Chapter

Skill Acquisition for Resource-Constrained Mobile Robots through Continuous Exploration

Markus D. Kobelrausch and Axel Jantsch

Abstract

We present a cognitive mobile robot that acquires knowledge, and autonomously learns higher-level abstract capabilities based on play instincts, inspired by human behavior. To this end, we (i) model skills, (ii) model the robot's sensor and actuator space based on elementary physical properties, and (iii) propose algorithms inspired by humans' play instincts that allow the robot to autonomously learn the skills based on its sensor and actuator capabilities. We model general knowledge in the form of competencies (skills) of the mobile robot based on kinematic properties using physical quantities. Thus, by design, our approach has the potential to cover very generic application domains. To connect desired skills to the primitive capabilities of the robot's sensors and actuators, it playfully explores the effects of its actions on its sensory input, thus autonomously learning relations and dependencies and eventually the desired skill. KnowRob is used for knowledge representation and reasoning, and the robot's operation is based on ROS. In the experiments, we use a millirobot, sized 2 cm^2, equipped with two wheels, motion, and distance sensors. We show that our cognitive mobile robot can successfully and autonomously learn elementary motion skills based on a playful exploration of its wheels and sensors.

Keywords: artificial intelligence, autonomous learning systems, cognitive architecture, reinforcement learning, knowledge representation and reasoning, resource-constrained systems, low-energy mobile robots

1. Introduction

Our starting point is a robot with (a) a set of sensors and actuators, (b) tight resource limitations, (c) access to a database that captures general motion-related competencies (e.g. moving along a rectangle or navigating to a target location), and (d) built-in assumptions about physical laws and geometric relations. Our objective is to develop methods that allow the robot to autonomously learn competencies stored in the database.

Initially, the robot does not know the meaning and effect of its sensors and actuators (e.g. if an actuator controls a LED or a wheel). Therefore, the first activities are concerned with learning the meaning of its sensors and the effects of its actuators. Then, basic competencies from the knowledge base are acquired followed by increasingly complex competencies. A priori, the robot only has built-in knowledge of how to interface its sensors and actuators and basic assumptions about physical laws and geometric relations, but not what the sensors and actuators mean or how a specific motion can be accomplished.

Our long term goal is to provide the robot with general methods that allow the robot to work with any kind of sensors and actuators, in any kind of physical environment, and learning any kind of competence, provided it is possible at all (e.g. if the robot has only LEDs but no motors, it cannot learn to move).

We consider this a worthwhile vision because this approach to minimize prior knowledge and assumptions will facilitate very flexible systems that can work with any kind of sensors and actuators, in wheel-equipped or flying robots, on level plains, rocky or grassy surfaces, or even in wet environments. It will allow the use of accurate or inaccurate sensors and actuators, and to adapt to aging and wear-out effects. This approach is general because the only assumptions we make are the laws of kinematics and geometry, the availability of and access to sensors and actuators, the availability of learning methods (e.g., RL), and the availability of a database describing the skills to be learned.

While this is our vision, in this article, we make the further assumption that the robot knows the meaning of its sensors and operates in a two-dimensional plane. Inspired by the play instinct observed in humans and animals we propose exploratory, hierarchical learning. Simple and elementary tasks are tried out and learned first, followed by complex and composite tasks. This means the robot starts by asking if it can move at all, then it tries to learn elementary linear and angular motions, based upon which it studies moving along rectangles and similarly simple shapes. For each learning task, we use Reinforcement Learning (RL) as it matches well the exploratory nature of the robot's setting. The learning tasks are identified based on entries from a knowledge database that describes the motion skills and the hierarchical relation between skills. Specifically, we use the KnowRob knowledge processing system [1], which is designed to provide autonomous robots with the knowledge base for performing motion and manipulation tasks.

In this paper, we propose and demonstrate the Skill Acquisition Method (SAM) for the case of a wheel-equipped tiny robot operating on a smooth, level plain; in future work, we will show that the same techniques generalize to other settings and environments. We evaluate our approach in a simulation environment for a two-wheeled and a four-wheeled mobile robot moving in a two-dimensional space. Experiments show that the system can learn and interpret its basic motion commands and derive complex motions, and finally, it succeeds in driving a rectangle (set of basic motion commands). Our contributions are summarized as follows:

i) We identify a minimal set of prior knowledge mandatory for learning basic movements.

ii) We propose a cognitive system behavior, the Playful Continuous Competence Acquisition (PCCA), that enables the learning and development of skills based on

a) the model of generic competencies (skills),

b) and the system's Sensor and Actuator Space (SAS) grounded in elementary physical properties.

2. Related work

The use of knowledge representation and reasoning in robots has a long tradition, where the *Shakey* robot had already 1984 an internal representation of its environment [2]. Extensive research has been done in robotics and artificial intelligence in recent decades, to which this article mainly refers. Since robots have specific demands on knowledge bases and appropriate methods, e.g., linking abstract knowledge representation and specific control systems, this can be best solved with frameworks explicitly designed for this purpose.

In this context, KnowRob was specifically developed to equip autonomous robots with knowledge and methods (Knowledge Representation and Reasoning (KR&R)) to perform everyday manipulation tasks and to provide an infrastructure for cognitively enabled robots [1, 3, 4]. It represents one of the most advanced knowledge processing systems for robots, which has evolved even further with OPEN-EASE ([5]), which integrates KNOWROB2 ([6]), and aims to provide a remote knowledge and reasoning service that offers unprecedented access to the knowledge of autonomous robotic agents performing human-scale manipulation tasks. This seems promising for agents performing such rich human-scale manipulations but also places significant demands on the system's resources, which is crucial for systems with limited resources. Therefore, we use KnowRob as the basis for knowledge processing and representation to take full advantage, but we target the approaches and methods that allow it to be deployed in such tiny systems.

A recent work dealing with the generalization of experience into abstract knowledge for novel situations, entitled Socio-Physical Model of Activities (SOMA [7]), consists of a comprehensive model for connecting physical and social entities that enable flexible execution by robotic agents. Since this representation seems essential, we use a similar approach, keeping our model flat in the first line due to resource-constraints. This limits the flexibility of the application (smaller knowledge base) but is crucial, and we aim for a reasonable trade-off. In this context, we also discuss a set of a small amount of prior knowledge.

RoboEarth has similar goals and approaches to our work [8]. Capabilities are also modeled, where we differ mainly in how they are used. We assume a set of general prior knowledge and basic methods to acquire skills, while their work accepts more complex algorithms to derive specific knowledge. Additionally, we further evaluate and improve skills to achieve continuous development.

Other works also deal with systems that learn semantically from different experiences, taking different approaches [9]. While learning relies on recorded experiences in semantic structures containing high-level representations. A key difference in our approach is that we generate skill-specific episodic knowledge through real-time learning methods, leading to knowledge abstraction at an earlier stage. To further leverage this, we define a set of prior knowledge that must be present to enable use in resource-constrained systems.

3. Skill acquisition

In the following, we present SAM, which starts from a set of general assumptions (knowledge and methods) to autonomously acquire and develop specific complex capabilities and aims for generic deployment in resource-constrained systems.

3.1 Overview

SAM (**Figure 1**) consists of various elements structured in layers. The bottom layer reflects the physical part, i.e., the robot and its environment. The layer above hosts the central computational agent, which abstracts the interface to the environment via the SAS. This general and generic interface is deliberately based on the fundamental physical properties of sensors and actuators. Thus, by definition, any environment can be integrated elegantly and efficiently as long as it follows the matching properties, defined in 3.4. Further, the agent has access to the knowledge base and reasoners. SAM follows a cognitive-behavioral architecture to autonomously learn skills using a KR&R methods combined with real-time learning from the physical environment.

3.1.1 Cognitive model

Cognitive models go beyond traditional behavioral models regarding what an entity (robot) knows, how that knowledge is acquired, and how it can be used. As a result, they are becoming increasingly popular in artificial intelligence. They are well suited for implementing highly autonomous systems that exhibit some intelligence and are expected to develop over time. There are several approaches to these models in the literature, particularly in robotics, which attempt to mimic the behavior of intelligent agents based on human cognition. Recent work on a generic form of this, such as the Socio-physical Model of Activities (SOMA) consists of a comprehensive model that combines physical and social entities and allows flexibility of execution by robotic agents through symbolic reasoning [7].

