universität
wien

MASTERARBEIT

Titel der Masterarbeit

„Goal-driven developmental learning on a mobile robot“

Verfasser

Christian Papauschek, BSc

angestrebter akademischer Grad

Master of Science (MSc)

Wien, 2012

Studienkennzahl lt. Studienblatt:	A 066 013
Studienrichtung lt. Studienblatt:	Joint Degree Programme MEi:CogSci Cognitive Science
Betreuer:	Dipl.-Ing. Dr.techn. Michael Zillich

Kurzfassung

Seit den frühen 1980er Jahren wurden mobile Roboter entwickelt, die Objekten ausweichen und durch verschiedenste Umgebungen navigieren können. Hierfür wurde Vorwissen über das Zusammenspiel zwischen Sensoren und Motoren des Roboters verwendet. Als man komplexere Anforderungen an Robotik-Systeme stellte, scheiterten traditionelle Architekturen daran, diesen gerecht zu werden. Einen interessanten Ansatz für dieses Problem bietet der Forschungsbereich *Developmental Robotics*, der größtenteils von der kognitionswissenschaftlichen Community beeinflusst wurde und von den gemeinsamen Interessen der Entwicklungspsychologie und der Robotik profitiert. Der Forschungsbereich beschäftigt sich unter anderem damit, wie ein autonomer Roboter lernen kann komplexe Aufgaben zu bewältigen indem er seine Umgebung erforscht. Das Komplexitätsproblem der Robotik soll mit diesem erfahrungsbasierten Ansatz gelöst werden.

In dieser Arbeit wird eine Lernarchitektur auf einem mobilen Roboter implementiert, welche auf den Kernprinzipien der *Developmental Robotics* basiert. Herangezogen werden die Prinzipien der *Verifikation*, des *Embodiment*, der *Subjektivität*, des *Grounding* und der *inkrementellen Entwicklung*. Die Architektur ermöglicht die autonome Konstruktion eines sensomotorischen Modells, ohne Vorwissen über die Sensorenanordnung. Der Roboter verwendet die Vorhersagen dieses Modells um bestimmte Sensorenwerte zu erreichen. Dies resultiert in zielgerichtetem Verhalten.

Die Ergebnisse zeigen, dass die Architektur es dem Roboter ermöglicht, die Sensorenwerte als Folgen seines Handelns vorherzusagen. Der Roboter war in der Lage, mit Hilfe seiner Ultraschall-Abstandssensoren bestimmte Entfernungen zu Objekten zu halten, ohne spezielle Vorprogrammierung. Bei der Auswertung wird gezeigt, dass der Roboter nach und nach seine Performance verbessert und ein genaueres sensomotorisches Modell konstruiert, sobald mehr Daten über die Motoren und die Umgebung gesammelt werden können. Wir schlussfolgern, dass erfahrungsbasiertes Lernen ein realisierbarer Ansatz in der Robotik ist, und es ist unsere Überzeugung, dass Forschung aus der Entwicklungspsychologie einen wichtigen Einfluss in diesem Gebiet darstellen wird.

Abstract

Since the early 1980s, mobile robots have been programmed to avoid objects and navigate environments. Prior knowledge about the robot's sensors and actuators was used to solve this problem. But as robotics systems were expected to address more complex tasks, traditional robot architectures failed to scale up to them.

One interesting approach to this problem can be seen in the field of developmental robotics, which was influenced in large parts by the cognitive science community and benefits from the mutual interests of developmental psychology and robotics. The main research question in this field is how an embodied agent can learn complex tasks by exploring its environment. This experience-based learning approach aims to solve the scaling problem of robotics.

In this thesis, a developmental learning architecture is implemented on a mobile robot. This architecture follows the core principles of developmental robotics: the verification principle, the principles of embodiment, subjectivity and grounding, as well as the principle of incremental development. It allows the robot to autonomously construct a sensomotoric model, without prior knowledge about its sensor configuration. This model can then be used to predict the outcome of the robot's actions. In the proposed architecture, the agent uses these predictions to reach specific sensor states, resulting in goal-driven behavior.

The results show that the architecture enables the robot to predict the consequences of its actions on its sensor states, and it was evaluated in several experiments. The robot was able to keep specific distances to objects using its ultrasonic distance sensors, without being preprogrammed with an algorithm that describes the necessary actions.

During the evaluation, we also show that the robot gradually improves its performance and constructs a more accurate sensomotoric model, as it collects more data about its motors and its environment. We conclude that experience-based learning is a feasible approach in robotics, and it is our belief that research from developmental psychology will become an important influence in this area.

Acknowledgments

I wish to express my gratitude to my supervisor Dipl.-Ing. Dr.techn. Michael Zillich, who offered invaluable assistance and support also with regard to the technical challenges of maintaining and repairing a mobile robot.

Special thanks also to the coordinators of the *MEi:CogSci master program*, which provided me the opportunity to pick a research project in the field of robotics. Also, I would like to convey thanks to the *University of Vienna* and the *University of Technology Vienna* for providing the laboratory facilities.

I wish to express my gratitude to my family, for their understanding and support, through the duration of my studies.

Table of Contents

1 Introduction

In the past few decades, there have been countless advances in the field of robotics. Among them are new approaches and improvements to systems which try to capture data and learn from it in a meaningful way during the deployment of the robot. In some cases, these systems are biologically inspired or claim to apply some mechanisms from nature in a robotics problem.

1.1 Assumptions of learning systems

Tasks like *object grasping*, that seem to be very easy from a human perspective, are still relatively hard problems in robotics. There are many ongoing projects that deal with the grasping problem alone. The general approach is to build a mathematical model of grasping, sometimes using machine learning techniques, based on a set of successful behaviors. The "training" set of these successful behaviors can be created manually or via an automated process, or a combination of both. Using the output of this model, we can then control the arm hoping that it will grasp in the right way, or from the right direction. If it fails however, we have to go back and improve our model. Since the robot learned from outside instructions, the robot cannot actually perceive its own failure. It follows that it cannot improve upon the model on its own.

Even though most projects that deal with object grasping use some kind of learning algorithm in order to train the software in performing the task better, robotics is still struggling with this seemingly simple task.

In this thesis we are going to argue that we need to take a closer look at the assumptions of these learning systems. In every implementation there are assumptions which may differ substantially from one system to the other. It is our belief that we should take a closer look at nature when it comes to designing learning systems. For a robotics system

that needs to learn how to move, it makes sense to ask oneself how a developing human child learns how to move.

The field of developmental psychology deals with this kind of learning, but it is nowhere near offering a complete algorithm that can be implemented in software, ready for use in robotics. There are however some basic ideas about what a child in its early development phase might try to learn. One thing that may be very important is the idea that the brain constantly produces expectations and predictions about how current movements will influence the environment and consequently sensor experience.

1.2 Goals

Motivated by this idea, this work will show an implementation of a learning system that is not motivated by outside goals, but learns mostly about its body. Even though the system is not oriented towards given behaviors, it will use these predictions to fulfill certain goals. This will show that a system which is able to predict some of the outcomes of its actions can more easily perform new tasks which it had not performed before.

A mobile robot will be programmed to learn autonomously by exploring its environment and collecting data about it. A learning system will be implemented, that will continuously construct a model of the interactions between the robots actuators and the environment. After learning from the observations, this experience-based model will be evaluated in three different situations. In each situation, the robot will have to reach a certain target sensor state. by using its predictions to reach certain sensor states.

1.3 Overview

This work deals with the development of a learning system for a mobile robot, that will enable it to learn autonomously about its interactions with the world, and use this

knowledge to reach specific goal states. The document is structured in sections that will deal with different aspects of this work:

- *Chapter 2: Background information* introduces the young field of development robotics and its basic research assumptions, as well as a quick overview of reinforcement learning.
- *Chapter 3: Related work* discusses related research in the areas of developmental robotics and reinforcement learning, as well as the mobile robot that will be used for the experiments.
- *Chapter 4: Hypothesis* proposes a reinforcement learning architecture that follows the principles of developmental robotics, and will be implemented on the mobile robot.
- *Chapter 5: Implementation* outlines the implementation of the proposed learning architecture on the mobile robot *Corvid*, including hardware and software descriptions.
- *Chapter 6: Experimental setup* shows the experimental setup that was chosen for the robot experiments.
- *Chapter 7: Experimental results* summarizes the results that have been obtained from the experiments with the mobile robot.
- *Chapter 8: Discussion* discusses the previously obtained experimental results with regard to the behavior of the robot and its prediction performance, as well as their relation to the principles of developmental robotics.
- *Chapter 9: Conclusion* summarizes the conclusions of this work and their application to future research.

2 Background information

Autonomous robots are often controlled by algorithms which are specifically tailored to the sensory information they receive, and to the goals that the robot should be able to perform in the end. This approach is widely used and has its advantages, if the robots goals are simple enough or do not change fundamentally. Also, some solutions include machine learning methods which allow the robot to learn while it is deployed in its environment [SMS09].

2.1 Learning in robotics

A lot of research in robotics involves some kind of learning, but there are many different approaches. We can describe the world in some abstract mathematical way, and then let a robot use this model to predict the outcome of different actions. The robot can then choose the best option and act accordingly.

For example: Robots that use SLAM techniques (Simultaneous Localization and Mapping) use their laser distance sensors to build a 3D-model of the world as they move around [LCJ10]. While constantly updating this world model, the robot can calculate whether moving forward is a good idea, depending on whether there is an obstacle in front of it or not. The world model can also be used for path planning, in order to navigate longer distances efficiently. While SLAM allows the robot to autonomously learn about its environment in real-time, the way the world model is derived from the sensor readings is fixed and usually not verifiable by the robot.

This way of programming a robot is very practical and has led to many successes in robotics, as it allows engineers to use their existing knowledge about the world. It is a top-down approach to learning, and it does not try to model the way humans acquire knowledge about the world.

However, it seems that these traditional approaches are not able to scale up to tasks that require a higher level of intelligence. This includes seemingly simple motor tasks, such as walking and catching objects, that children are able to quickly learn and master. It is simply not feasible to preprogram all the domain knowledge that a robot needs, before it is deployed in its environment. As a reaction to these difficulties, a new approach of dealing with learning tasks in robotics emerged.

2.2 Robotics and developmental psychology

In the past decade, the fields of robotics and developmental psychology discovered their mutual interest in the big question of how embodied systems are able to learn autonomously. Developmental psychologists are looking at the development of children, and develop theories how babies are eventually able to understand their body and use it as they grow older. The field of robotics would benefit greatly from a theory that could be used in a system that allows robots to learn like children do. Hence, a new field emerged: developmental robotics, also called epigenetic robotics.

The basic research assumption behind developmental robotics, which was influenced in large parts by the cognitive science community, is that

> "true intelligence in natural and (possibly) artificial systems presupposes three crucial properties:
>
> 1. the *embodiment* of the system;
> 2. its *situatedness* in a physical and social environment;
> 3. a prolonged *epigenetic developmental process* through which increasingly more complex cognitive structures emerge in the system as a result of interactions with the physical and social environment."
>
> [ZB01, emphasis in original]

Even though developmental robotics is a very young field, some basic principles have been proposed. Following from the research assumptions, these five principles are: [Sto09]

1. The *verification principle*, which serves as the basis for all the following principles, and says that an agent "can create and maintain knowledge only to the extent that it can verify that knowledge itself." [Sto09]

2. The *principle of embodiment* says that an agent has to have the ability to affect its environment, and stands in contrast to early AI systems in which data processing occurred without any interaction with an environment. This follows from the verification principle, as a body is needed for the actions that allow verification of what the agent learned.

3. The *principle of subjectivity* also follows from the verification principle and deals with the question of learning: agents should only learn from their own experiences, their own history of interactions with the environment. One interesting aspect that follows from this principle is that robots with different bodies will learn different object affordances, depending on how each robot can interact with the object.

