcement learning by exchanging rewards with each other.

In reinforcement learning, the i^{th} agent executes an action a_i at the current state s_i. It then goes to the next state s_i' and receives a numerical reward r as feedback for the recent action [21], where $s_i,\ s_i' \in \mathrm{s},\ a_i \in \alpha,\ r \in \mathrm{R}$. Ideally, agents should explore state space (interact with environment) to build an optimal policy π^*.

Let $Q(s,a)$ – represent a *Q-function* that reflects the quality of the specified action a in state s. Optimal policy can be expressed in terms of *optimal Q-function* Q^*:

$$\pi^*(s) = \arg\max_{a \in \alpha(s)} Q^*(s,a). \tag{6.1}$$

The initial values of *Q*-funcions are unknown and equal to zero. The learning goal is to approximate the *Q-function*, (e.g. to find *true Q*-values for each action in every state using received sequences of rewards).

A model of influence-based multi-agent reinforcement learning depicted in Figure 6.3.

In this model, a set of body agents with identical policies π acts in a common, shared environment. The i^{th} body agent M_i in the state s_i selects an action a_i using current policy π, and then moves to the next state s_i'. The head agent observes changes resulting from the executed action and then calculates and assigns a r_i' to the agent as an evaluative feedback.

Equation (6.2) is a variation of the *Q*-learning update rule [21] used to update the values of the *Q*-function, and where learning homogeneity and parallelism are applied. Learning homogeneity refers to all agents building the same *Q*-function, and parallelism requires that they can do it in parallel. The following learning rule executes *N* times per step for each agent in parallel over single-shared *Q*-function:

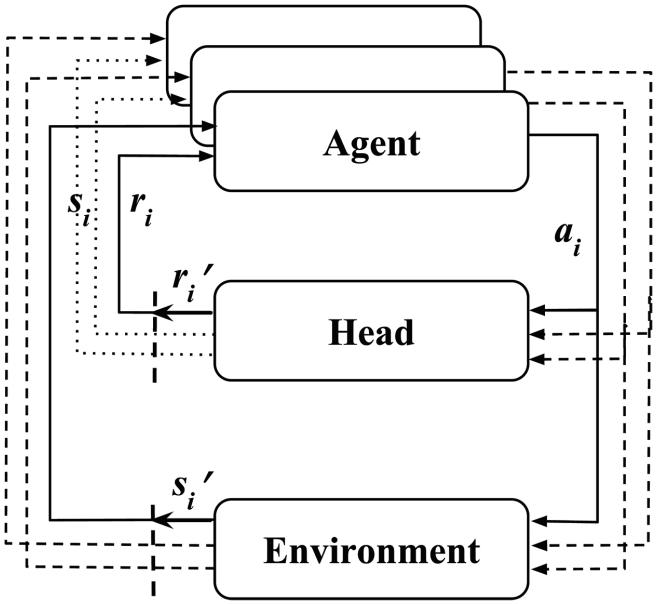


Figure 6.3 Model of Influence Based Multi-Agent Reinforcement Learning in the Case of a Holonic Homogenous Multi-Agent System.

$$\Delta Q\left(s_i, a_i\right) = \alpha \left[r_i' + \gamma \max_{a \in \alpha(s_i)} Q\left(s_i, a\right) - Q\left(s_i, a_i\right) \right]. \tag{6.2}$$

6.3 Vehicle Steering Module

The platform is based on four vehicle steering modules. The steering module consists of two wheels powered by separate motors and behaves as a differential drive. It is mounted to the platform by a bearing that allows unlimited rotation of the module with respect to the platform (Figure 6.4). The platform may be equipped with three or more modules.

The conventional approach for the platform control is a kinematics calculation and an inverse kinematics modeling [3]. The inverse kinematics calculation is known for the common schemes: the differential scheme, car scheme, and bicycle scheme. In the case of production module platforms, the four modules are controlled independently. As a consequence, the control system can only perform symmetric turning. Hence, the platform has limited maneuverability [3]. The other problem is the limitations of the robot

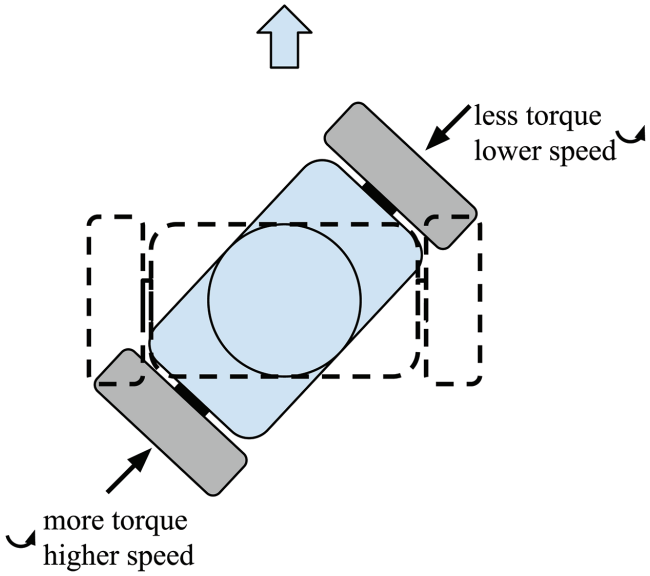


Figure 6.4 The Maneuverability of one module.

configuration. Previous systems require recalculations if modules are added or removed from the platform. These recalculations require a qualified engineer.