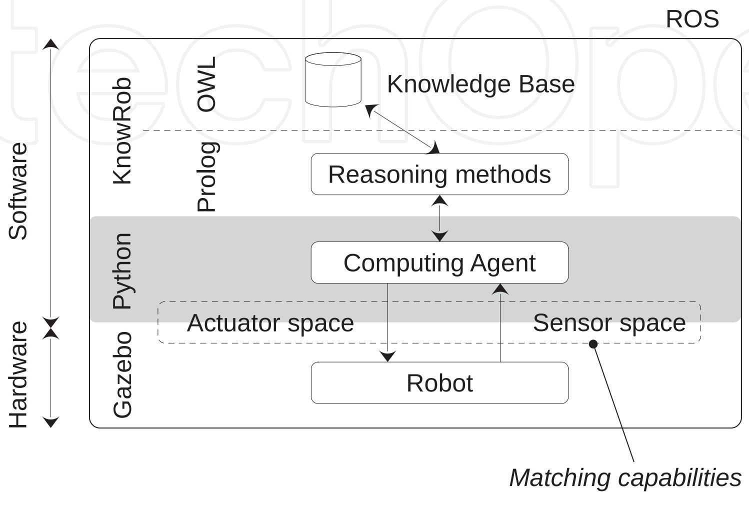

Figure 1.
SAM overview with its layered architecture and the distinction between software and hardware. The KnowRob layer consists of methods for KR&R, while the computing agent (python) drives the system flow to autonomously acquire skills. It has access through the SAS to the physical environment and the database. All components are integrated in ROS.

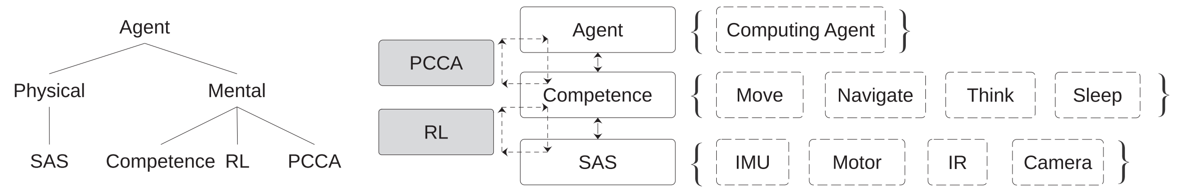

Figure 2.
SAMs cognitive system model. The left side shows the distinction between physical and mental entities. The right side shows their relations, where an agent seeks to acquire and further improve a particular competence using the PCCA method. RL methods are used to learn a specific skill while interacting when required with SAS to access the physical environment. The modules in the dashed line show examples of instances.

Since cognitive models, in the broader sense, represent complex processes and behavior, we focus our modeling on the core elements that we consider essential for our millirobot to acquire basic skills. In the medium term, we intend to adapt them to SOMA.

Figure 2 illustrates our proposed cognitive system model, divided into the physical and mental domains. The SAS depends on the physical properties of the robot. We divide the mental part into elementary capabilities and behavioral methods. A competence reflects knowledge of a particular skill acquired and applied through the PCCA behavioral method, while RL is used to learn a specific competence (e.g. motion commands). We will introduce and discuss these essential elements step by step in this article.

3.1.2 Use of KR&R and episodic memory

For each activity that SAM performs and observes (physical interaction, knowledge inference, learning, etc.), it generates skill-specific knowledge as episode memory and stores it with timestamps. Such episodic memory could include what the robot saw, reasoned, and did, how it did that, why, and what effects it caused [5]. It can be used for further conclusions and learning at any time. While the size and scope of the episodic memory directly relate to the resources required for the particular system. Many approaches attempt to collect a large amount of extensive detailed knowledge, which directly impacts computing time. This seems impractical for systems with limited resources. Therefore, we propose to keep episodic knowledge flat and small and to store only highly relevant information. In this context, we also consider a set of general prior knowledge that an intelligent system must have to learn and exhibit sufficient episodic memory for a given skill. We argue that these two facts are essential to consider for use in systems with tight resource limitations. Section 3.5.1 outlines an approach to a set of concrete prior knowledge and episodic knowledge developed by SAM, intended for use in resource-constrained systems.

Our long-term vision is that all relevant parts of the proposed SAM are hosted on such a system, e.g., a tiny millirobot powered by a micro-controller. We are aware of the challenges of migrating databases and logical reasoning to resource-constrained systems. As an intermediate step, we propose separating the acquisition and exploitation phases, where the system has access to KR&R in the first phase. Once the skill has been successfully acquired (sufficient episodic memory) to some degree, the system may be able to master it independently. Then it exploits the acquired skill with appropriate methods on the tiny millirobot. Whenever the system detects significant changes or decides to search for new capabilities, it contacts the database again. In this way, we can elaborate a similar knowledge acquisition behavior for resource-constrained systems compared to those with fewer constraints that host KR&R directly.

3.2 Competence

A central core element of SAM is competence, generally understood as mental property. The focus of this work is on the modeling of competencies that, when defined, lead to physical actions of the system through the SAS. However, competence in itself does not always have to be related to the physical facts of the system. It could also be a purely mental ability, such as spatial awareness, concentration, attention, reasoning, logic, and so forth. To model capabilities in an intelligent system, essential basic elements of those capabilities must be considered to grant an appropriate developmental progression. In a nutshell, a system should learn a skill independently and reason with appropriate knowledge about how good that skill is. Moreover, the evaluation of skills is of particular interest, used to continuously improve the respective skills. In this way, a cognitive system that also has an interest in developing itself further can become better over time.

In this context, two fundamental elements of competence have been attributed. These are (i) *fitness*, which is a statement of how well system masters the skill, and (ii) *learnability*, which indicates a skill that can be learned by the agent.

Figure 3 illustrates the general concept of competence modeled in the knowledge base. The *fitness* is represented with a numeric value and the *learnability* with a boolean value. The *learnability* is fulfilled if (a) all properties for learning the skill are satisfied, and (b) the system provides methods to learn this competence. The properties of (a) can be determined either by inference knowledge from the database or, if they depend on the physical space, directly by physical interaction. For instance, in the case of the movement skill, we determine the physical agent's ability to move through physical interactions (Section 3.5.1). For (b), certain methods must be in place to learn specific skill knowledge. Such knowledge could be, for example, a set of specific actions and their command values. We use RL to learn specific motion commands executed via SAS. Other learning methods such as Deep RL or supervised/unsupervised methods could also be utilized. However, the goal is to acquire a subset of episodic memory sufficient to exhibit a particular skill. The *fitness* is used to evaluate how well the skill is mastered and is represented by a number from 0 to 100, with 100 being the maximum achievable. For example, we directly assign the RL method reward to *fitness* of a basic motion competence (Section 3.5.4). In addition to the general properties of competence, a corresponding instance may also store specific

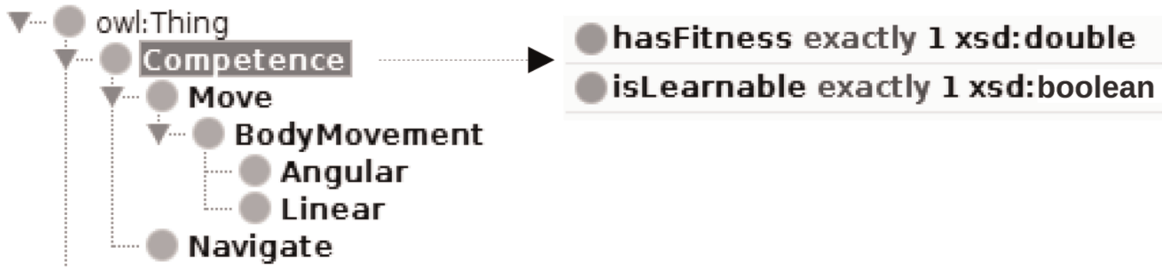
owl:Thing
Competence
Move
BodyMovement
Angular
Linear
Navigate
hasFitness exactly 1 xsd:double
isLearnable exactly 1 xsd:boolean

Figure 3.
The competence entity with its two fundamental properties (learnability and fitness), modeled in the knowledge base using ontologies.

knowledge relevant to the execution of skill in a particular system. In our case, we memorize the action commands, their fitness, and the timestamp, as discussed in Section 3.5.5.