4. The *principle of grounding* is a necessity of the verification principle in that all verifications must have some basis, that does not have to be further examined. It is proposed that grounding consists of the reproducible observation of action-outcome pairs.

5. The *principle of incremental development* follows from the time-constraints of subjective learning, and says that learning can only occur gradually, as an agent accumulates more and more experience in dealing with the environment. This relates to many theories in developmental psychology, as it is often stated that development of a child proceeds in stages, or milestones.

The importance of the verification principle cannot be overstated, and developmental robotics makes this assumption explicit. This does not mean, however, that the principle

is ignored in other robotics research. In some cases, monitoring systems are implemented that constantly observe the robots own performance. It can be argued that these systems allow the robot to verify its own actions.

Many self-regulatory systems may implement the verification principle, without explicitly considering it. Artificial robotic immune systems are an example of this, which are currently being developed for use in the *RoboCup* robots [SK11]. The proposal of a *Robotic Immune System* (RIS) is a reaction to the increase in complexity of robotic systems, and aims to improve the fault tolerance of these systems by detecting anomalies:

> "Our work meets this challenge by developing a mechanism for robotic systems that is capable of detecting defects, selecting feasible counter measures, and hence keeping robots in a sane and and consequently safe state." [SK11]

While not strictly in the realm of developmental robotics, this research has many parallels and some of the same motivations behind it. One of these parallels are self-inspecting systems: "Introspection is an inevitable feature for any immune system, biological as much as artificial." [SK11]

2.3 Reinforcement learning

Reinforcement learning is a model that fits particularly well to the requirements of developmental robotics, including the five principles mentioned before. It has been used in this field of research before, and an example of this will be given in Chapter 3.

The reason is that the reinforcement learning model generally assumes an embodied agent which learns from its environment gradually, via systematic trial and error. It is a supervised learning technique in which only a reward signal has to be provided to the agent [KLM96]. The reward signal tells the robot how well it is currently doing. In contrast to

other supervised learning mechanisms, the robot is not told which action is the best in each situation. A variation of reinforcement learning will also be used in this thesis.

The proposed software architecture uses these five principles as a basis, as it allows a robot to learn about the influences of its actuators on the environment by experience. The learning progress is expected to be incremental, and is measured by using the robots self-acquired knowledge to reach different sensor states. It will be explained in more detail in Chapter 4.

3 Related work

As mentioned in the previous section, the reinforcement learning model fits particularly well to the principles of developmental robotics. This chapter will discuss the related research in this area and the reinforcement learning models used.

3.1 Previous work in developmental robotics

One important influence on this thesis was the work by Odeyer et al., on what is called an "Intrinsic Motivation System" [OK04]. The phenomenon that they addressed in their research is learning by self-motivated actions. It seems that children learn a great deal about the world by exploratory activities and are intrinsically rewarded by engaging in these activities. Furthermore, in developmental psychology, a child's learning progress is often described as going through several developmental stages.

In their work, they present a developmental algorithm called "Intelligent adaptive curiosity" (IAC) [OK04]. Using this algorithm, a robot learns to predict the position of a toy using reinforcement learning. In this publication, the experiments were conducted in a simulated 3D-environment. Unlike other reinforcement learning systems however, this system rewards itself by trying to focus on "learnable" tasks, therefore being intrinsically rewarded. What is learnable is measured by keeping track of the learning progress of the predictor, constantly measuring its prediction accuracy. The system then tries to split the sensomotoric space into so-called regions which are explored separately by the agent. The idea is that the robot will focus on learning affordances that can actually be learned, and tries to ignore tasks that appear to be unpredictable at the moment.

From a reinforcement learning perspective, the reward is not given by the experimenter, but is intrinsically given by the "progress of learning". In other words, the system rewards itself for being in situations where it is learning the most.

The algorithm has been successfully applied not only to simulated robots but also to real devices [OKH07] [BO09]. These experiments did not involve mobile robots, but an implementation of the IAC algorithm on the *Corvid* mobile robot, which will also be used for the experiments in this thesis, has been done [Län10] (see Section 3.5).

Of particular interest in the IAC experiments is the way the system was evaluated. The focus here was to reproduce something similar to the learning stages that occur in children. The authors argue, that this has been achieved [OK04].

3.2 Active learning

Another important related project deals with active learning [SMS09], in which a humanoid robot learns to use its arm in order to reach objects. The robot used motor babbling (random movements) in order to explore its environment and collect data about it. Using this data, the authors trained a system that predicts the robot's state in the next time step. In addition, the system reflected on its own prediction and measured its own prediction confidence. The confidence data was then chosen in order to guide further exploration. The authors used a learning strategy that distinguished between learning and exploration phases. A similar system is also used in this thesis:

> "A robot explores the environment to collect learning data, and evaluates sensorimotor functions on-line. After exploration, the robot optimizes sensorimotor functions with the collected learning samples off-line. These two processes are repeated alternatingly until the desired performance is reached." [SMS09]

There is an interesting parallel between this active learning approach and the IAC (Intelligent Adaptive Curiosity) algorithm mentioned in the previous section: both use some kind of measurement of confidence in order to direct the robot to explore more interesting environments.

3.3 Other related work

Many research projects follow some of the principles of developmental robotics, even if they do not explicitly state it. One of these projects has already been mentioned in Section 2.2 [SK11], but many other works implement some type of self-monitoring or self-regulating systems that follow part of the verification principle.

Another example of this is a project in which machine learning methods are used to acquire visual 3D object models [ZPMV11]. The authors of this work propose a system that allows online learning of these object models, and implemented a way in which the system is able to assess its own object detection performance. They used "three learned probabilistic measures for *observed detection success*, *predicted detection success* and *model completeness*" [ZPMV11]. This self-assessment of performance is a way of verifying the knowledge that the system already gained. In this case however, the system was not embodied and therefore not able to gain the knowledge in a subjective matter. It does not follow the principles of developmental robotics outlined in Section 2.2, but the authors plan to develop a complete system as a next step of the project.

3.4 Mobile robot platform

In this thesis, a mobile robot is used in order to implement the proposed learning architecture. The platform we used is the recently developed Corvid [ZBKL10]. The robot was previously used in developmental robotics experiments (see Section 3.5).

Corvid is a robot capable of traversing rough terrain, high resolution vision and manipulation of objects via arm and grasper (see Figure 1). In addition to its camera, the robot is capable of avoiding obstacles through eight distance sensors. In some previous experiments, the robot also used a compass sensor for navigation. To facilitate quicker development and testing of the implementation, a virtual 3D model of the robot and its environment has been previously created using the simulation platform Player/Gazebo.

This also allows basic testing of controlling software in a virtual environment before making experiments on the real device.

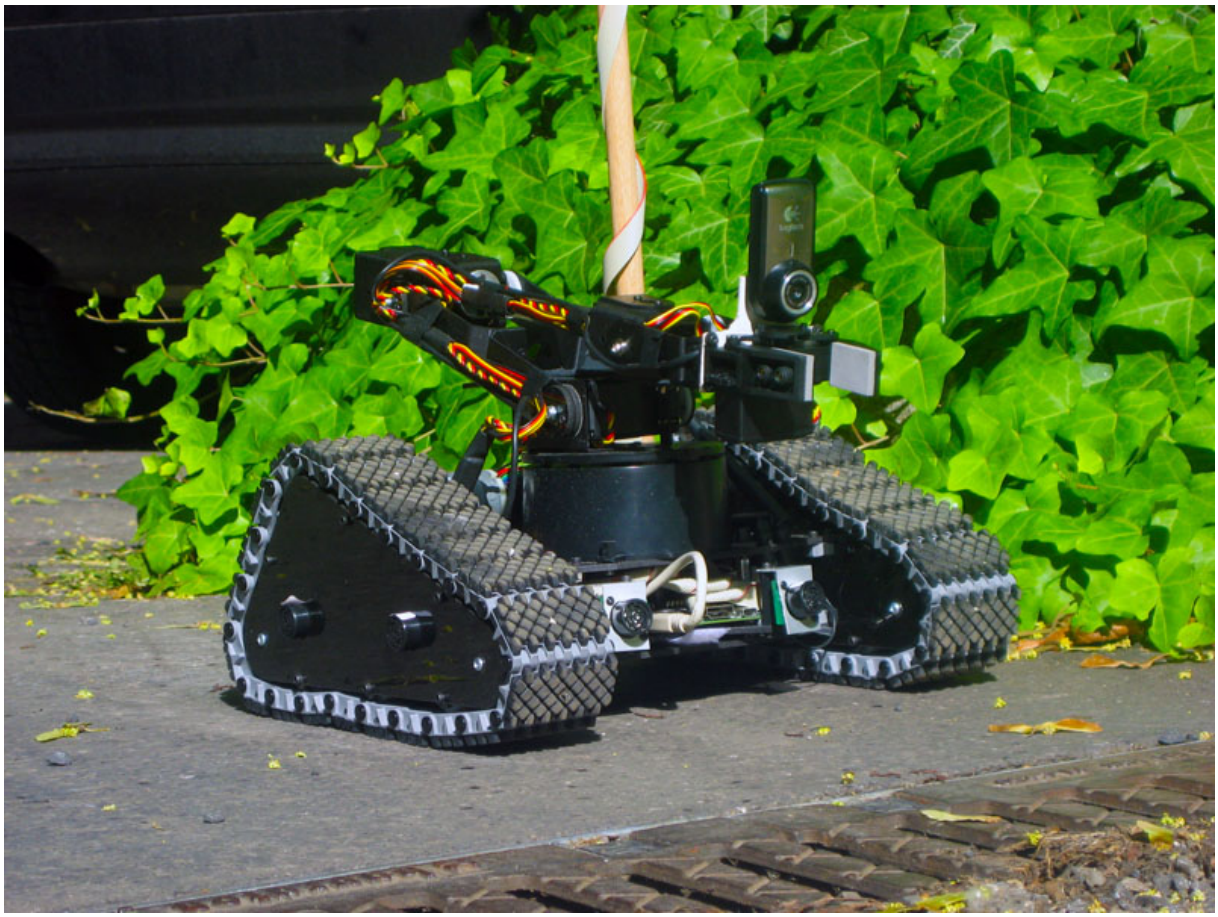

Figure 1: *The Corvid mobile robot.* It is capable of traversing rough terrain, high resolution vision and manipulation of objects via arm and grasper [ZBKL10]. In addition to its camera, the robot can measure distances in all directions by using its eight ultrasound distance sensors.

3.5 Developmental robotics and Corvid

The robot platform *Corvid* has been previously used in experiments involving the *Intelligent Adaptive Curiosity* algorithm [OK04] mentioned in Section 3.1. The question of this work was whether a mobile robot could learn affordances using this architecture.

During the experiments [Län10], the robot had to learn which movements can be made in a restricted environment much like the one used in this thesis. The conclusion was that some affordances could be learned, but they do not include longer behaviors such as object avoidance or wall following.

However, because the agent does not try to reach specific goals in the *Intrinsic Motivation System*, the evaluation of such a system is difficult. Part of the evaluation was done by measuring how well the system could predict its environment, and find learnable movements, or affordances.

Here, we will evaluate not only how well the system predicts its environment, but mainly also if and how well it can perform certain tasks using the learned model. The authors of the IAC focused on splitting the learning tasks into regions that can be learned separately, also in order to avoid trying to learn unpredictable events. In this thesis, the focus lies on the evaluation of such autonomously learned predictive models. We therefore use a number of goals that the robot will have to reach, in order to allow a quantitative evaluation.

4 Hypothesis

One central idea of developmental robotics is that agents need to learn incrementally and from their own interaction with the environment, in order to become more robust against unexpected properties of the sensors and the environment.