The problem of steering the robot along the trajectory is illustrated in Figure 6.5. This trajectory consists of four segments:

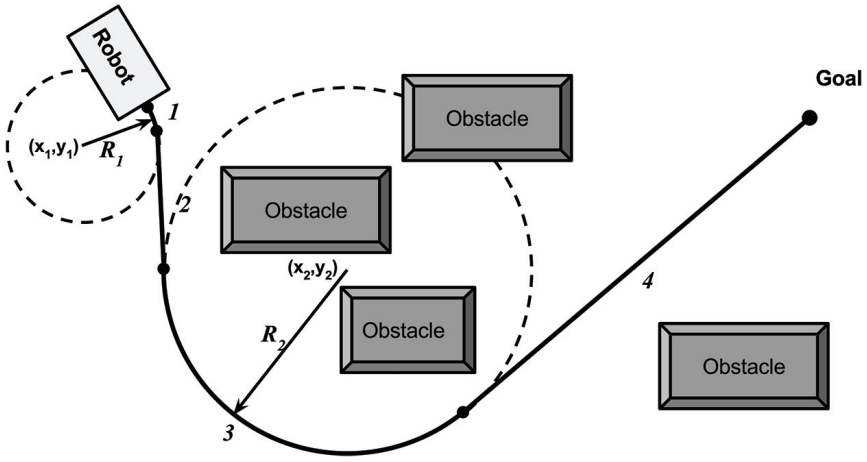


Figure 6.5 Mobile Robot Trajectory Decomposition.

- The turning radius length is R_1, the center of rotation is (x_1,y_1);
- The straight segment;
- The turning radius length is R_2, the center of rotation is (x_2,y_2);
- The straight segment.

The steering of the robot also fulfills the following specifications:

- At the starting point, the robot rotates all modules in the direction of the trajectory;
- A robot cannot stop at any point of the trajectory. The trajectory always has smooth transitions from one segment to another.

6.4 A Decomposition of Mobile Platform

A platform is composed of identical modules attached to the platform in the same way as a *multi-agent decomposition*. This is a prominent way to develop a distributed control strategy for such platforms. Mobile platforms with four identical independent driving modules can be represented as homogenous, holonic, multi-agent systems as described in Section 6.2. The driving modules are represented as body agents (or module agents) and the head agent (or platform agent) represents the whole platform. The process of multi-agent decomposition described above is shown in Figure 6.6.

The whole platform reflects global information, such as the shape of the platform and the required module topology, including its desired positions relative to the centroid of the platform. To highlight this information, we can attach a *virtual coordinate frame* to the centroid of the platform to create the virtual structure.

Figure 6.7 shows an illustrative example of the virtual structure approach with a formation composed of four vehicles capable of planar motions, where

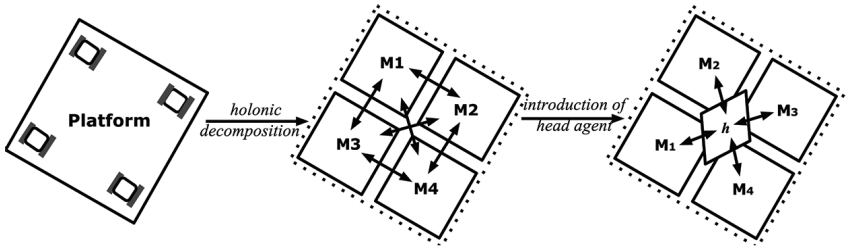


Figure 6.6 Holonic Decomposition of the Mobile Platform. Dashed lines represent the boundary of a Multi-Agent System (the Holon). Introduction of the Head Agent Leads to a reduction of communication costs.