3.3 Playful continuous competence acquisition

Another key core element is behavior, which ensures continuous development by learning new skills and further improving existing ones. Generally, a system that acquires specific skills should not consider them finally learned after the first success. Instead, the goal is to evaluate what has been learned and, if necessary, to develop further and improve it. In this way, a system can evolve autonomously and continuously adapt to certain changes in its environment. To this end, we consider the following key behavioral elements crucial: (a) the striving for new skills and (b) the continuous improvement of already learned skills.

Figure 4 illustrates our proposed PCCA method, focusing on knowledge acquisition and skill development. An interpretation of the learned skills in terms of possible application scenarios and their combinations in specific contexts, i.e., for which purpose skill could be used, is future work and not considered here. Further, to generalize the high-level system flow, a promising approach would be to model it directly in the knowledge base in tasks and actions. For that, KnowRob offers a promising approach that might also be applicable to our system [1, 7].

However, SAM's high-level behavioral process is determined using the PCCA reasoner, directly queried by the computing agent. We define two different high-level-behavioral phases acting on the competence model properties (*fitness* and *learnability*), shown in **Figure 4**: *(i) seeking for new competencies* and *(ii) improving known competencies*. Phase *(i)* and *(ii)* are general cognitive-behavioral patterns based on the competence model (presented in Section 3.2) that are independent of the skill being learned. Whereas skill-specific learning methods (dashed lines in **Figure 4**),

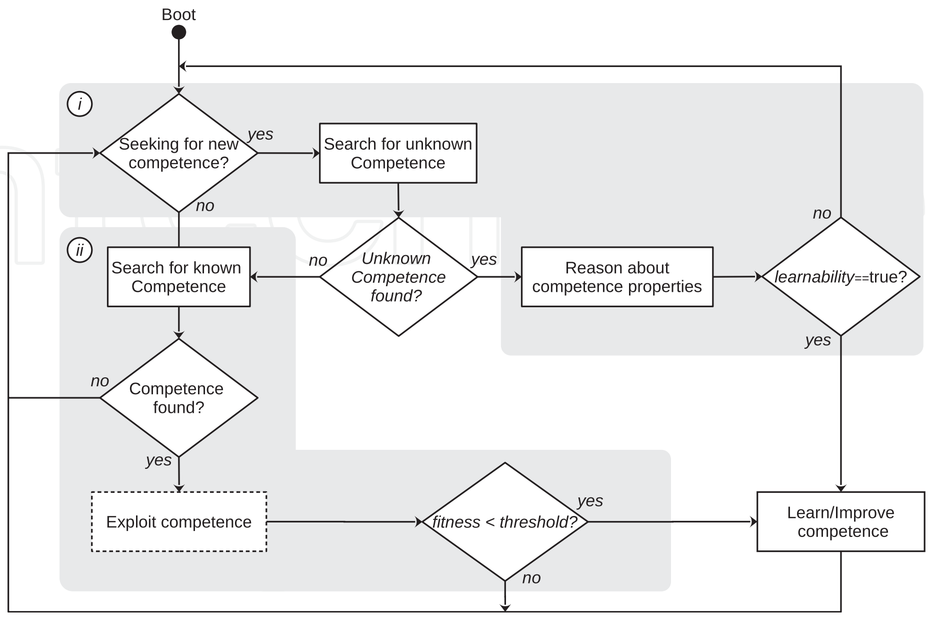
Boot
i
Seeking for new competence?
yes
Search for unknown Competence
no
ii
Search for known Competence
no
Unknown Competence found?
yes
Reason about competence properties
no
learnability==true?
yes
no
Competence found?
yes
Exploit competence
fitness < threshold?
yes
no
Learn/Improve competence

Figure 4.
The PCCA flow is divided into two high-level-behavioral phases, (i) and (ii). It acts based on the competence properties (fitness and learnability). It covers key behavioral elements (a) and (b) by utilizing phases (i) seeking for new competencies and (ii) improving known competencies, in an incremental fashion. The elements marked by dashed lines represent skill-specific learning methods.

triggered by the PCCA, acquire the respective competence-specific knowledge (e.g. an RL element for a navigation skill). It switches playfully between these two phases and can thus develop and improve over time.

3.3.1 Seeking new competencies

SAM searches for new capabilities based on the instances available in the knowledge base. Currently, these still need to be instantiated manually, with the long-term goal being to create them automatically. If one is present, the system uses the competence's *fitness* property to determine if it is already known and learned. If not, the *learnability* property is used to determine if it can be learned. If yes, it enters the skill-specific learning phase, and otherwise, it continues searching.

3.3.2 Improving known competencies

The system decides whether a competence can still be improved based on the *fitness* property. When the *fitness* value is below a certain threshold, SAM relearns the skill by re-running the RL method exploration phase. If a better solution is found, it memorizes it as the best for further use. Moreover, it operates on an incremental basis, ensuring that the best solution is found after a certain period of time. It further allows to react to changes in the environment and thus make immediate adaptations.

3.3.3 Skill-specific learning methods

A specific competence is explored, learned, and exploited using appropriate learning methods (RL, supervised/unsupervised learning). These methods are competence specific and must be designed according to the particular skill. In principle, it is possible to integrate highly optimized learning algorithms for the respective functions. However, our goal is to use basic algorithms and execute them using general knowledge modeled in the knowledge base. In this way, we expect even more flexible usage, where only the primary parameters in the database need to be adjusted while the algorithm remains the same. When needed, the skill-specific learning method is triggered by the PCCA. In Section 3.5.4, we further discuss this approach and propose an RL basic algorithm that we extend with methods from KR&R to achieve generalization.

3.4 Sensor and actuator space

The sensor and actuator space (SAS) represents a generic interface to the robot environment, solely based on physical quantities. For example, consider an Inertial Measurement Unit (IMU), an odometry sensing unit as sensors, and two motors as actuators. SAM's *matchingcapabilities* rely on the physical quantities of those sensors and actuators that the robotic-system must provide. **Figure 5** illustrates the resulting abstracted interfaces for sensors (ψ, x and y) and actuators (m_1 and m_2), with their physical quantities shown in **Table 1**.

We assume that these interfaces abstract the robot-specific sensor data and actuator commands. For example, how the respective motor of the robot is controlled (using a motor controller that takes the acceleration properties into account) needs to

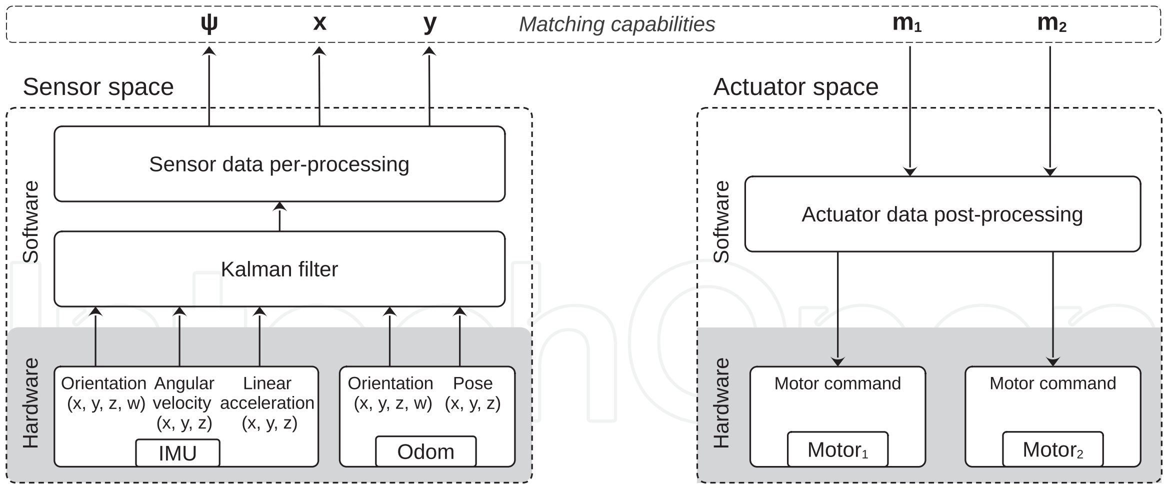

Figure 5.
Example of a sensor and actuator space (SAS) consisting of two motors, an IMU, and an odometry sensing unit. The respective data's pre- and post-processing is robotic-system-dependent and must be addressed individually. Thus, SAS must abstract the low-level data appropriately to meet SAM's matching capabilities.