This chapter will describe a reinforcement learning architecture that should enable an agent to learn about its sensors and motors autonomously. The robot will start with exploring its motors and sensors, and therefore its environment. While exploring, the proposed system collects sensomotoric data that will be used to build a predictive forward model of its actuators. After collecting enough data, it is expected that the agent can perform some simple tasks using this autonomously learned forward model.

In order to evaluate this model, the architecture will be implemented in the mobile robot *Corvid*.

4.1 Exploration

Each agent can be thought of having a vector of sensor readings $S(t)$ and a number of motor outputs $M(t)$ at each time step t. For example, the mobile robot *Corvid* has 8 real-valued distance sensor readings from its ultrasonic sensors, and 2 real-valued motor outputs for going forward and turning, respectively, at each time step t (see Figure 2).

In the beginning, the agent may use its motors randomly in order to collect some basic data about the relationship between its actuators and sensors. This data can then be used to build a first predictive model, as explained in the next section.

4.2 Prediction of sensor readings

A simple forward model tries to predict future sensors $S(t+1)$ while using current sensor values $S(t)$ and motor values $M(t)$ as a basis (Figure 3).

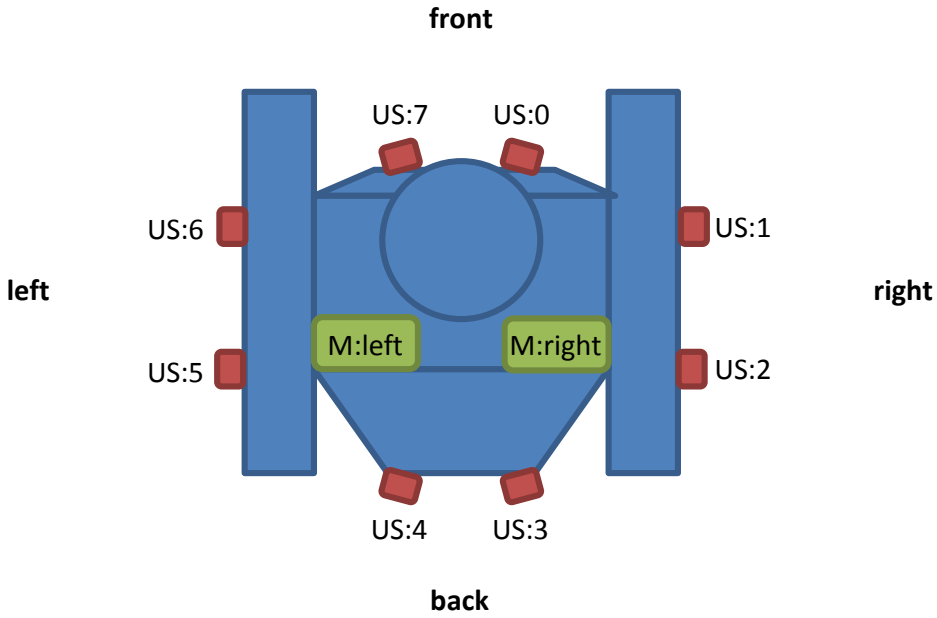

Figure 2: Illustration showing the arrangement of the 8 ultra-sound sensors of Corvid, as well as left and right motors.

This forward model can be learned by any machine learning algorithm that allows training by example, including neural networks and many other techniques. This section will focus on a possible implementation using feed-forward neural networks or similar algorithms. These models do not have memory and therefore cannot take into account past sensor readings. Some alternatives have been proposed, including back-propagation through time [Moz95] and more elaborate systems such as EVOLINO [SWGG07].

When using predictors that do not remember past inputs, historical data can be provided to the model as additional inputs (Figure 4). Of course, depending on the implementation, dealing with many additional inputs can be a practical problem. For neural networks, the computational complexity of deriving a model rises at least with polynomial time as the number of inputs increases. Nevertheless, providing the network with a limited amount of past inputs is still an option.

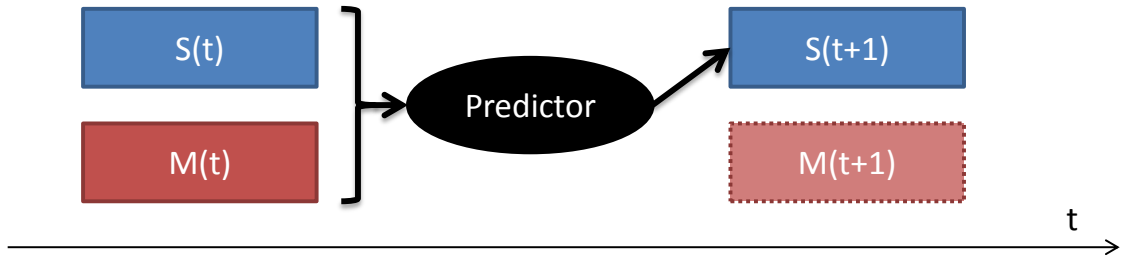

Figure 3: A simple forward model which predicts future sensor data $S(t+1)$ from current sensomotoric values $S(t)$ and $M(t)$.

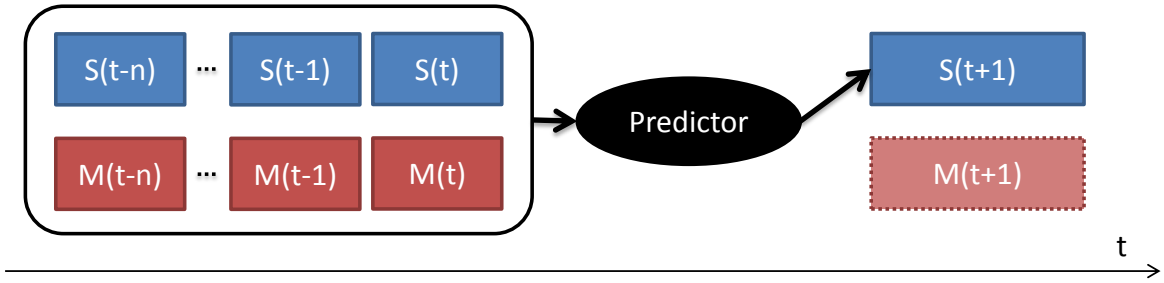

Figure 4: A forward model which predicts future sensor data $S(t+1)$ and takes into account previous sensomotoric values from $n+1$ time steps.

After training a predictor using the collected sensomotoric data, it can be used to select actions with specific outcomes, as the next section explains.

This way of building a sensomotoric model also satisfies the principle of subjectivity (see Section 2.2) in developmental robotics: no data from outside the subjective experience of the robot is used in creating the predictor.

4.3 Generation of possible outcomes

It is proposed that the autonomously learned sensomotoric model can be used to allow the agent to reach specific goals, as long as these are easily expressible through sensomotoric states. One way this can be done is by predicting the outcome of different motor actions,

by feeding different generated $M(t+1)$ values into the predictor.

If the predictions were perfect, it would be possible to predict sensor states far into the future by chaining predictions: predicted sensor states $\bar{S}(t+1)$ can again be used as current inputs for the predictor, and a prediction of $\bar{S}(t+2)$ can be made. In practice however, it is expected that predictions will degrade too much in their quality as to make useful predictions far into the future.

By generating possible motor actions of the agent, it is possible to generate a tree of possible future sensor states that it can reach.

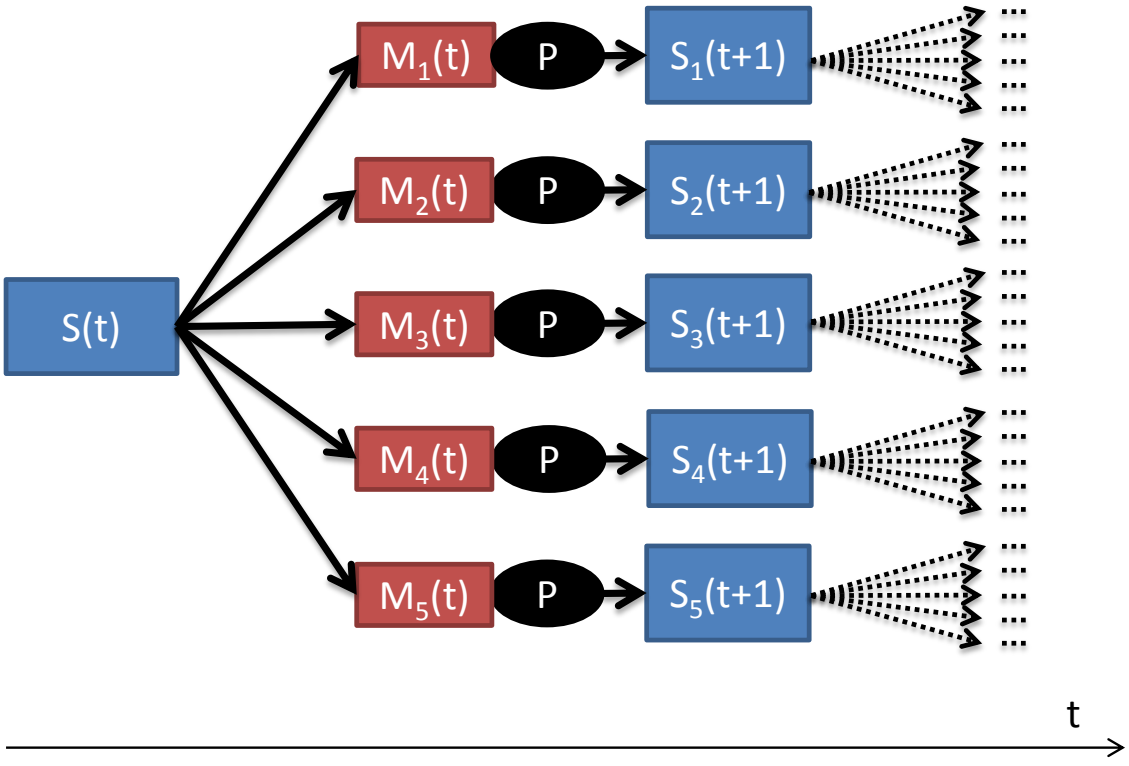

Figure 5: Tree of possible future states showing predicted sensor states for five different actions at each time t, resulting in 5 possible outcomes $S_1(t+1)$ to $S_5(t+1)$ at depth 1.

In Figure 5, the robot is able to choose between 5 different actions at each time t, resulting in 5 possible outcomes. When using the predictor at $t+1$ again, it is possible to predict

all 25 possible outcomes at $t + 2$.

In order to make use of the predicted possible outcomes, a reward signal is introduced by the experimenter, in order to tell the agent how well it is doing with a given task. With this addition, the architecture becomes a type of reinforcement learning system.

4.4 Reinforcement learning

By introducing a real-valued reward signal r_t at each time-step t we enable the robot to evaluate each sensomotoric state $SM(t)$, which is a combination of the sensor state $S(t)$ and the motor state $M(t)$. This reward depends on the goal state that we want the agent to reach. The agent can then predict the reward from each possible future state $\overline{SM}(t + x)$. The final architecture can be seen in Figure 6.