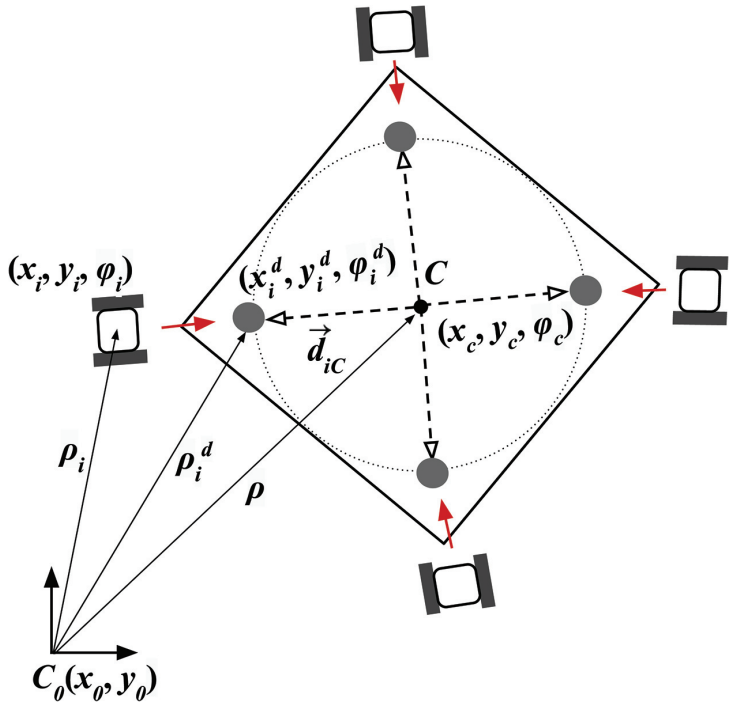


Figure 6.7 Virtual Structure with a Virtual Coordinate Frame composed of Four Modules with a known Virtual Center.

C_0 represents the beacon frame and C represents a virtual coordinate frame located at a virtual center (x_{vc}, y_{vc}) with an orientation φ_{vc} relative to C_0. Values of $\rho_i = [x_i, y_i]^T$ and $\rho_i^d = [x_i^d, y_i^d]$ represent, respectively, the -th vehicle's actual and desired position. Values of φ_i and φ_i^d represent the actual and desired orientation, respectively, of the -th vehicle. Each module's desired position $\left(x_i^d, y_i^d, \varphi_i^d\right)$ can be defined relative to the virtual coordinate frame.

For a formation stabilization with a static formation centroid, if each vehicle in a group can reach a consensus on the center point of the desired formation and specify a corresponding desired deviation from the center point, then the desired motion can be achieved [22]. If each vehicle can track its desired position accurately, then the desired formation shape can be preserved accurately.

The vectors $\vec{d}_i = (x_c - x_i, y_c - y_i)$ and $\vec{d}_{iC} = \left(x_c - x_i^d, x_c - y_i^d\right)$ represent, respectively, the -th vehicle's desired and actual deviation relative to C. The deviation vector $\vec{d}_i^{err}$ of the *i*-th module relative to the desired position is defined as:

$$\vec{d}_i^{err} = \vec{d}_i - \vec{d}_{iC}. \tag{6.3}$$

Each module's desired position can be defined relative to the virtual coordinate frame. Once the desired dynamics of the virtual structure are defined, the desired motion for each agent can be derived. As a result, path planning and trajectory generation techniques can be employed for the centroid while trajectory tracking strategies can be automatically derived for each module [23].

6.5 The Robot Control System Learning

The main goal of the control system is to provide the movement of the robot along the desired circular trajectory. The objective is to create a cooperative control strategy for any configuration of *N* modules so that all the modules within the platform achieve circular motion around the beacon. The circular motions should have a prescribed radius of rotation ρ_C defined by the center of the platform and the distance between neighbors. Further requirements are that module positioning before movement must be taken into account, and the adaptation of angular and linear speed during circular movement to reach optimal values.

We divide the process of learning into two steps:

- *Module positioning* – a learning of the module to rotate to the trajectory direction (6.5.1);
- *Cooperative movement* - a learning of cooperative motion of modules within platform (6.5.2).

The overall control architecture is depicted in Figure 6.8.

From this decomposition, every module agent will have two control policies, π_v and π_ω , for both forward and angular velocity, respectively. Policy π_ω is responsible for correct module orientation around the beacon. Each module follows this policy before the platform starts moving. Policy π_v is used during circular motion of the platform along curves. Both policies are created via reinforcement learning, which allows for generalization.

In the simulation phase, the head agent interacts with the modeling environment. In experiments with real robots, the head agent interacts with the planning subsystem. The Environment/Planning subsystem provides information about the desired speed of the platform v^d and the global state of the multi-agent $S = \left\{U_{i=1}^{N} s_i\right\} \cup \{s_h\}$, where $s_i \in \mathrm{s}$ is the state of the *i*-th module

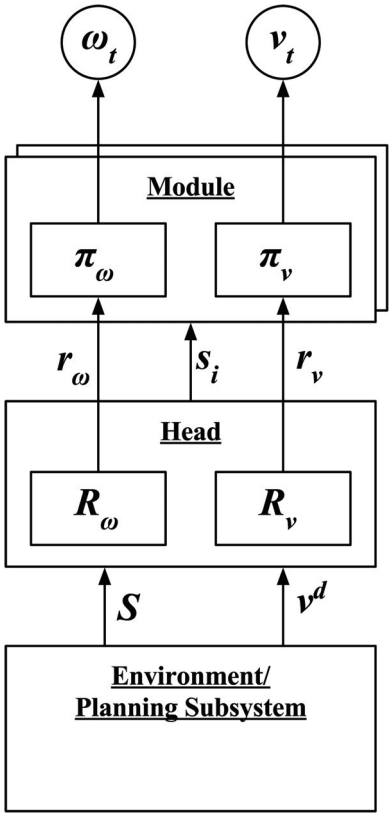


Figure 6.8 A unified view of the control architecture for a Mobile Platform.

is defined by values in Table 6.1 and $s_h = \langle C,\ C_0, \{U_{i=1}^N (x_i^d, y_i^d, \varphi_i^d)\}\rangle$ is the state of the head agent describes the virtual coordinate frame.