Data	Description	Physical Quantity	Unit
Sensor space			
ψ	Yaw rotation	Angle	°
x	Horizontal cartesian coordinate	Length	m
y	Vertical cartesian coordinate	Length	m
Actuator space			
m_1	Wheel torque	Torque	N m
m_2	Wheel torque	Torque	N m

Table 1.
Matching capabilities: SAS with its physical quantities.

be modeled robotic-system-dependent. In our case, the respective sensor data pre-processing layer and the actuator data post-processing layer take care of this.

This approach is generic, and we argue that the system initially does not need to know which actuators or sensors it is dealing with. A long-term goal is to employ appropriate methods and knowledge to identify and learn its capabilities. The Semantic Sensor Web follows this approach, annotating sensor data with various semantic metadata (including physical quantities) [10]. Further, there is promising work in automatic semantic knowledge acquisition for sensor data, which aims to annotate raw data with semantic knowledge [11]. Thus, our approach aims to leverage generic interfaces to integrate those methods seamlessly in future work.

However, the specific experimental setup is illustrated in **Figure 5** for a two-wheeled mobile robot. It is equipped with two motors (for a 4-wheeled robot, extended by two additional motors), each driving a wheel, an inertial measurement unit (IMU), and an odometry sensing unit (obtained from the simulation environment) that is used to reduce the drift error of the IMU over time using a Kalman filter [12]. We are well aware of the challenges to the precision of these sensor measurements required for stable localization, which is extensively discussed in many publications [13, 14]. However, we do not further discuss this and assume that the problem is well understood. In conclusion, with this generic design, any robot environment can interact with SAM as long as the required physical *matchingcapabilities* are supported.

3.5 Motion skills

As mentioned earlier, this work focuses on modeling competencies that lead to physical actions of the system through SAS. Considering this fact and the physical characteristics of a wheeled robot, specifically the actuators in the form of wheels, potential movement possibilities can be assumed. For that, we consider basic movements, which in turn are subdivided into atomic and more complex movements. In a broader sense, for atomic actions, the robot is assumed to always be stationary, moving by applying torque to the actuators and stopping when it is removed. Such an atomic motion thus represents a sub-element of a more complex motion. It is not claimed that those movements are the most efficient in terms of smoothness and speed. However, they still allow the robot to approach all positions in a given space. **Figure 6** illustrates a set of motion skills where atomic movements such as angular and linear movements ground complex movement patterns such as rectangles, cycles, or even more generally, a navigation path. The acquisition of these skills occurs in the same hierarchical manner that enhances the physical learning methods discussed in the next section.

3.5.1 Hierarchical knowledge acquisition

Let us first consider the knowledge we can gain about a movement, which we draw from a small set of prior knowledge. Assuming the system has not yet acquired any specific knowledge about motion, it has first to find out whether it can move at all with its given actuators: (*I*)*"Am I able to move?"* To answer this question, the system initiates random actions and observes their consequences. In our case of a two-wheeled robot, both actuators are moved randomly, and the physical effects are evaluated based on a spatial position change. At the level (*I*) the question is only about the possibility of any movement, as depicted in **Figure 7**. If the system has an actuator that controls only a LED, it would be recognized as irrelevant for movements. Next, at level (*II*), we can start asking for basic movement patterns without specific lengths or angles. (*II*) *"Am I able to turn forward/backward/left/right?"* The actuators are triggered again, and SAM searches for angular (left/right) and linear (forward/backward)

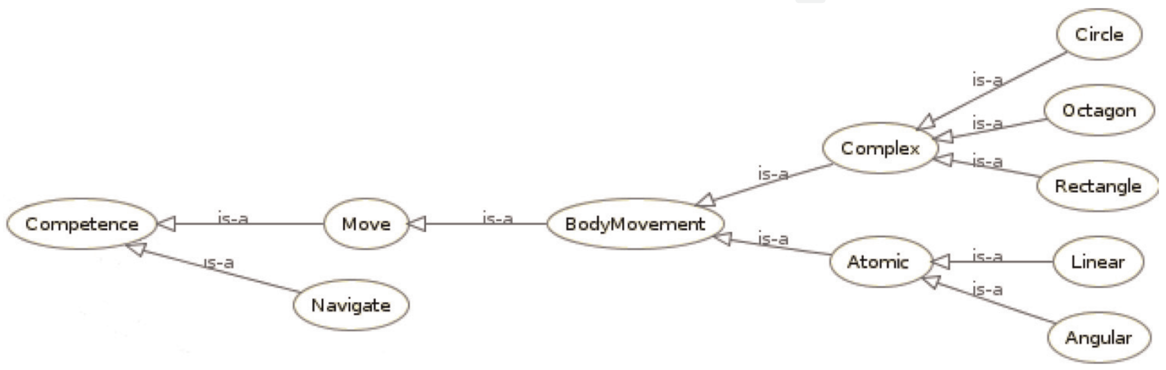

Figure 6.
Competence graph (modeled in the database), with a set of motion skills, sub-divided into atomic and complex. Where the atomic movements form the basis for more complex patterns.

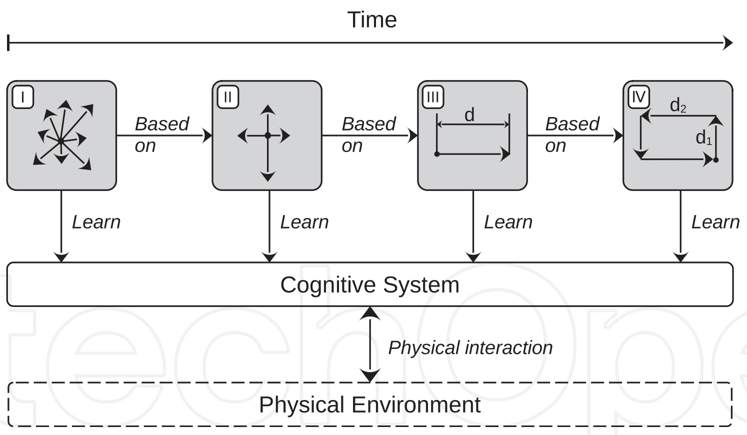

Figure 7.
The development of hierarchical knowledge over time. The motion skill acquisition starts with the fundamental question (I)"Am I able to move?", followed by (II) "Am I able to turn forward/backward/left/right?", (III) "Am I able to move a specific length/angle?" and (IV) "Am I able to follow a specific rectangular path?". While SAM draws associations directly from actions performed in the physical environment to answer these questions.

movement patterns. Turning left/right may be caused by a two-wheeled robot turning one wheel forward while the other wheel is moving backward, where forward/backward patterns may result from driving both actuators simultaneously. Hence, the system learns general natural language-based motion patterns. At the next level (III) these rules are used to learn a specific distance, say 1 cm, and angle, say 10°. Further building on this, more complex movements are learned at level (IV), which in turn consist of a series of specific movements. For example, for a rectangle with lengths of 3 cm and 2 cm, the following sequence of commands would be constructed: three times straight 1 cm, then 9 times left with 10°, two times straight with 1 cm, and so on until the rectangle is closed. Following this hierarchical knowledge acquisition approach, we can significantly limit the search space and thus bootstrap the learning performance $I - III$.

3.5.2 Basic motions

For an atomic, basic motion, we refer to the basic kinematic and dynamic properties of a system, where kinematics describes the relationship between coordinates in motion space. Dynamics correlates the torque and force in each joint (wheels of the robot) with the acceleration of the joint and the velocity over time. When the wheels touch the ground, these forces act indirectly on the overall system and thus cause it to move. With the aid of the kinematic properties, inferences about this resulting motion can be drawn. Motion control for mobile robots is extensively covered in the literature. To navigate accurately, kinematic or dynamic models are used to generate accurate motion commands, considering all effects, including the resulting tracking error [15–19]. We are aware of the challenges of designing or even learning motion controls that lead to accurate robot movements. Thus, our work demonstrates the possibility of a generic approach to learning movements with general knowledge, even if the movements are still subject to certain errors. We will address minimizing this error by following the same general approach in future work.