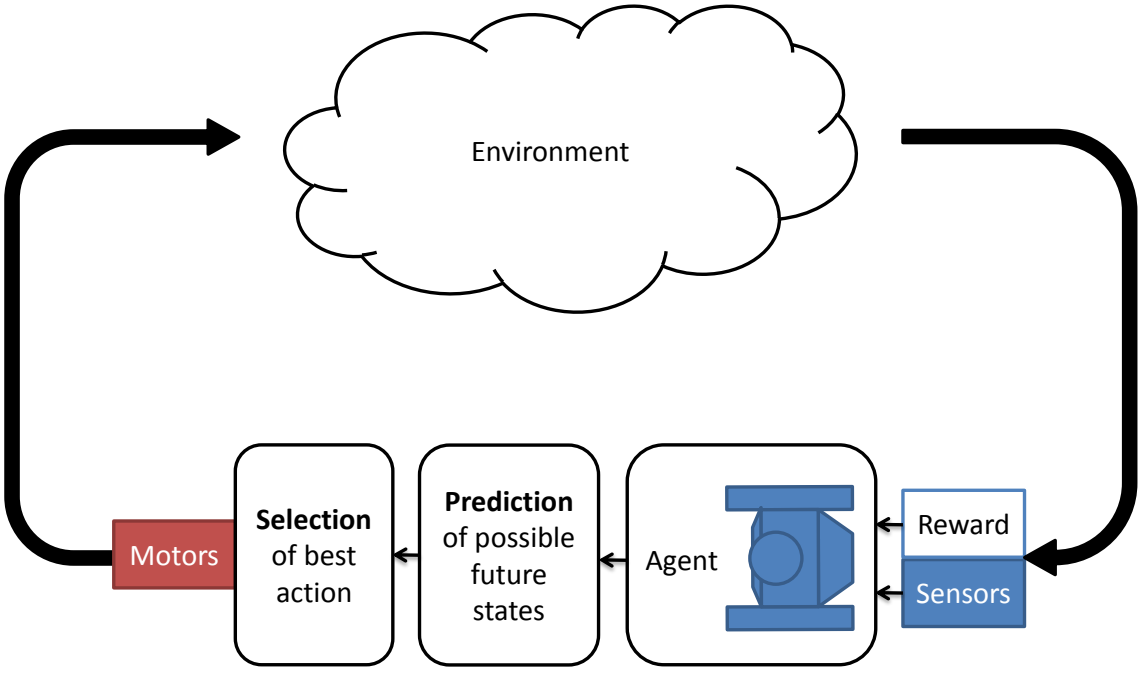

Figure 6: Final architecture showing the interaction between the agent and its environment, typical for a reinforcement learning setup.

On the basis of its current and past sensor readings, the agent generates possible future states and evaluates them based on the current goal. At depth 2 (when predicting states until $t + 2$) we get two predicted rewards r_{t+1} and r_{t+2} for each predicted outcome. We propose to select the action with the highest predicted sum of these rewards $\overline{r}_{total}$, weighting future rewards higher than immediate rewards, as formulated in Equation 1.

$$\overline{r}_{total} = 1r_{t+1} + 2r_{t+2} + \cdots + nr_{t+n} \tag{1}$$

Weighting the rewards by depth should ensure that the robot considers actions that have high reward in future, even if they result in lower immediate reward.

After choosing the action, the agent influences the environment and receives new sensor data and reward. This reinforcement learning architecture allows the sensomotoric prediction model to be trained independently from the reward signal. That means it should be possible to use the agents experience with its actuators and sensors to be used with novel goals, as will be tested in our implementation.

4.5 Evaluation

In order to test how well this architecture can be used for the purposes of goal-oriented behavior, the robot will have to reach three different goal states (or maximize three different reward signals) using the same sensomotoric prediction model. It is expected that the robot will not be able to reach the goal states very well in the beginning, since there is no data to base the predictions on. After collecting more and more data, it is expected that the robot will improve its ability to perform the given tasks.

The next sections will explain more details about the mobile robot that will be used, and the specific implementation of the learning architecture.

5 Implementation

The previous sections described a general autonomous learning architecture. To evaluate this architecture, it was implemented on the mobile robot *Corvid* which was introduced in Section 3.4. Now follows a more detailed description of the hardware and software implementation.

5.1 Hardware description

The mobile robot *Corvid* was previously developed at ACIN (Automation and Control Institute) at the *University of Technology Vienna* [ZBKL10]. The name is derived from the configuration of the arm, grasper and camera. The camera is mounted on the arm directly above the grasper, resulting in a "eye in hand" setup much like the fixed relationship between the beak and the eyes of crows (*corvids*).

The robot lends itself to student research with reinforcement learning because of its inexpensive hardware and maintenance costs. The basis of the platform is the commercially available *Lynxmotion Tri-Track Chassis*. The tracks of the vehicle are moved by two motors, resulting in a differential drive system. There is a manipulator directly mounted on this basis, which carries the camera and grasper. For the purposes of our experiments, the arm was retracted and disabled in order to prevent damage to the more fragile parts.

The most important hardware for this project are the ultrasonic sensors. These *Devantech SRF02* sensors can measure distances from 0.15 to 6 meters by sending out inaudible sound waves and measuring the time until they are reflected back from a surface. Eight of these sensors are mounted on the robot, sensing in different directions (see Figure 2).

The brain of the robot is a *Gumstix Overo Fire Board*, which uses an energy-saving ARM processor (the same processor used in most smartphones today). With a clock frequency of 600[MHz] and 256[MB] of RAM, the robot is able to run a full Linux operating system,

including a graphical user interface. In order to work on the operating system, monitor and keyboard can be attached. In most cases however, the robot is accessed via wireless network (W-LAN), which is also included on the main board.

The energy consumption of the robot allows it to be active for at least 23 minutes with a fully charged 3200[mAh] battery pack, assuming that all its motors constantly draw the maximum of energy. In practice however, not all motors are constantly used at full speed, and therefore the robot can usually be used longer than half an hour.

5.2 Robot interface

For controlling the sensors and motors of the robot, a *Player* driver has previously been developed. *Player* provides an interface and a protocol for standardized communication with robots and their sensors [GVH03] (such as motor position sensors, distance sensors, cameras), as well as actuators (such as servo motors, LEDs and LCD displays). All of the robot's capabilities can be accessed via the interface provided by Player. This interface is used via a network protocol, and is therefore hardware-independent.

An overview of the experimental setup can be seen in Figure 7. The motors and sensors of the robot are accessed in a hardware-dependent manner via the *Player* drivers. The *Player* server offers this functionality in a platform-independent manner over a network protocol. The controller program runs on a computer that is connected via network to the robot, and is able to remote control the robot. The implementation details of the controller are further discussed in Section 5.5.

5.3 Simulation

Since the *Player* interface to the robot is platform-independent, a simulated robot can be exchanged for the real robot. A simulation tool for *Player* is provided by a separate

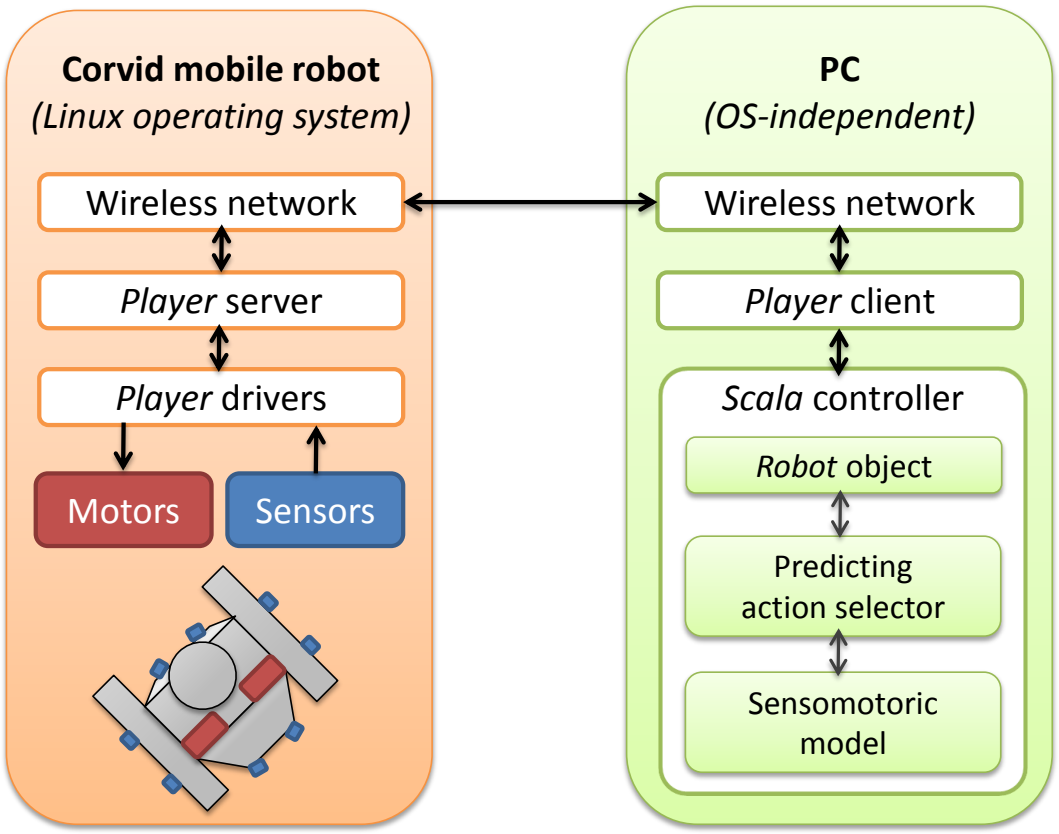

Figure 7: Experimental setup with the *Corvid* mobile robot.

project, *Gazebo*[1]. Gazebo is a 3D robot simulator which uses a physics engine to simulate realistic sensor feedback. In principle, and this is the goal of *Player*, a controller program can be used for a simulated robot and for the hardware without any changes. For this project, *Player/Gazebo* was used to simulate the robot before testing the controller on the real device.

[1]Player and Gazebo are open source projects and can be found here: `http://playerstage.sourceforge.net/`

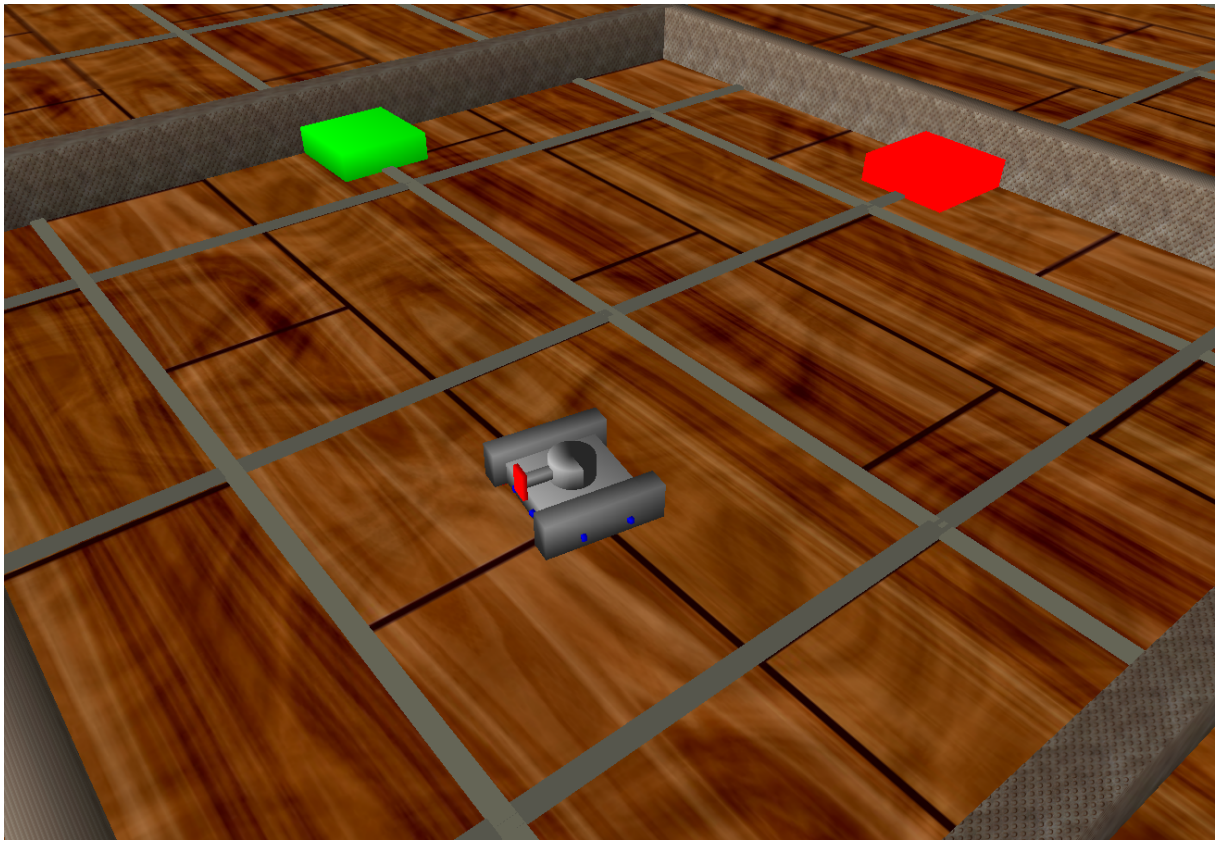

Figure 8: *The simulated version of the robot.* A 3D description of Corvid was created in order to simulate its motors, distance sensors and camera with the *Gazebo* simulator. The software uses a physics engine to simulate the robots interactions with the environment, and to provide realistic sensor feedback.