6.5.1 Learning of the Turning of a Module-Agent

This subsection describes the model for producing an efficient control rule for the positioning of a module, based on the relative position of the module with respect to the beacon. This control rule can be used for every module, since every steering module agent is homogenous.

The agent stays in a physical, 2-D environment with a reference beacon, as shown in Figure 6.9. The beacon position is defined by coordinates (x_0, y_0). The rotation radius ρ is the distance from the center of the module to the beacon.

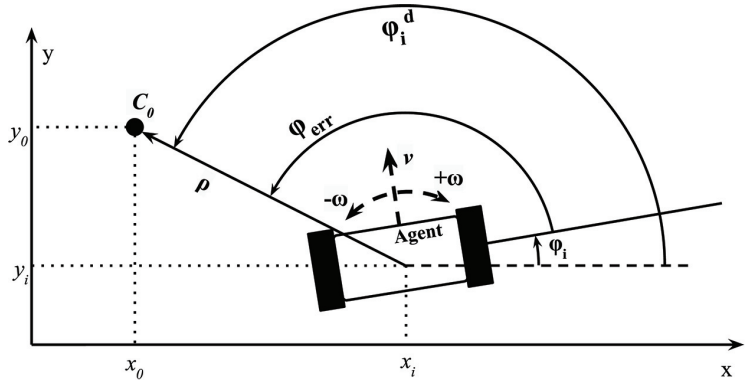


Figure 6.9 State of the Module with Respect to Reference Beacon.

The angle error is calculated using the following equations:

$$\phi_c = \arctan 2(x_0 - x_i, y_0 - y_i), \tag{6.4}$$

$$\varphi_{err} = \varphi_i^d - \varphi_i. \tag{6.5}$$

Here, φ_i^d and φ_i are known from the environment.

In the simulated model environment, all necessary information about an agent and a beacon is provided. In a real robotic environment, this information is taken from wheel odometers and a module angle sensor. The environment information states are illustrated in Table 6.1.

The full set of actions available to the agent is presented in Table 6.2. The agent with actions $A = \{A_\omega, A_v\}$ can change an angle speed by actions $A_\omega = \{\omega_+, \omega_-\}$ and linear speed by actions $A_v = \{v_+, v_-\}$. To turn, an agent controls the angular speed A_ω .

Table 6.1 The Environment Information

No	Robot Get	Value
1	X robot position, x	Coordinate, m
2	Y robot position, y	Coordinate, m
3	X of beacon center, x_b	Coordinate, m
4	Y of beacon center, y_b	Coordinate, m
5	Robot orientation angle, φ_i	Float number, radians $-\pi < \varphi_i \leq \pi$
6	Desired orientation angle relative to robot, φ_i^d	Float number, radians $-\pi < \varphi_i^d \leq \pi$
7	The radius size, r	Float number, m
8	The desired radius size, r^d	Float number, m

Table 6.2 Agent Actions

No	Robot Action	Value
1	Increase force, $v-$	+0.1, m/s
2	Reduce force, $v+$	–0.1, m/s
3	Increase turning left, $\omega+$	+0.1, rad/s
4	Increase turning right, $\omega-$	–0.1, rad/s
5	Do nothing, $\emptyset$	+0 m/s, +0 rad/s

The learning system is given a positive reward when the robot orientation is closer to the goal orientation ($\varphi_{err} \rightarrow 0$) and is using optimal speed ω_{opt}. A penalty is received when the orientation of the robot deviates from the goal orientation or the selected action is not optimal for the given position. The value of the reward is defined as:

$$r'_{\omega} = R_{\omega}(\phi^{t}_{err}, \omega^{t}). \tag{6.6}$$

Where R_{ω} – is a reward function, which is represented by the decision tree depicted in Figure 6.10. Here, φ_{stop} represents the value of the angle, where