However, based on universal laws of physics, we derive atomic base motions, illustrated in **Figure 8**. The robot's position is represented by a vector with a pair of numerical coordinates $x(t)$ and $y(t)$ from the cares coordinate system and orientation $\psi(t)$. The robot is indirectly set in motion with constant acceleration by applying an

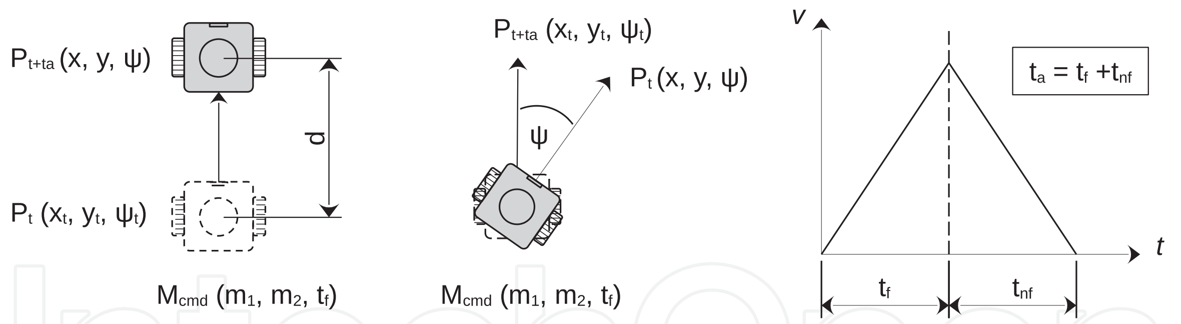

Figure 8.
Basic commands and observations for atomic motions. The left image shows a linear motion, while the middle shows an angular turn. The illustration on the right shows the respective action times and the velocity progression over time.

arbitrary torque m_1 and m_2 of the robot's actuators for a specific time t_f. As soon as this torque is removed, the system brakes with the same constant negative acceleration until it stops after some time t_{nf}. Thus, the atomic motion time is expressed by the total action time of $t_a = t_f + t_{nf}$, as depicted on the right in **Figure 8**. The resulting spatial movement (distance d and yaw angle ψ) for the respective actuator torques is determined by the change in position over time t_a using an inverse kinematic reasoner. Using RL, we search for the best actions (actuator torques and action time t_f) for a given spatial position change. This applies to all atomic actions, while more complex movements are simply composed of a series of atomic actions.

3.5.3 Kinematic reasoner

KnowRob ([1]) provides a kinematic reasoner, which we adapt to our SAM's needs. It derives motion-specific competence knowledge based on general kinematic laws and is utilized during hierarchical knowledge acquisition level $II - IV$. We distinguish two types of motion knowledge, (i) basic movement patterns and (ii) specific motion distances. To reason about (i), we define the following logical rules:

```
is_basic_linear_motion_pattern(X0, Y0, YAW0,
X1, Y1, YAW1, Distance): -.
DX is X1 - X0,
DY is Y1 - Y0,
Angle is wrap(YAW0, YAW1),
Distance is sqrt((DX*DX) + (DY*DY)),
Distance! = 0.0, abs(Angle) == 0.0.
is_basic_angular_motion_pattern(X0, Y0, YAW0,
X1, Y1, YAW1, Angle): -.
DX is X1 - X0,
DY is Y1 - Y0,
Angle is wrap(YAW0, YAW1),
Distance is sqrt((DX*DX) + (DY*DY)),
Distance == 0.0, Angle!= 0.0.
```

A basic linear motion pattern is detected when the robot's angle does not change during an action, but the distance does. Further, we can restrict it to a *forward* motion pattern if the position change has a positive value and *backward* if it is negative.

For an angular movement, the same rules apply. To detect angular motion patterns (*left*, *right*), we use an angle wrap function that calculates the angle moved and the direction of rotation.

For (ii), we define the following reasoner to argue about specific distances and angles used in level *III* of hierarchical knowledge acquisition.

```
is_spatial_motion(X0, Y0, YAW0, X1, Y1, YAW1,
Distance, Angle): -.
DX is X1 - X0,
DY is Y1 - Y0,
Angle is wrap(YAW1, YAW0),
Distance is sqrt(((DX*DX) + (DY*DY))) .
```

These rules represent a general knowledge of the kinematic properties of a two-dimensional system, where we argue that SAM can be easily extended to three-dimensional systems by adding appropriate kinematic reasoners.

3.5.4 Skill specific learning methods

A specific competence is explored, learned, and exploited using appropriate skill-specific learning methods. Since SAM primarily focuses on acquiring skills that lead to physical actions, the respective atomic motion commands have to be learned in real-time by interacting with the environment. An appropriate learning procedure is required, whereas reinforcement learning methods achieve good results in this domain. The method dates back to the early 1990s when Q-learning was already used to learn specific, mostly robotic, tasks. However, many works solve various tasks with RL, whereby these are primarily designed in a context-specific, goal-directed manner and without explicit general prior knowledge, which significantly limits the learning of complex skills. We attempt to overcome this with our approach by using generally formulated prior knowledge to learn skill-specific, in our case, atomic motion commands.

In the following, we introduce our RL-based approach, which we extend with KR&R methods. In RL, an agent interacts with its environment over periods of discrete time steps t. An action a_t is taken following a policy π based on the observed state s_t and the reward r_t, as shown in **Figure 9**. The main difference from traditional RL methods is that we use KR&R to infer the reward and the state. More specifically, the kinematic reasoner is applied to argue with the general kinematic knowledge about the newly observed state s_{t+1}, which in turn is defined as the distance and angle traveled during a time step t. Where the reward r_{t+1} is computed with an RL Reward Reasoner, following Eq. (2) and Eq. (3).

We chose a model-free approach for the specific RL algorithm based on a simple Q-Table RL method ([20]) for resource reasons. Keeping the required resources low seems to be the most intuitive first step for tackling our long-term vision, where all relevant parts of SAM, including RL, are hosted on a resource-constrained system. Q-Learning is a value-based method, where the Q-value is computed from the action-sate value function (Eq. (1)). It seeks to find the optimal Q-value for pairs of states and admissible actions. During exploration, the agent computes and stores them in a Table (Q-Table), where the Q-value indirectly represents the optimal policy π. Once the agent performs exploitation, it simply selects its actions from the Q-Table. This method performs well in systems with limited resources since it scales with the size of

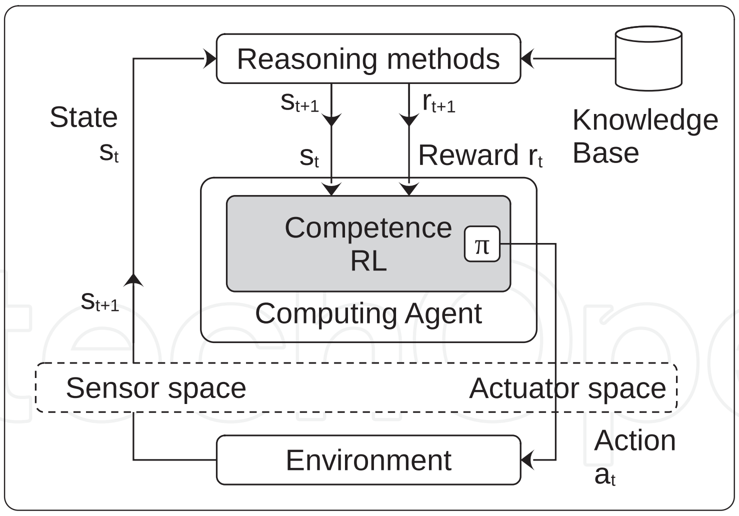

Figure 9.
RL flow with an extension of using KR&R to compute to state and reward using general knowledge.

the Q-Table in terms of resources. One significant challenge is to define the search space well, which directly affects the size of the table. We address this with our hierarchical learning approach, which constrains the respective search space quite well and thus achieves good results with Q-learning.

The Q-value function is defined according to the Bellman equation and notated as:

$$Q^{\pi}(s_t, a_t) = Q(s_t, a_t) + \alpha[r(s_t, a_t) + \gamma maxQ(s_{t+1}, a_{t+1})] \tag{1}$$

Where α is the learning rate, and γ is the discount rate for the expected future reward. The action space consists of the motor force m_1, m_2, and the applied time t_f. The state space is represented by the distance covered d_{t_a} and the angle ψ_{t_a} as well as the time required t_a.