5.4 Learning architecture implementation

As mentioned previously, the Corvid mobile robot is a tracked vehicle that is equipped with 8 ultrasonic sensors which measure horizontal distances in every direction. For the purposes of this project, only those distance sensors and the movement motors are used. The 8 distance sensors deliver real-valued measurements which are used as the sensor inputs $S(t)$ (from Section 4.1) that the agent should learn to predict. The actuator output $M(t)$ consists of two values, forward speed and turn speed respectively, both in the interval [-1;1]. A negative forward speed causes the robot to go backwards, while a

positive or negative turn speed causes the robot to turn right or left, respectively. One time step will last 0.3 seconds, since that is the update interval of the robots ultrasonic sensors.

5.4.1 Neural network predictor

As a prediction module, a standard feed-forward neural network is used [RHW86], that receives motor and sensor data from the past two time steps, as well as future motor states as input (a concatenation of the vectors $SM(t-1)$, $SM(t)$, $M(t+1)$, which results in a total of 22 input values). The desired output of the predictor is the vector of future sensor state changes $\Delta S(t+1)$. Therefore, the neural network will only predict changes in distance measurements, instead of total distance values. The neural network outputs are normalized to [-1;1], where the maximum value of 1 corresponds to 10 centimeters. The absolute predicted measurement values are then given by Equation 2.

$$S(t+1) = \Delta S(t+1) + S(t) \tag{2}$$

For the purposes of neural network training, the values of the distance sensors are normalized to the interval $[0;1]$, representing 0 to 3 meters, while the motor values are in [-1;+1].

The neural network was trained with an optimized version of the backpropagation algorithm, "resilient backpropagation" (RPROP) [Rie94]. After some preliminary testing, the neural network was set to have one hidden layer and 8 hidden units. Given by the sensomotoric inputs and sensor outputs, the network has 22 input units and 8 output units. As RPROP is self-adjusting, no additional parameters are needed.

5.4.2 Action generation

In order to use the predictions from the neural network, actions have to be generated. At each time step t, the robot is allowed to choose between 5 actions:

1. stop (or slow down, if moving fast),
2. accelerate (go forward)
3. decelerate (go backward)
4. turn right
5. turn left

Using the predictor model, it is now possible to calculate predicted outcomes $\overline{S_x}(t+1)$ for each of the five actions x.

5.4.3 Reward signal

In order to evaluate the predictive model, we will consider 3 different goal states that the robot should try to reach. To communicate how well the agent is doing, it receives a real-valued reward signal.

Also, the agent needs a way to predict reward signals from its state. This reward predictor module was described previously in Section 4.4. For our purposes, the reward signals will be formalized in a way that can be calculated directly from the robots state $SM(t)$.

It is therefore not necessary to train a separate neural network that predicts rewards. The following goals and reward formulas are used in the experiments:

1. *"hold a distance of 1 meter to surrounding objects"* (which can be formalized as Equation 3)

$$reward = -|min(S(t)) - 1| \tag{3}$$

where $(min(S(t))$ is the minimum of the 8 distance sensor values at time t.

2. *"drive to the center of the room"* (or "maximize the distance sensor readings" which can be formalized as Equation 4)

$$reward = \sum_{i=0}^{7} \sqrt{S_i(t)} \tag{4}$$

which is just the sum of the square roots of all 8 distance sensor values, therefore giving smaller distances more weight on the reward.

3. "keep a forward distance of 80 centimeters" (which can be formalized as Equation 5)

$$reward = -|S_0(t) - 0.8| - |S_7(t) - 0.8| \tag{5}$$

where $S_0(t)$ and $S_7(t)$ are the two distance values from the forward ultra-sound sensors.

5.4.4 Action selection

The goal of the agent is to maximize the reward signal. Using the generated actions from Section 5.4.2, it is possible to generate many possible future states of the robot. In order to maximize the rewards, the agent will have to choose an action that will lead to the best possible future state.

Another way to describe this problem is by using a decision tree, in which the nodes represent states of the agent, connected by different decision branches.

In our implementation, the agent will enumerate all possible actions 2 time steps into the future, beginning with all 5 predictions of $\overline{S}(t+1)$, since we consider the 5 different

actions from Section 5.4.2. Starting with each of these 5 outcomes as a current agent state, the predictor can be used to predict all 25 possible outcomes of $\overline{S}(t+2)$.

Depending on which task the robot should try to perform, the possible outcomes $\overline{S}(t+2)$ can be evaluated using one of the reward formulas from Section 5.4.3. In our experiments, the robot always chooses the action with the highest predicted reward. Since one time step lasts about 0.3 seconds in our experiments, the robot is able to "plan ahead" about 0.6 seconds.

Chapter 6 will explain how this method of selecting actions will be tested experimentally.

5.5 Software implementation

Since the *Player* interface uses a network protocol, the controller program could be developed and run entirely on a standalone computer, that was connected to the robot via wireless network. This means that the controlling software was not actually running on the robot, but was instead communicating with it in order to receive current sensor data and to control the actuators.

The software was written in the programming language *Scala*, which is a relatively new *Java*-compatible language. It was chosen because of its expressiveness, as it allows the programmer to use mathematical and functional expressions as well as object-oriented programming principles, while at the same time being relatively performant and capable of accessing existing machine learning algorithms written in *Java*.

The source code is an integral part of this master thesis, and is available as a separate download [2]. The controlling software has a simple graphical user interface that allows to select basic operations for our experiments (see Figure 9). Many parameters have to be provided in the source code itself. The screen displays the current distance data as received by the robot from all 8 ultrasonic distance sensors.

[2] For the full source code see `http://github.com/papauschek/corvid-devrob` (MIT/X11 license)

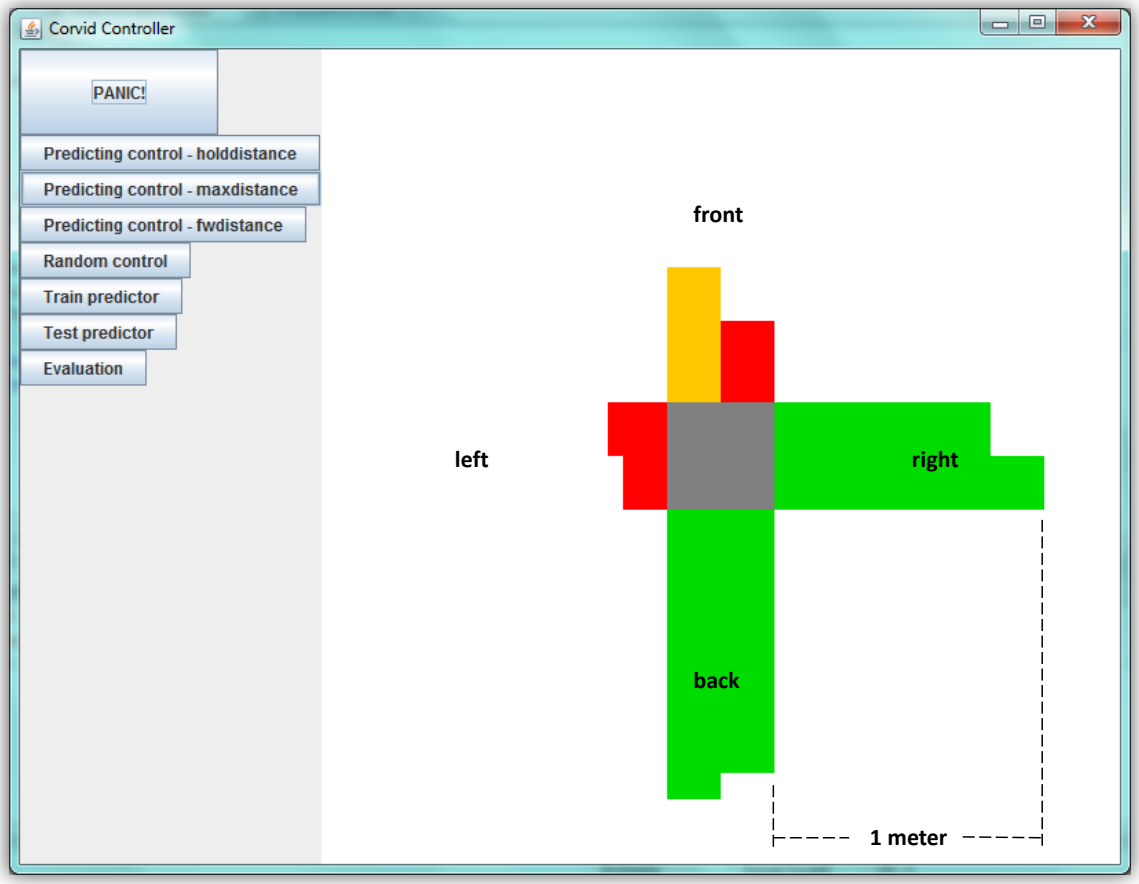

Figure 9: Graphical user interface of the controller software.

The program is structured into different classes. A combination of object-oriented and functional programming paradigms has been used during the software development process. The following list describes the most important classes and their function.

- **Main**: This object is the main entry point of the application. It shows the graphical user interface (as specified in the *MainWindow* class) which allows basic interaction with the software. Simultaneously, a network connection to the robot is established by an instance of the *Controller* class. If the robot cannot be reached, the robot controller cannot be used and only offline functionality is available (such as training a sensomotoric model from previously recorded sensor experiences).

- **Controller**: This class is responsible for establishing a network connection to the robot and controlling it. Different controlling routines can be used. For the experiments here, we used two different implementations which can be found in the *PredictingControl* and *RandomControl* classes. The controller makes sure that new sensor readings are taken about every 300 milliseconds from the robot, and stores the collected sensor, motor and reward data on disk. This data can later be used for building a sensomotoric model from the robots experience, or evaluating the average reward.

- **Robot**: The robot class encapsulates all capabilities of the robot and provides a standard interface for them. For example, all units are converted to meters, and the full forward speed is always 1.0, while full reverse speed is always -1.0. This class makes it possible to use the real robot in the same way as the virtual robot in simulation. Calibration data is provided via the object *RobotParameters*, which contains default settings for both simulation and the real device.

- **SensorModel**: After collecting sensomotoric data via the *Controller* class and a controlling mechanism such as *RandomControl*, the routines in the *SensorModel* object can be used to train a sensomotoric model. The main routine loads sensomotoric data from disk and filters it. The filter excludes duplicate sensor readings and sequences of readings which are too short or have time-delays outside the normal range of 250 to 350 milliseconds. This data is then used to train a new neural network using the *Predictor* class. The acquired neural network represents the sensomotoric model and is then saved to disk, and can be later used by the *PredictingControl* class to control the robot.