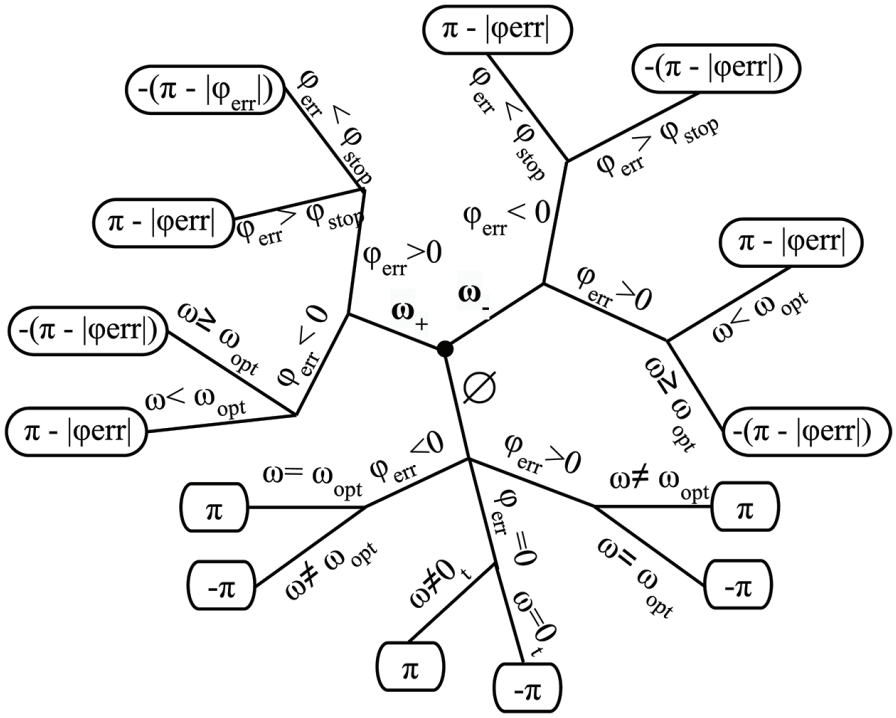


Figure 6.10 A Decision tree of the reward function.

the robot reduces speed to stop at the correct orientation, ω_{opt} $[0.6 .. 0.8]$ rad/s, which is the optimal speed to minimize module power consumption. The parameter φ_{stop} is used to decrease the search space for the agent. When the agent angle error becomes smaller than φ_{stop}, an action that reduces the speed will receive the highest reward. The parameter ω_{opt} shows the possibility of power optimization by setting a value function. If the agent angle error is more than φ_{stop} and $\omega_{opt}^{min} < \omega < \omega_{opt}^{max}$, then the agent reward will increase. This coefficient which determines the increase ranges between [0 .. 1]. The optimization allows the use of the preferred speed with the lowest power consumption.

6.5.1.1 Simulation

The first task of the robot control is becoming familiar with robot positioning through simulation. This step is done once for an individual module before any cooperative simulation sessions. The learned policy is stored and copied for other modules via knowledge transfer. The topology of the *Q*-function trained during 720 epochs is shown in Figure 6.11.

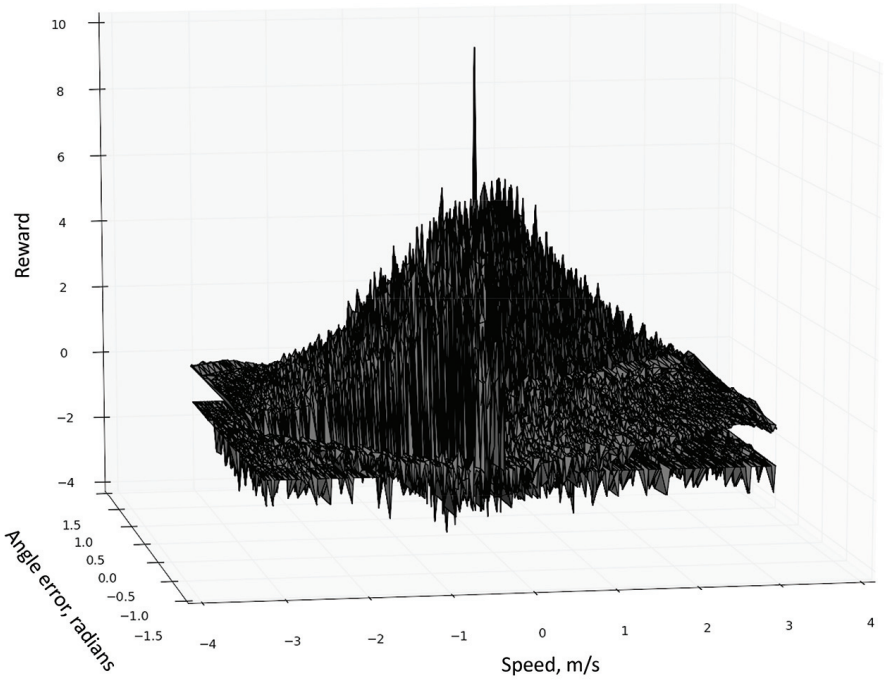


Figure 6.11 Result Topology of the Q-Function.

The external parameters of a simulation are:

- Learning rate $\alpha = 0,4$;
- Discount factor $\gamma = 0,7$;
- Minimal optimal speed $\omega_{opt}^{min} = 0,6$ rad/s;
- Maximum optimal speed $\omega_{opt}^{max} = 0,8$ rad/s;
- Stop angle, $\varphi_{stop} = 0,16$ radians.

Figure 6.12 shows the platform's initial state (left) and the positioning auto-adjustment (right) using learned policy [23].

6.5.1.2 Verification

The learning of the agent was executed on the real robot after a simulation with the same external parameters. The learning process took 1440 iterations. A real learning process takes more iterations on average because the real system has noise and sensor errors. Figure 6.13 illustrates the result of execution of a studied control system used to turn modules to the center, which is on the rear right side of the images [24].