The design of the reward function is formulated to learn specific basic motion distances and angles, while the angular and linear motion skills are learned separately and denoted as:

$$r_{linear}(s_t, a_t) = 100 * \left(1 - abs\left(d_{target} - d_{t_a}\right)\right) \tag{2}$$

$$r_{angular}(s_t, a_t) = 100 * \left(1 - abs\left(\psi_{target} - \psi_{t_a}\right)\right) \tag{3}$$

In principle, they each reflect a simple assumption: the closer a performed basic movement is to the desired distance, the higher the reward for that action. Thus, these rewards can also be considered a piece of specific general knowledge and are assessed by the RL Reward Reasoner. The resulting extended Q-Learning algorithm (Algorithm 1) follows a traditional flow, where the reward r_t and the state s_{t+1} are computed by the Kinematic- and RL Reward Reasoners. Their computation time is essential for systems that learn from the physical environment in real-time. In particular, these decisions must be made in a specific period, especially for tasks requiring a time-dependent control cycle, e.g., the robot is in motion and must receive its commands in time to navigate accurately. In the current work, we have solved this problem by using atomic motions that result in the robot being stationary, eliminating the time-dependent requirements during KR&R. In future work, we will investigate these considerations on real hardware that learns various skills from its environment in real-time.

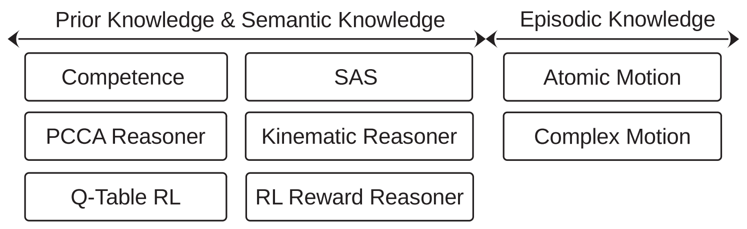

Figure 10.
A set of prior knowledge, including semantic knowledge, about competencies, SAS, PCCA, kinematic properties, and skill-specific learning methods (dashed lines) to acquire motion-related episodic memory.

Algorithm 1: Q-Learning with KR&R interactions.

```
1: Init $Q - Table$ with random data.
2: Observe initial state $s_1$.
3: for episode = 1, N do.
4: Select an random action action $(a_t|\pi)$.
5: Execute $a_t$.
6: Observe new state $s_{t+1} = Kinematic\_Reasoner()$.
7: Observe reward $r_t = RL\_Reward\_Reasoner()$.
8: Calculate Q-value $Q^{\pi}$.
9: Update $Q - Table$.
10: end for.
```

3.5.5 Prior knowledge and episodic memory

As discussed in Section 3.1.2, we propose to keep episodic memory flat and small and to store only highly relevant information. Further, we seek a set of general prior knowledge (semantic knowledge and general methods) that needs to be provided to learn and exhibit sufficient episodic memory for a given skill. We argue that these two facts are essential to consider for use in systems with limited resources. The following outlines how this might be addressed specifically in the case of SAM.

Figure 10 illustrates a set of prior assumptions, including semantic knowledge and general methods for acquiring motion-related episodic knowledge. The KR&R part might be provided by an edge device during the acquisition phase, while for the motion-specific learning, we deliberately propose Q-Table RL that requires few resources and thus can be hosted directly on the tiny millirobot. Further, we memorize only the motion commands learned by the RL with a timestamp and *fitness* to continuously evaluate their performance. In the case of SAM, this amount of episodic memory is sufficient to develop and improve motion skills over time. With this hybrid system flow and the conscious design of a set of generic prior and episodic knowledge, we argue that movement skills can be learned and used even on a system with limited resources. While these are general considerations, we will specifically address this subject on real hardware to consider all implications and requirements in future work.

4. Experiments

To evaluate the proposed SAM, we base our experiments on a simulation of a millirobot. Based on ROS, we use Gaezbo as a simulation environment, a Python ROS node for the computing agent, and KnowRob for the knowledge base and reasoning methods. We show the development phases $I - IV$ (Section 3.5.1), starting with evaluating a principle movement possibility up to the execution of a rectangular path. Moreover, we study two different robot models, i) a two-wheeled model and ii) a four-wheeled model.

The primary experimental question is whether SAM can a) autonomously learn a motion skill based on a small set of prior knowledge, b) evaluate and continuously improve it, c) exhibit reasonably good time performance, and d) cover a generic application on various robot models.

4.1 Two-wheeled model

In the first experiments, a two-wheeled robot with dimensions of 2 cm^2 and a mass of 100 g is used. The action space of the wheels (m_1, m_1), which expects a torque, was selected with 0.01 N m to 0.3 N m. The action period (t_f) was set to 50 ms to 1000 ms and the *fitness* has a range from 0 to 100, directly computed from the reward. In the following context, the term *step* indicates a basic movement over time t_a, while an *episode* is a set of five steps.

For the initial fundamental question, (I)*"Am I able to move?"*, SAM succeeds in the very first step and computes the *learnability* to *TRUE*. This is not surprising since as long as the two-wheeled robot is in contact with the ground, it can initiate a movement. SAM then begins learning a basic movement by randomly exploring movement patterns and reasoning about them with prior kinematic knowledge. **Figure 11** illustrates the results of level *II*, in which all patterns (*forward*/*backward*/*left*/*right*) were successfully found in only 50 steps, taking a total of 65 s. For a model with two actuators (wheels), the search space is manageable and works relatively fast, but the performance decreases as the number of actuators increases, which we will observe with the four-wheeled model. However, this can be addressed with suitable heuristics.

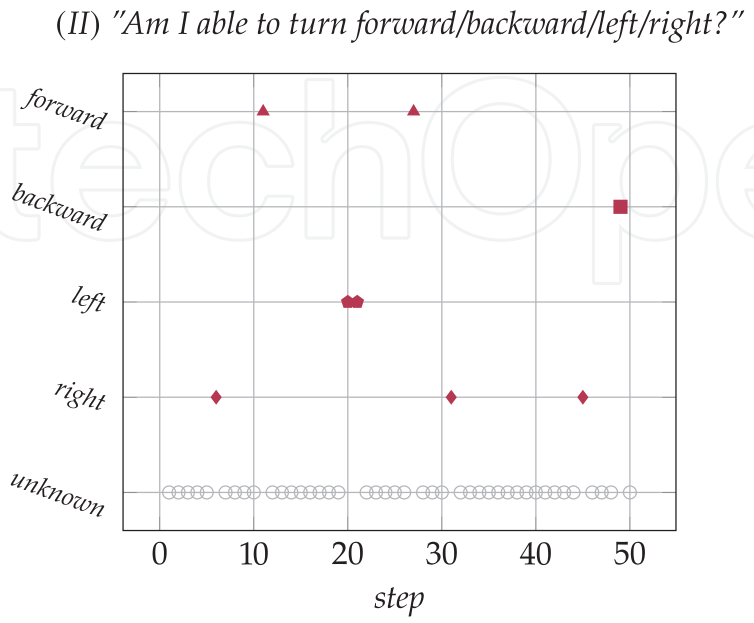

Figure 11.
Acquisition of basic motion patterns (level II - section 3.5.1). The red markers represent the respective patterns (forward = triangle, backward = square, left = pentagon, right = diamond) argued and identified in a particular step (physical interaction) with kinematic knowledge.

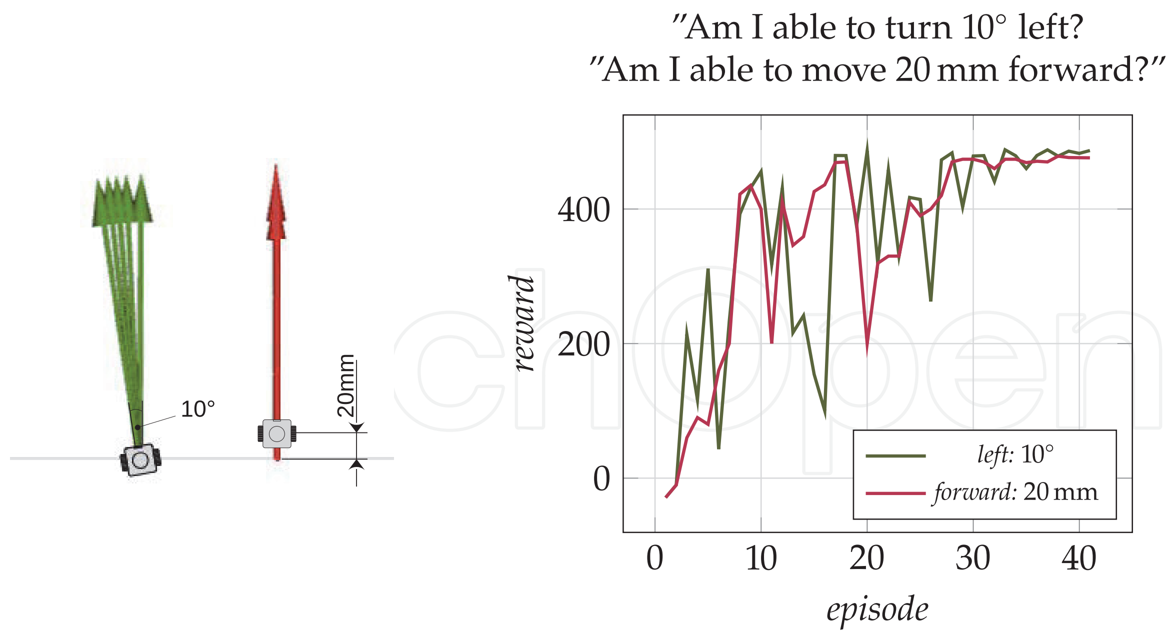

Figure 12.
Acquisition of specific motion distance and angle (level III - section 3.5.1). The left illustration shows the successful learned specific motions (green: 10 and red: 20 mm). The right image depicts the accumulated reward from the Q-table RL per episode, where an episode consists of five steps.