- **Predictor**: The predictor class represents a neural network that predicts future

sensor states from current and past sensomotoric experience. For neural network training, the well-known Java library *Encog*[3] is used. Our implementation uses the *R-PROP* training algorithm [Rie94], which needs no parameters except the number of hidden layers and neurons.

- **PredictingControl**: This class loads a previously trained *Predictor* (the sensomotoric model) from disk and uses it to predict future sensor data in order to choose optimal actions for the robot. It implements the action generation (see Section 4.3) and action selection part from the hypothesis (see Section 4.4). It also expresses the three reward formulas from Section 5.4.3 as *Scala* functions. The class takes an *exploration* parameter, which determines the probability of random exploration versus exploitation of the sensomotoric model.

- **Evaluation**: The *Evaluation* object contains routines for calculating the average reward from previously recorded experiments, including statistics such as confidence intervals. It also contains the routine used to evaluate the neural network prediction performance after training with different amounts of sensomotoric data.

For more detailed information and documentation of all other classes and objects, please refer to the full source code at `http://github.com/papauschek/corvid-devrob`.

[3]For further information about the Java library *Encog* see `http://github.com/encog` and `http://www.heatonresearch.com`

6 Experimental setup

In order to test how well the system is able to perform the given tasks from Section 5.4.3, the system first needed to gain some experience using its sensors and motors, in the form of collecting data.

In all the following experiments, the robot was be put into a closed environment (see Figure 10) with a diameter of about 3.5 meters.

Figure 10: The closed area in which the mobile robot was free to move around. The largest diameter is about 3.5 meters long.

6.1 Learning phases

In order to measure how the robot improves its behavior over time, the experiments were split into many learning phases. After each learning phase, the neural network was trained with the new sensor and motor data, in order to build a prediction model. In fact, many neural networks with different parameters were trained (one hidden layer, varying amounts of hidden neurons), and the best one was kept.

Learning phase 1

In the first learning phase, the agent was not actually given data to learn and was tested with a randomly initialized neural network predictor. As can be expected from an untrained neural network, its predictions were not useful to the agent. But these experiments served as a baseline, which were then compared to the trained predictors. The robot was tested with all 3 goals with this random predictor, each for about 1000 time steps (around 5 minutes):

1. hold a distance of 1 meter to surrounding objects
2. drive to the center of the room
3. keep a forward distance of 80 centimeters

During testing, in all phases, the robot is always configured to pick the action with the highest predicted reward.

Learning phase 2

To collect some initial training data for the next learning phase, the robot was first put into an exploration phase, where it selected random actions (see Section 5.4.2 for an

overview of all possible actions) for a very short period of time of 100 time steps, which took about 30 seconds. Now, in learning phase two, the neural network was trained with the collected data from this exploration phase.

Again, the robot was tested on all goals for about 5 minutes.

Learning phases 3 and 4

For the remaining learning phases 3 and 4, this procedure was repeated, with a few differences. The time of collecting sensor data was increased, and instead of randomly selecting actions all the time, the robot was selecting random actions only with 50% chance, and spent the other time trying to maximize reward of a goal that was randomly selected every 30 seconds. This was done so that the robot could already exploit some of the knowledge it acquired in previous learning phases. It helps to make sure that the robot collects sensor data that is somewhat relevant to the tasks it will have to perform.

An overview of the number of training data collected after each learning phase can be seen in Table 1.

Learning phases overview		
Learning phase	Total time trained	Total time steps trained
1	0 minutes	0
2	0.5 minutes	100
3	5 minutes	1000
4	25 minutes	5000

Table 1: Total time steps trained after each learning phase. Each time step corresponds to 0.3 seconds real-time.

As mentioned before, after each learning phase the robot was evaluated for each task for about 5 minutes. It is expected that the average rewards will become higher as the

robot improves its predictive model after each learning phase. Furthermore, the predictive model will cease to improve significantly after a certain point. We expect that it will not make sense to continue learning after the fourth learning phase, in which the robot was trained with about 25 minutes worth of data.

6.2 Choice of environment

In the preliminary experiments, there was the problem that the robot has to be allowed to move in a random manner. This would normally cause the platform to hit obstacles and walls, and be in danger of damaging itself. To avoid mechanical damage, we had to implement a safety mechanism that automatically slows the robot down when the distance sensors measure a distance of less than 0.4 meters. This is a general problem of reinforcement learning, or learning by experience, in robotics. It will be further discussed in Chapter 8, as this topic will apply to a wider range of developmental robotics experiments.

Furthermore, depending on the angle of reflection, some objects cannot be recognized reliably by the ultrasonic sensors, as Figure 11 illustrates. It shows one of the ultrasonic distance sensors of the robot on the left side, sending a sound in the direction of the obstacle on the right. In the first case, a surface with about 45% degrees angle reflects the sound away from the sensor, which cannot pick up the signal to measure the travel distance. As the second case indicates, round objects are very well detectable by ultrasonic sensors, as they reflect sound back in all directions.

In this sense, the chosen environment for the experiments was not optimal. An optimal environment for ultrasonic sensors would consist only of round objects, such as the legs of the green chair that was placed in the middle of the environment, or the yellow pillars (see Figure 10).

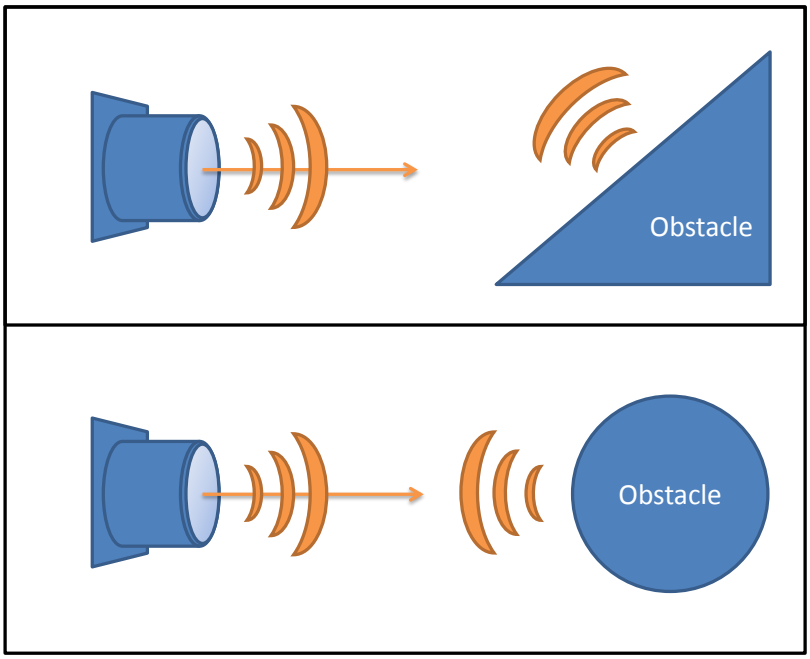

Figure 11: Illustration showing the reflections of sound by different objects. The ultrasonic sensors on the left side need to measure a reflection of the sound wave they are sending out, in order to calculate a distance to an obstacle. Some obstacles, especially flat objects, do not reflect back the sound very well, making their detection difficult.

The following section will give a detailed account of all the results obtained from the experiments.

7 Experimental results

The experiments with the learning architecture have been performed in different learning phases, as described in the previous section. After each learning phase, there was a 5-minute evaluation of each task. This section will describe the results of each of these 5-minute evaluations, and focus on only one task at a time.

In all of the following results, the displayed results are average reward values measured after each learning phase. Table 1 shows how many samples the agent was able to use to build its prediction model.

7.1 Experiment: Hold distance

In this experiment, the goal of the robot is to keep a constant minimum distance of 1 meter to its surroundings. The results from this experiments can be seen in Figure 12, which shows the average reward after each learning phase.

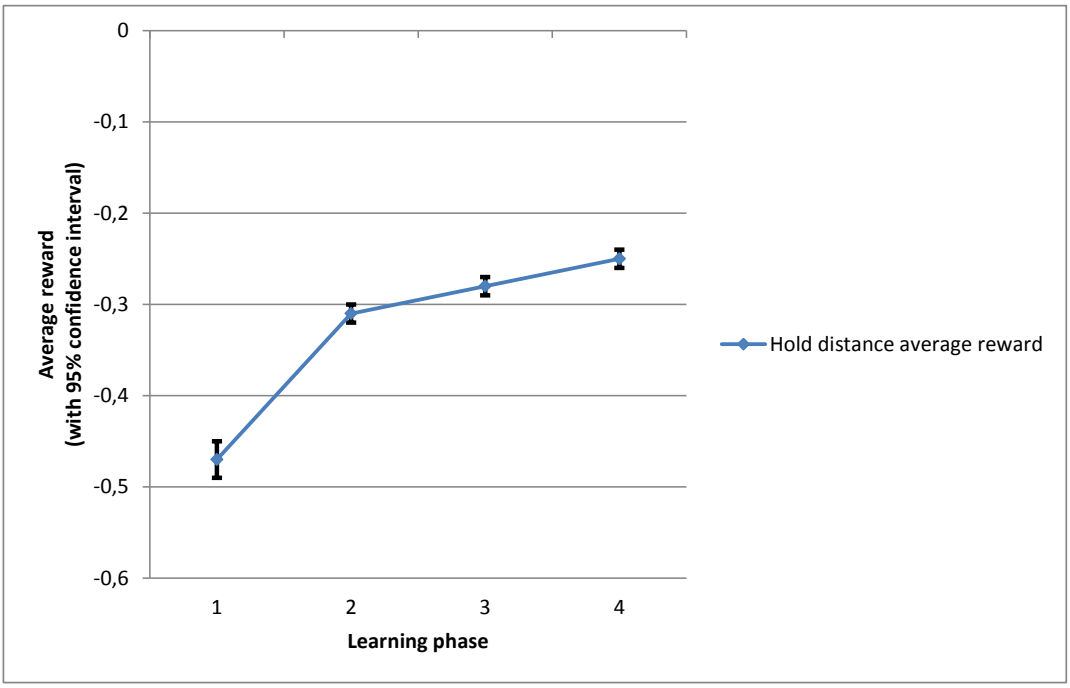

Figure 12: Results from "hold distance" experiment, showing the average rewards after each learning phase. The error bars show the 95% confidence interval. The maximum possible reward in this experiment is 0. Table 1 shows how much time the agent was trained before each learning phase.

As can be seen from the equation that was used to calculate the reward for this goal (see

Equation 3), the result is necessarily a negative value. The best possible reward is 0, which can only be achieved if the smallest sensor reading is exactly 1 meter. Due to the noise of the sensors, this could never be achieved for a long duration. The best result was an average reward of -0.25, which corresponds to an average distance of 25 centimeters to the optimum.

In phase 1 of the experiment, the robot was moving around aimlessly, since the prediction model was randomly initialized and not useful. The safety mechanism outlined in Section 6.2 prevented the robot from damaging itself. After phase 2, the robot already exhibited some goal-directed behavior, but never managed to stay at an optimal solution. In phases 3 and 4, the robot would sometimes stop at an optimal position when the reward was very high, and when predictions would indicate that any movement would result in a lower reward. If the robot would find such an optimal place, an obstacle would be placed close to the robot, forcing it to find another optimal solution. This was done in order to make certain that it was not just by chance that a solution was found, and to ensure that the different performances after each learning phase were reproducible.

7.2 Experiment: Maximize distances

In this experiment, the goal of the robot is to maximize the distance sensor readings, which can be interpreted as trying to find the center of the room. The results from this experiment can be seen in Figure 13, which shows the average reward after each learning phase.