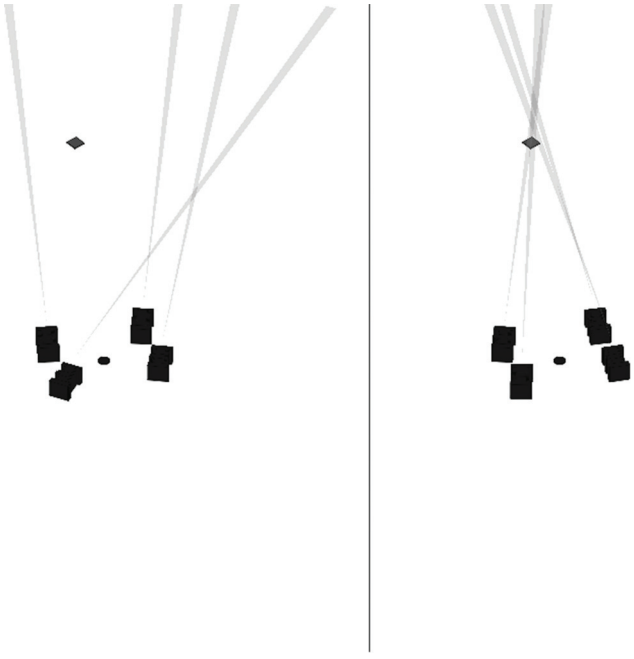


Figure 6.12 Initial and Final Agente Positions.

Figure 6.13 Execution of a Learned Control System to turn modules to the center, which is placed on the rear right relative to the platform.

6.5.2 Learning of the Turning of a Module-Agent

This subsection describes multi-agent learning for producing an efficient control law in the case of cooperative motion using an individual module's speed. The module's desired linear speed A_v should be derived through the learning process relative to the head agent so that the whole platform is moved in a circular motion.

Let the state of the module be represented by $s_t = \{v_t, \vec{d}_i^{err}\}$, where v_t is the current value of linear speed, and $\vec{d}_i^{err}$ is the error vector calculated by (6.7). Action set $A_v = \{\emptyset, v_+, v_-\}$ is represented by the increasing/decreasing of the linear speed from Table 6.2 and action $a_t \in A_\nu$ is a change of forward speed $\Delta\nu^t$ for given moment in time t.

The virtual agent receives error information for each module and calculates the displacement error. This error can be positive (module ahead of the platform) or negative (module behind of the platform). The learning process progresses toward the minimization of error $\vec{d}_i^{err}$ for every module. The maximum reward is given for the case where $\vec{d}_i^{err} \rightarrow 0$, and a penalty is given when the position of the module deviates from the predefined position.

The value of the reward is defined as:

$$r_v^{'} = \begin{cases} 1, & \vec{d}_{i\,err} > \vec{d}_{i\,err}^{t} \\ -1, & \vec{d}_{i\,err} < \vec{d}_{i\,err}^{t} \\ 10, & \vec{d}_{i\,err}^{t} = 0 \end{cases} . \tag{6.7}$$

6.5.2.1 Simulation

Figure 6.14 shows the experimental results of the cooperative movement after learning positioning [23]. It takes 11000 epochs on average. The external parameters of a simulation are:

- Learning rate α = 0,4;
- Discount factor γ = 0,7.

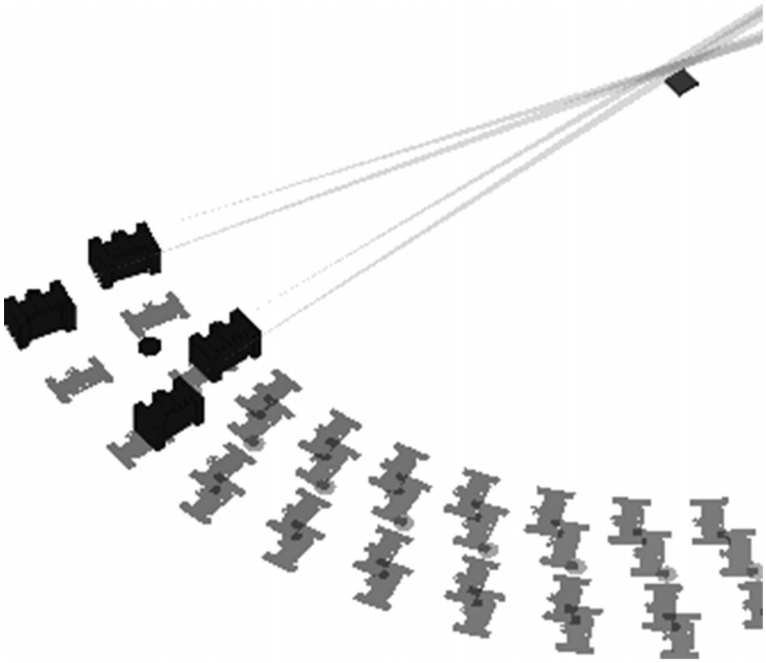


Figure 6.14 Agents Team Driving Process.