At the next level (III), these rules are used to learn a forward liner movement $a_{lin}\{fw\}\{20mm\}$ with a specific distance of 20 mm and a left turn angular movement $a_{ang}\{lt\}\{10°\}$ with an angle of 10°. The RL Q-Table learning is applied for each motion action, where the reward (0–100) directly represents the *fitness* of each. **Figure 12** depicts the learning performance of $a_{ang}\{lt\}\{10°\}$ (green) and $a_{lin}\{fw\}\{20mm\}$ (red). The reward settles at episode 35, with the total time of the 50 episodes averaging 4 min 30 s. The respective learned motion commands are:

$$\begin{aligned} &a_{ang}\{lt\}\{10°\}: \; m_1 = -0.15\,\text{Nm},\; m_2 = 0.15\,\text{Nm},\; t_f = 355.71\,\text{ms},\; fitness = 83,\; and. \\ &a_{lin}\{fw\}\{20\,\text{mm}\}: \; m_1 = 0.26\,\text{Nm},\; m_2 = 0.26\,\text{Nm},\; t_f = 450.71\,\text{ms},\; fitness = 80. \end{aligned} \tag{4}$$

In the first attempt, we achieve relatively good results in an early phase, after only a few minutes. This is promising for use in resource-constrained systems, as it meets the resource requirements for migration mentioned earlier. The acquired competence knowledge is further used in level IV to accomplish a more complex skill. **Figure 13** shows the execution of a complex motion(rectangular path), where the continuous improvement of the respective motion commands is investigated. The blue rectangular path shows the first attempt using the learned angular and linear motions ($a_{ang}\{lt\} : fitness = 83$ and $a_{lin}\{fw\} : fitness = 80$). Clearly visible, the fitness is not yet sufficiently developed to follow a reasonably good rectangular path. In the following, SAM tries to improve those (using PCCA) over several iterations until a sufficient fitness ($threshold = 99$) is learned. After about 60 min it has improved its capabilities and successfully navigates the red rectangular path significantly better than the green (after 25 min) one. When performing complex actions, it is also clearly visible (red path) the effects of a small movement error, which accumulates over further steps. This is due to the non-consideration of the actual respective error of action. In this work, we consciously accept this fact, but we will attempt to reduce it in a general way in further work.

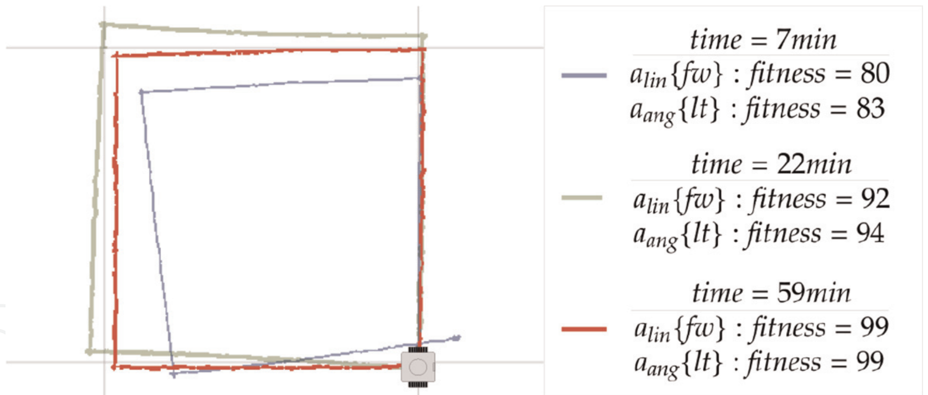

Figure 13.
The exploitation of complex motion following a rectangle path (level IV - section 3.5.1). The respective color shows the development (improvement) over time, starting with the blue path (accomplished with the commands learned from the first few attempts), followed by the green, and finally the red, representing the best movement competence.

4.2 Four-wheeled model

Further, we extend our experiments to a four-wheeled robot model as a first step to verify the general applicability of SAM. The basic assumptions and implementations of the robot model remain the same, except for two additional wheels. Due to these two further actuators, the search space increases, which leads to significant differences in the learning phase (level *I*) of the motion patterns (*forward*/*backward*/*left*/*right*), depicted in **Figure 14**. Unlike the two-wheeled model, SAM requires significantly more time, i.e., 200 steps (four-wheeled model) instead of the previous 50 steps (two-wheeled model). However, this was expected and will become even more complex with other systems, such as drones (acting in three-dimensional space). Once this phase is overcome, SAM can achieve the same good RL

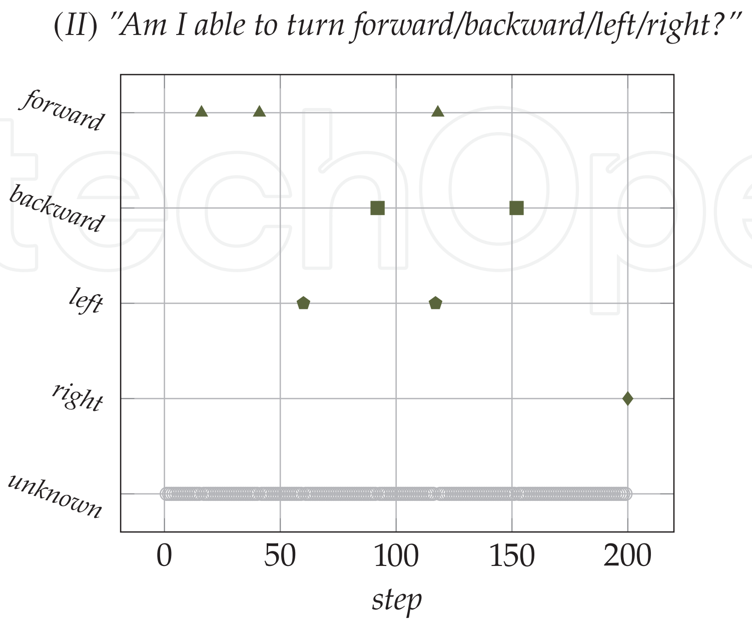

Figure 14.
Acquisition of basic motion patterns (level II - section 3.5.1) of the four-wheeled robot. The green markers represent the respective patterns (forward = triangle, backward = square, left = pentagon, right = diamond) argued and identified in a particular step (physical interaction) with kinematic knowledge. In contrast to the two-step model, SAM requires significantly more time, i.e. 200 steps instead of the previous 50 steps (two-wheeled model).