The equation that was used to calculate the reward is Equation 4), which always gives a positive value that depends on the sum of the square root of all distance sensor readings. Therefore, the maximum reward depends on the size of the environment. Since the environment here is very small, the maximum reward is about 10.0. If the room would be large enough to allow a maximum distance measurement of 6 meters for all 8 sensors, the

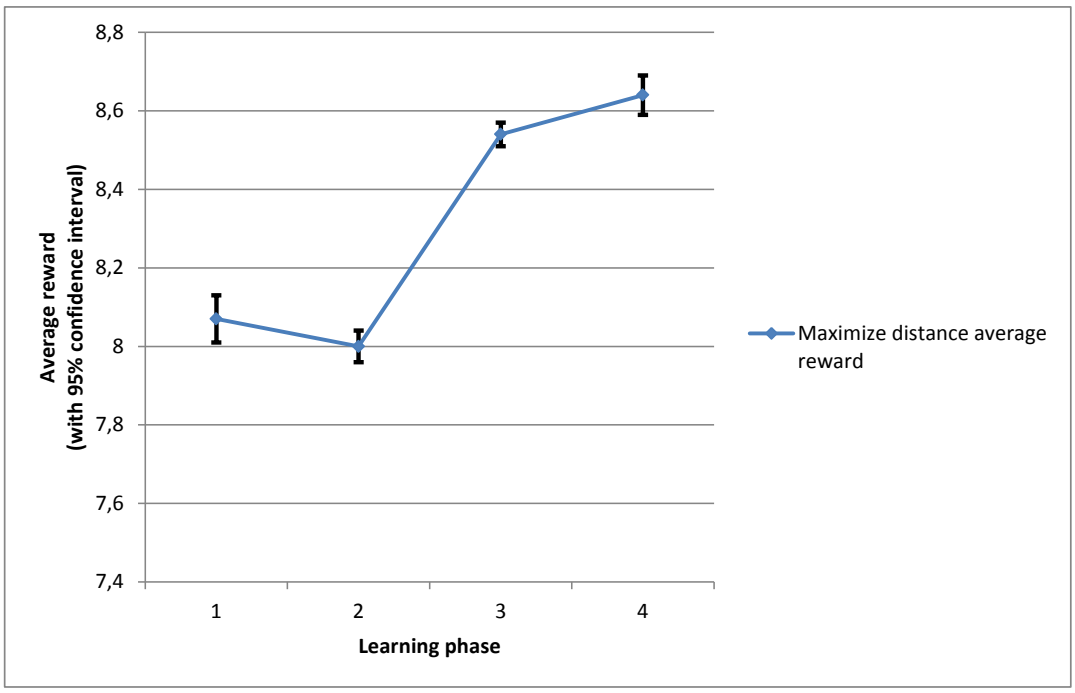

Figure 13: Results from "maximize distances" experiment, showing the average rewards after each learning phase. The error bars show the 95% confidence interval. The maximum possible reward in this experiment is about 10.0. Table 1 shows how much time the agent was trained before each learning phase.

maximum possible reward would be about 20.0. In the beginning, the untrained robot was able to achieve an average reward of about 8.1. After learning phase 4, the robot was getting consistently higher rewards, which averaged to 8.65.

In phase 2, the robot did not move at all, except when an obstacle was placed very close to it. After learning phase 3, the robot was spinning constantly. When placed close to a wall, it would gain some distance to the wall and continue spinning, but getting relatively good rewards. In phase 4, the robot was wandering around normally, keeping large distances to the walls and sometimes crossing the middle of the room, but not staying there.

7.3 Experiment: Keep forward distance

In this experiment, the goal of the robot is to keep a forward distance of 0.8 meters to obstacles. The results from this experiment can be seen in Figure 14, which shows the average reward after each learning phase.

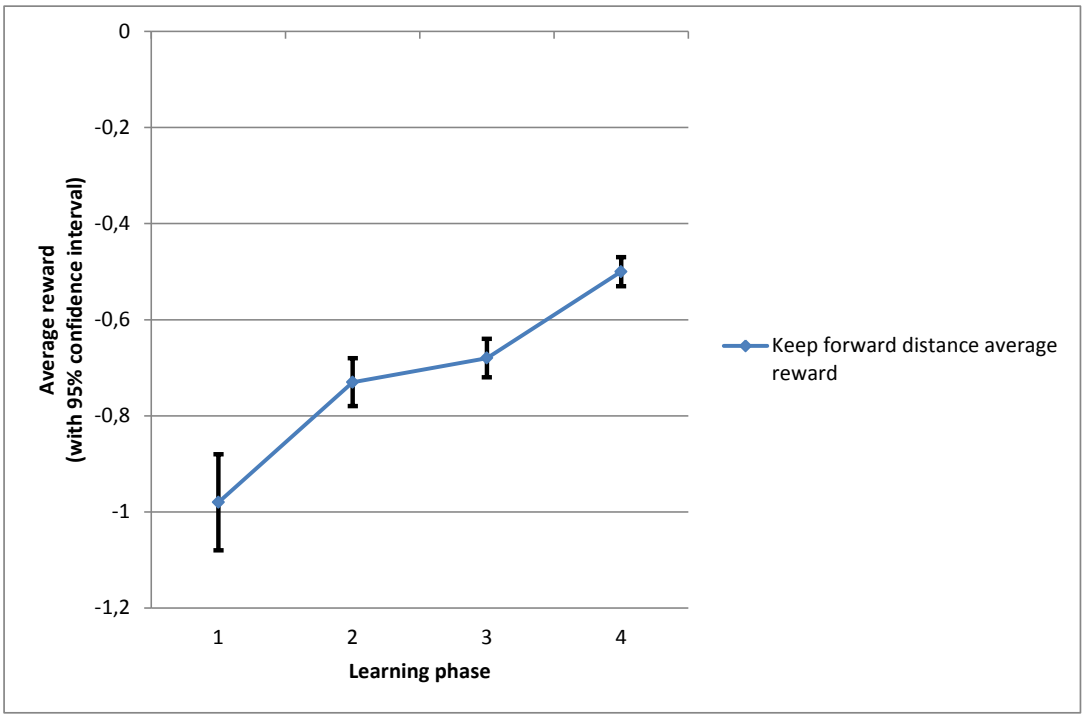

Figure 14: Results from "keep forward distance" experiment, showing the average rewards after each learning phase. The error bars show the 95% confidence interval. The maximum possible reward in this experiment is 0. Table 1 shows how much time the agent was trained before each learning phase.

The reward was calculated using Equation 5, which gives a negative reward for the distance to the desired 0.8 meters of both forward ultrasonic sensors. Similar to the "hold distance" experiment, the maximum reward is 0.

In phase 1, in which the robot had no useful prediction model, the result was a behavior

in which it was switching between going forward and backward, never turning around. In phase 2, the robot started to turn around often, sometimes being able to find a flat wall and stay within the goal distance. After phase 3, the behavior was improved, and following learning phase 4, the robot was often able to find a flat wall and stay within the target distance. The highest average reward achieved was -0.5 after the last learning phase. Since this value is the negated sum of two distance differences measured in meters, the difference to the goal of 0.8 meters was 25 centimeters on average, on each sensor.

7.4 Qualitative results

Overall, the robot was already showing goal-oriented behavior after the second learning phase. In two of the experiments, the "hold distance" and the "keep forward distance" experiments, the robot was able to find locally optimal solutions and stay close to those solutions after the third and fourth learning phase. In the "maximize distances" experiment however, the robot still was in constant movement after the last learning phase. It did not find a locally optimal solution and stop moving, in contrast to the other experiments.

In some cases, the robot was not able to keep a safe distance to obstacles. This can be attributed to the properties of the ultrasonic distance sensors. The sensors only have a 40° receptive field, as illustrated in Figure 15. Because the way the sensors are placed on the robot, there is a blind spot in which objects cannot be sensed.

7.5 Predictor performance testing results

After each learning phase during the experiments, neural networks were trained with the collected training data, and the best neural network was chosen for evaluation purposes. In this section we analyze in more detail what the neural networks were able to learn, and how well they predicted sensor readings.

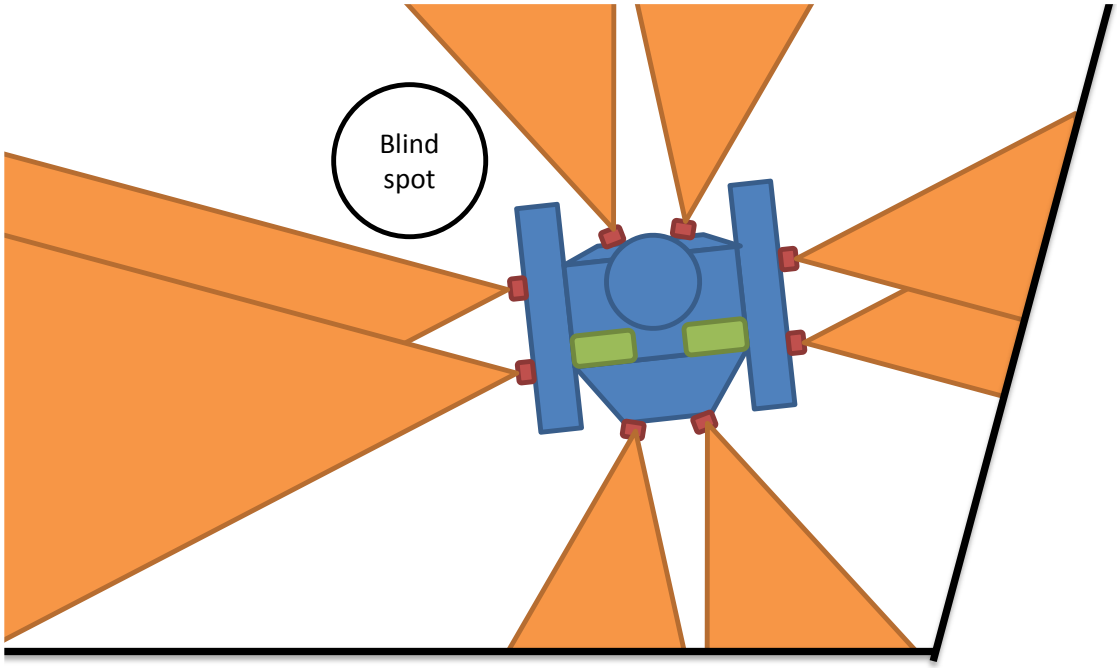

Figure 15: Illustration showing the arrangement and 40° receptive field of Corvids ultra-sound sensors, as well as several areas where no obstacles can be detected ("blind spot"). The sensors have a maximum range of 6 meters.

For this purpose, the neural networks were evaluated after training them with a different amount of data pairs. The testing set always consisted of 10000 data pairs (time steps), which corresponds to about one hour of collecting sensor data. In order to get the best results, the neural networks were trained with different parameters. The networks were trained with 2, 4, 6, 8, 10, 12, 14 and 16 hidden neurons each, with a single hidden layer. Each neural network configuration was trained three times to get reproducible results. Only the best result is displayed in the following graphs.

7.5.1 Prediction of direction

Figure 16 shows how often the neural network was able to correctly predict whether the distance sensor reading would increase or decrease. In other words, the percentage of

correctly predicting the sign of the distance change was measured, and averaged over all 8 distance sensors. It can be seen that the prediction performance of an untrained neural network is about 50% as expected, and increases steadily after providing more and more training data to the network. A maximum performance of about 64% was reached after training and testing with a set of 10000 time steps.