During the modules learning, the control system did not use any stabilization of the driving direction. This is because a virtual environment has an ideal, flat surface. In the case of the real platform, stabilization will be provided by internal controllers of the low-level module software. This allows us to consider only the linear speed control.

6.5.2.2 Verification

The knowledge base of the learned agents was transferred to the agents of the control system on the real robot. Figure 6.15 demonstrates the process of the platform moving by the learned system [25]. At first, modules turn in the driving direction relative to the center of rotation (the circle drawn on white paper), as shown in screenshots 1–6 in Figure 6.15. Then, the platform starts driving around the center of rotation in screenshots 7–9 in Figure 6.15. The stabilization of the real module orientation is based on a low-level controller with feedback. This controller is provided by the software control system of the robot. It helps to restrict the intellectual control system by manipulating the linear speed of modules.

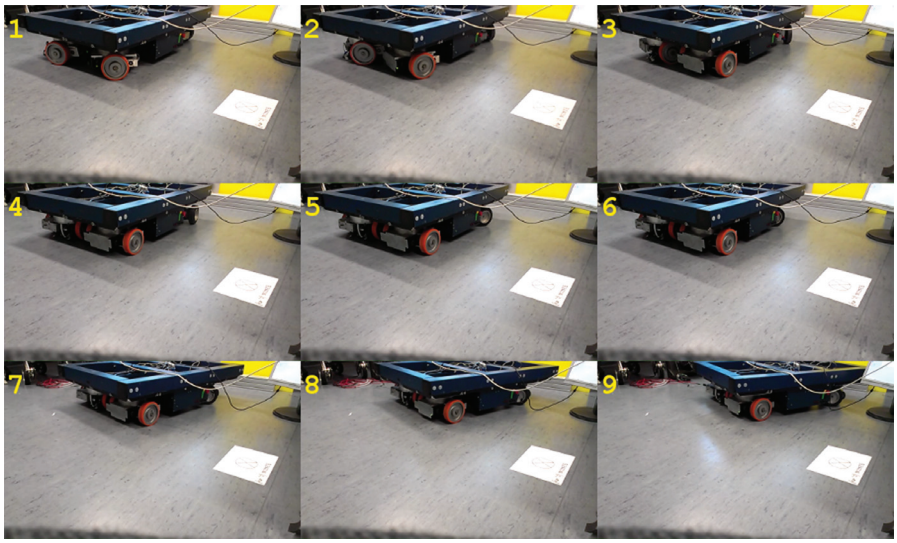


Figure 6.15 The Experiment of modules turning as in the Car Kinematics Scheme (1–6 screenshots) and movement around a White Beacon (7–9).

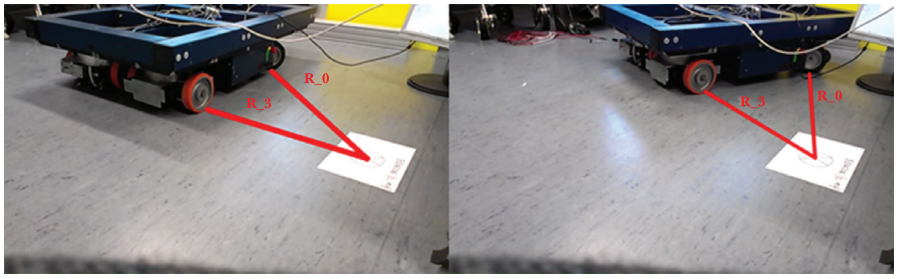


Figure 6.16 The Experiment shows that the radius doesn't change during movement.

The distance to the center of rotation is always the same on the entire trajectory of the platform. This is confirmed by Figure 6.16. Hence, the robot drives around in a circle where the coordinates of the center and the radius are known.

6.6 Conclusions

This paper focuses on an efficient, flexible, adaptive architecture for the control of a multi-wheeled, production, mobile robot. The system is based on a decomposition into a holonic, homogenous, multi-agent system and on influence-based, multi-agent reinforcement learning.

The proposed approach incorporates multiple *Q*-learning agents, which permits them to effectively control every module relative to the platform. The learning process was divided into two parts:

- *Module positioning* – where agents learn to minimize the error of orientation;
- *Cooperative movement* – where agents learn to adjust the desired velocity to conform to a desired position in formation.

A head agent is used to coordinate modules through the second step of learning. From this decomposition, every module agent will have a separate control policy for both forward and angular velocity.

The reward functions are designed to produce efficient control. During learning, agents take into account the current reward value and the previous reward value that helps to find the best policy of agent actions. Altogether, this provides efficient control where agents must cooperate with each other and use the policy of least resistance between each other on a real platform.

The advantages of this method are as follows:

- *Decomposition* means that instead of trying to build a global Q-function, we can build a set of local Q-functions;
- *Adaptability* – the platform will adapt its behavior for a dynamically assigned beacon and will auto-reconfigure its moving trajectory;
- *Scalability and generalization* – the same learning technique is used for every agent, for every beacon position, and for every platform configuration.