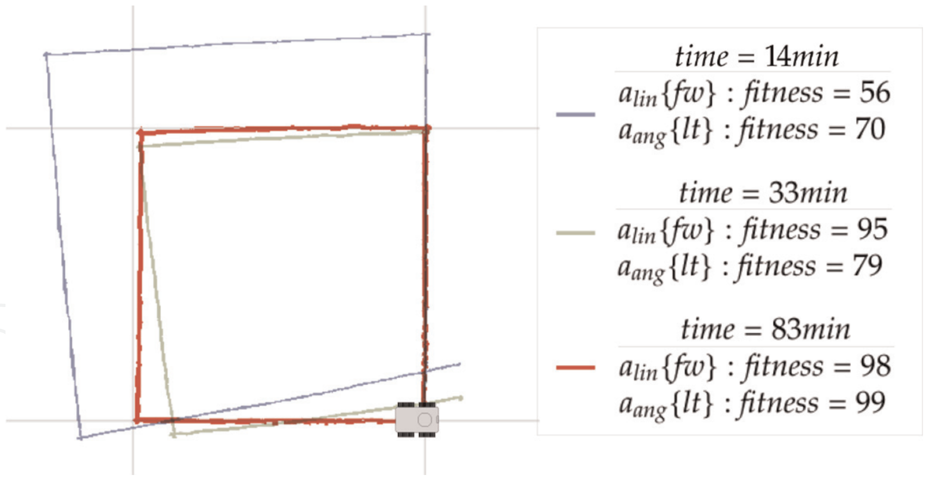

Figure 15.
Exploitation of complex motion following a rectangle path (level IV - section 3.5.1) or the four-wheeled robot. The respective color shows the development (improvement) over time, starting with the blue path (accomplished with the commands learned from the first few attempts), followed by the green, and finally the red, representing the best movement competence. No significant differences to the two-wheeled model can be identified in this skill level due to the hierarchical learning approach.

results (level *III*) with the four-wheeled as with the two-wheeled model. We did not experience any significant difference in terms of learning performance.

The reason for this is the hierarchical learning approach, where the level above is abstracted from the level below in terms of performance. This gives us confidence that SAM is well suited for generalization. The last image of our experiments shows the development of the rectangular path by the four-wheeled robot with SAM, which was successfully mastered in 50 min (see **Figure 15**).

In summary, we have demonstrated with our experiments that SAM can learn autonomously complex motion skills based on a small set of prior knowledge and can further develop them with reasonable good time performance. We showed the first step for a generic application to various robot models by demonstrating the different wheel-based models.

5. Conclusions

In this article, we introduced SAM, which starts with a set of general prior knowledge and appropriate methods to autonomously acquire and develop specific complex skills. It combines methods of KR&R with methods of learning from the physical environment and aims to be applied in resource-constrained systems. We proposed a cognitive behavior (PCCA), which enables the continuous acquisition of skills, their evaluation, and the further development and adaptation of already learned skills. To this end, we modeled generic competencies using ontologies and formulated SAS based on elementary physical quantities to build a generic interface to the physical environment. Specifically, we demonstrated SAM based on motion skills learned through a general knowledge of kinematics laws and geometry. Further, we applied hierarchical knowledge acquisition with RL to acquire basic and more complex movements. We argue that this approach is general because the only assumptions we make are the laws of kinematics and geometry, the availability of and access to sensors and actuators, and the availability of a database describing the skills to be learned. Based on this generic knowledge, we demonstrated the acquisition of basic motion and a complex movement where the robot successfully moved along a rectangular path. To prove the generic approach, we evaluated it through experiments with a two-wheeled and four-wheeled millirobot. Where the acquisition performance in terms of resources delivers promising results for further deployment of the method in resource-constrained systems.

Thus, in the first step, we have demonstrated a cognitive system that develops more complex behaviors with a set of general prior knowledge and appropriate methods to function in arbitrary environments. In this work, we still assume that the robot knows the meaning of the actuators and sensors, although these do not necessarily have to be present a priori. In the next step, we want to remove this assumption. There is promising work in automatic semantic knowledge acquisition for sensor and actuator data that could help address this problem in a meaningful manner, which we will investigate further. Moreover, we will continue to develop an even more general approach, where an exhilarating challenge in this context could be the applicability of our method in a three-dimensional system. In addition, there are still limitations to the use of KR&R methods in resource-constrained systems, which we discussed in this work. Another medium-term goal is to study SAM in resource-constrained systems. Therefore, we will specifically address the transition to a real resource-constrained system in the form of a millirobot. In summary, our first results indicate that the use of SAM has an advantage for generic applicability, and we will continue to try to advance this approach.

Abbreviations

RL	Reinforcement Learning
SAM	Skill Acquisition Method
PCCA	Playful Continuous Competence Acquisition
SAS	Sensor and Actuator Space
KR&R	Knowledge Representation and Reasoning
SOMA	Socio-physical Model of Activities
IMU	Inertial Measurement Unit

Author details

Markus D. Kobelrausch*† and Axel Jantsch†
Technical University of Vienna, Vienna, Austria

*Address all correspondence to: markus.kobelrausch@tuwien.ac.at

† These authors contributed equally.

References

[1] Tenorth M, Beetz M. KNOWROB — knowledge processing for autonomous personal robots. In: 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems. 2009. pp. 4261-4266. DOI: 10.1109/IROS.2009.5354602

[2] Nilsson NJ. Shakey the Robot. Technical Report 323, AI Center, SRI International. CA, USA: Menlo Park; 1984

[3] Tenorth M, Beetz M. KnowRob: A knowledge processing infrastructure for cognition-enabled robots. The International Journal of Robotics Research. 2013;**32**(5):566-590

[4] Tenorth M, Beetz M. Representations for robot knowledge in the KnowRob framework. Artificial Intelligence. 2017; **247**:151-169

[5] Beetz M, Tenorth M, Winkler J. Open-EASE. In: 2015 IEEE International Conference on Robotics and Automation (ICRA). 2015. pp. 1983-1990. DOI: 10.1109/ICRA.2015.7139458

[6] Beetz M, Beßler D, Haidu A, Pomarlan M, Bozcuoğlu AK, Bartels G. Know Rob 2.0 — A 2nd Generation Knowledge Processing Framework for Cognition-Enabled Robotic Agents. In: 2018 IEEE International Conference on Robotics and Automation (ICRA). 2018. pp. 512-519. DOI: 10.1109/ICRA.2018.8460964

[7] Beßler D, Porzel R, Pomarlan M, Vyas A, Höffner S, Beetz M, et al. Foundations of the socio-physical model of activities (SOMA) for autonomous robotic agents. Frontiers in Artificial Intelligence and Applications. Volume 344. arXiv preprint arXiv: 2011.11972. 202

[8] Riazuelo L, Tenorth M, et al. RoboEarth semantic mapping: A cloud enabled knowledge-based approach. IEEE Transactions on Automation Science and Engineering. 2015;**12**(2):432-443

[9] Wyatt JL, Aydemir A, et al. Self-understanding and self-extension: A systems and representational approach. IEEE Transactions on Autonomous Mental Development. 2010;**2**(4): 282-303

[10] Sheth A, Henson C, Sahoo SS. Semantic sensor web. IEEE Internet Computing. 2008;**12**(4):78-83

[11] Ganz F, Barnaghi P, Carrez F. Automated semantic knowledge acquisition from sensor data. IEEE Systems Journal. 2016;**10**(3):1214-1225 Conference Name: IEEE Systems Journal

[12] Moore T, Stouch D. A generalized extended Kalman filter implementation for the robot operating system. In: Menegatti E, Michael N, Berns K, Yamaguchi H, editors. Intelligent Autonomous Systems 13. Vol. 302. Cham: Springer International Publishing; 2016. pp. 335-348

[13] Alatise MB, Hancke GP. Pose estimation of a Mobile robot based on fusion of IMU data and vision data using an extended Kalman filter. Sensors. 2017; **17**(10):2164

[14] Durrant-Whyte H, Bailey T. Simultaneous localization and mapping: Part I. IEEE Robotics Automation Magazine. 2006;**13**(2):99-110 Conference Name: IEEE Robotics Automation Magazine

[15] Fierro R, Lewis FL. Control of a nonholomic mobile robot: Backstepping

kinematics into dynamics. Journal of Robotic Systems. 1997;**14**(3):149-163

[16] Morin P, Samson C. Motion control of wheeled Mobile robots. In: Siciliano B, Khatib O, editors. Springer Handbook of Robotics. Berlin, Heidelberg: Springer; 2008. pp. 799-826

[17] Martins FN, Celeste WC, Carelli R, Sarcinelli-Filho M, Bastos-Filho TF. An adaptive dynamic controller for autonomous mobile robot trajectory tracking. Control Engineering Practice. 2008;**16**(11):1354-1363

[18] Das T, Kar IN. Design and implementation of an adaptive fuzzy logic-based controller for wheeled mobile robots. IEEE Transactions on Control Systems Technology. 2006; **14**(3):501-510 Conference Name: IEEE Transactions on Control Systems Technology

[19] Antonini P, Ippoliti G, Longhi S. Learning control of mobile robots using a multiprocessor system. Control Engineering Practice. 2006;**14**(11): 1279-1295

[20] Christopher JCH. Watkins and Peter Dayan. Q-learning. Machine Learning. 1992;**8**(3):279-292