In this chapter, we showed successful experiments with the real robot where the system provides robust steering of the platform. These results indicate that the application of intellectual adaptive control systems for real mobile robots have great potential in production.

In future works, we will consider a comparison of the developed approach to mobile robot steering with existing approaches and will provide further information about efficiency of the developed control systems relative to real control systems.

References

[1] J. C. Andreas, 'Energy-Efficient ElectricMotors, Revised and Expanded', CRC Press, 1992.

[2] A. T. de Almeida, P. Bertoldi and W. Leonhard, 'Energy efficiency improvements in electric motors and drives', Springer Berlin, 1997.

[3] R. Stetter, P. Ziemniak and A. Paczynski, 'Development, Realization and Control of a Mobile Robot', In Research and Education in Robotics-EUROBOT 2010, Springer, 2011:130–140.

[4] U. Dziomin, A. Kabysh, V. Golovko and R. Stetter, 'A multi-agent reinforcement learning approach for the efficient control of mobile robot', In Intelligent Data Acquisition and Advanced Computing Systems (IDAACS), 2013 IEEE 7th International Conference on, 2, 2013:867–873.

[5] Y. Mei, Y.-H. Lu, Y. C. Hu, and C. G. Lee, 'Energy-efficient motion planning for mobile robots', In Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on, 5, 2004:4344–4349.

[6] S. Ogunniyi and M. S. Tsoeu, 'Q-learning based energy efficient path planning using weights', In proceedings of the 24th symposium of the Pattern Recognition association of South Africa, 2013:76–82.

[7] Y. Mei, Y.-H. Lu, C. G. Lee and Y. C. Hu, 'Energy-efficient mobile robot exploration', In Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on, 2006: 505–511.

[8] N. Ceccarelli, M. Di Marco, A. Garulli and A. Giannitrapani, 'Collective circular motion of multi-vehicle systems with sensory limitations', In Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on, 2005:740–745.

[9] N. Ceccarelli, M. Di Marco, A. Garulli, and A. Giannitrapani, 'Collective circular motion of multi-vehicle systems', Automatica, 44(12): 3025–3035, 2008.

[10] D. Benedettelli, N. Ceccarelli, A. Garulli and A. Giannitrapani, 'Experimental validation of collective circular motion for nonholonomic multivehicle systems', Robotics and Autonomous Systems, 58(8):1028–1036, 2010.

[11] K. Fischer, M. Schillo and J. Siekmann, 'Holonic multiagent systems: A foundation for the organisation of multiagent systems', In Holonic and Multi-Agent Systems for Manufacturing., Springer, 2003: 71–80.

[12] N. Vlassis, 'A concise introduction to multiagent systems and distributed artificial intelligence', Synthesis Lectures on Artificial Intelligence and Machine Learning, 1(1):1–71, 2007.

[13] L. Gasser, 'Social conceptions of knowledge and action: DAI foundations and open systems semantics', Artificial intelligence, 47(1): 107–138, 1991.

[14] C. Gerber, J. Siekmann and G. Vierke, 'Flexible autonomy in holonic agent systems', In Proceedings of the 1999 AAAI Spring Symposium on Agents with Adjustable Autonomy, 1999.
[15] C. Gerber, J. Siekmann and G. Vierke, 'Holonic multi-agent systems', Tech. rep. DFKI Deutsches Forschungszentrum fr Knstliche Intelligenz, Postfach 151141, 66041 Saarb.
[16] C. Castelfranchi, 'Commitments: From Individual Intentions to Groups and Organizations', In *ICMAS*, 95, 1995:41–48.
[17] P. Stone and M. Veloso, 'Multiagent Systems: A Survey from a Machine Learning Perspective', Autonomous Robots, 8(3):345–383, 2000. [Online]. http://dx.doi.org/10.1023/A%3A1008942012299
[18] A. Kabysh and V. Golovko, 'General model for organizing interactions in multi-agent systems', International Journal of Computing, 11(3): 224–233, 2012.
[19] A. Kabysh, V. Golovko and A. Lipnickas, 'Influence Learning for Multi-Agent Systems Based on Reinforcement Learning', International Journal of Computing, 11(1):39–44, 2012.
[20] A. Kabysh, V. Golovko and K. Madani, 'Influence model and reinforcement learning for multi agent coordination', Journal of Qafqaz University, Mathematics and Computer Science, 33:58–64, 2012.
[21] A. G. Barto, 'Reinforcement learning: An introduction', MIT press, 1998.
[22] W. Ren and N. Sorensen, 'Distributed coordination architecture for multi-robot formation control', Robotics and Autonomous Systems, 56(4):324–333, 2008.
[23] [Online]. https://www.youtube.com/watch?v=MSweNcIOJYg
[24] [Online]. http://youtu.be/RCO-j32-ryg
[25] [Online]. http://youtu.be/pwgmdAfGb40