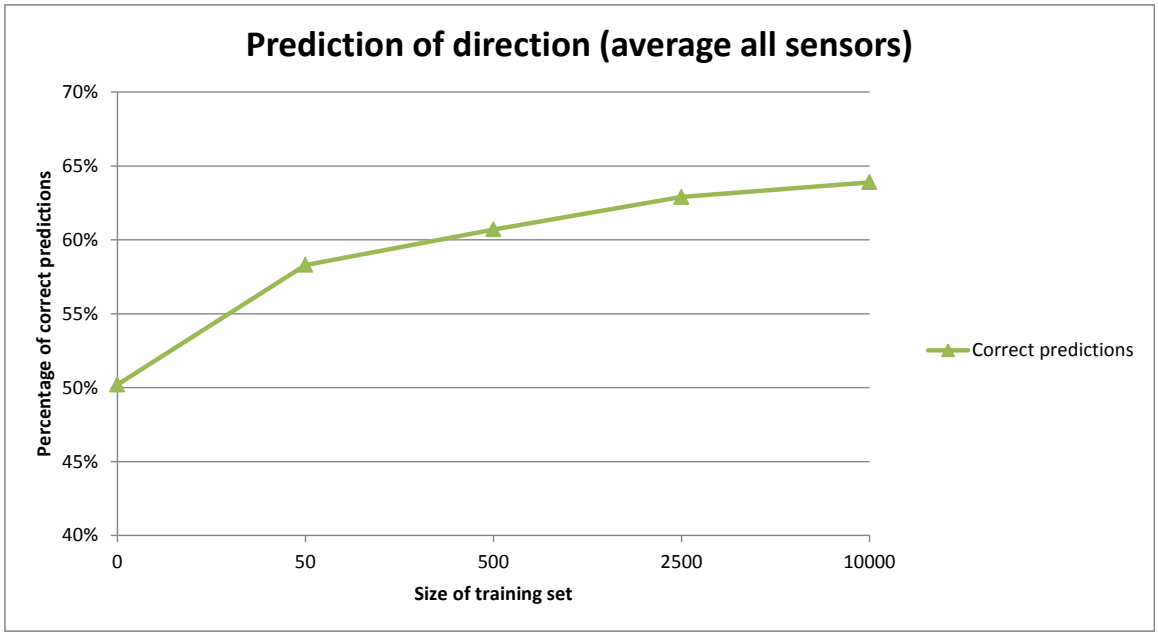

Figure 16: Percentage of correct predictions of the direction of change of all distance sensors. Only the best neural network results are shown, while varying the training set sizes, and keeping the testing set size at a constant 10000 time steps.

As it was interesting to see whether there was any difference in prediction performance between the forward/backward sensors and the sensors on the side of the robot, we also measured the percentages for both sensor sets. Figure 17 shows the results for this comparison. It can be seen that it was easier for the network to predict the front and back facing sensor readings, as compared to the distance readings on both sides of the robot. The quality of the prediction of both sensor sets increases as more training data is provided, up to a maximum of 69% for the forward/backward facing sensors, and 59% for

the sensors on the side, as the training set is increased to 10000 time steps.

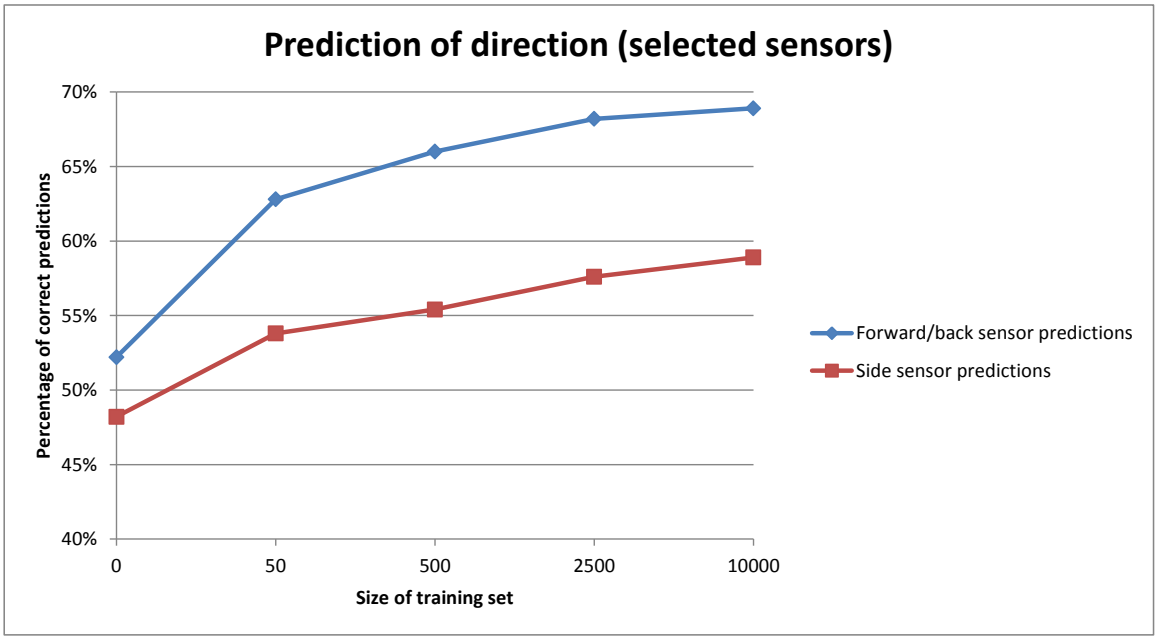

Figure 17: Percentage of correct predictions of the direction of change of forward/backward and side distance sensors. Only the best neural network results are shown, while varying the training set sizes, and keeping the testing set size at a constant 10000 time steps.

7.5.2 Prediction of distances

The performance of the neural networks was further analyzed by measuring the difference between the predicted and the actual sensor readings. This was done by calculating the *root mean squared error* (RMSE) of the differences (in centimeters) after testing each neural network. Figure 18 shows the results for each training set.

In addition to the neural network prediction performance, the residuals of the naive predictor are shown. The naive predictor simply always assumes that the sensor value will not change, and is used as a baseline for comparison. In the beginning, the neural network actually performs worse than the naive predictor, but this quickly changes after

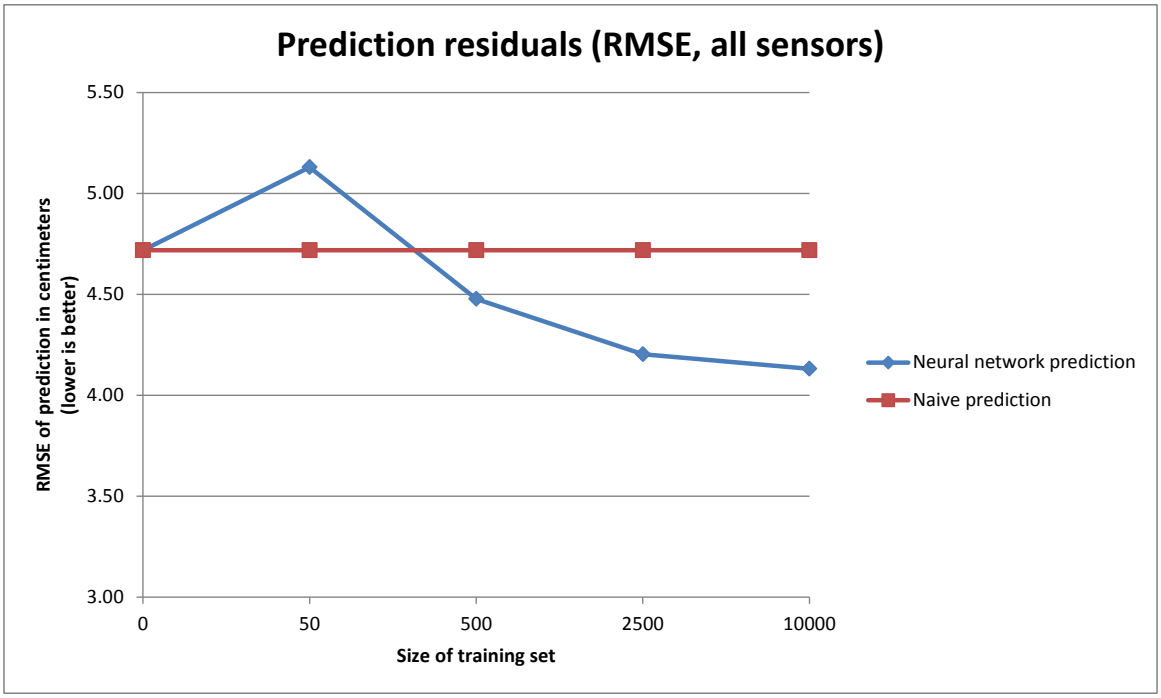

Figure 18: Residuals (root mean squared error) of neural network predictions of all sensor readings compared to the naive prediction. Only the best neural network results are shown, while varying the training set sizes, and keeping the testing set size at a constant 10000 time steps.

a training set size of 500, which corresponds to about 2.5 minutes of collecting sensor readings, is reached.

Again, it was interesting to see if there is a difference between the prediction performance for the forward/backward facing sensors and the sensors on the side of the robot. The results can be seen in Figure 19. Here, the results of the neural networks should only be compared to the respective naive prediction, as comparisons between them are meaningless due to the different nature of the sensor data. The results show that the forward/backward facing predictions are much better compared to the side sensor predictions, when using the naive predictions as a baseline.

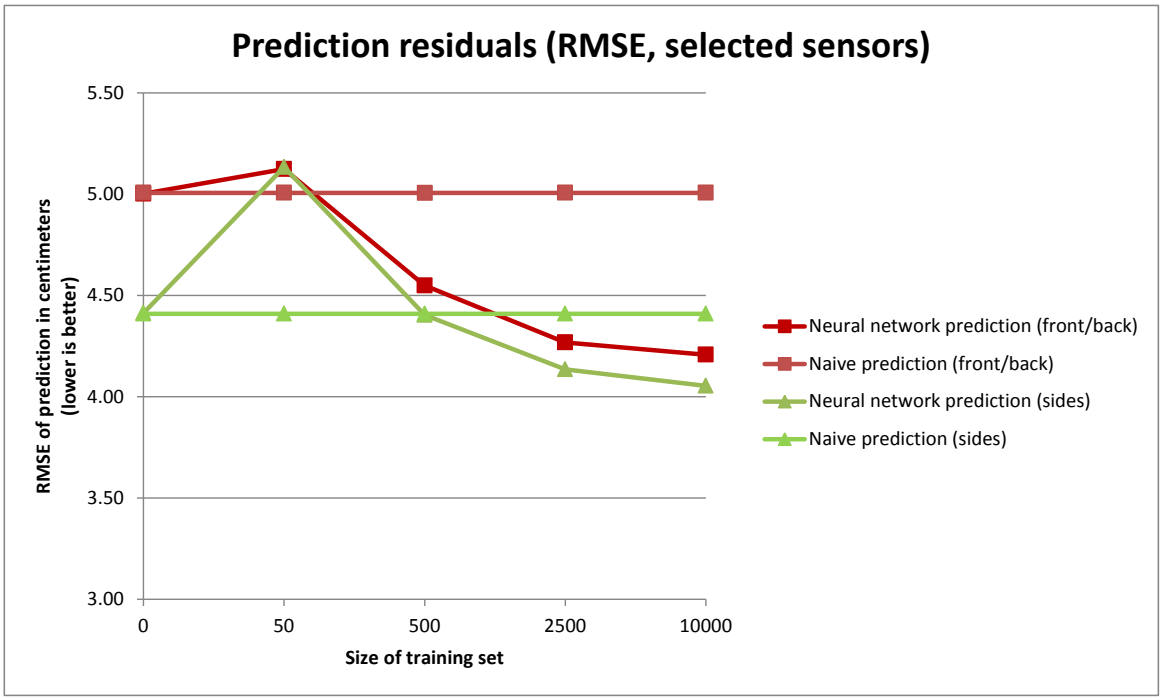

Figure 19: Residuals (root mean squared error) of neural network predictions of backward/forward and side sensor readings compared to the naive prediction. Only the best neural network results are shown, while varying the training set sizes, and keeping the testing set size at a constant 10000 time steps.

8 Discussion

In Chapter 4 we introduced a system that would be able to learn automatically. After presenting the experimental results in the previous section, we will now discuss what follows from these outcomes.

One of the central ideas of developmental robotics is incremental learning. To test whether the robot would improve its goal-driven behavior incrementally, the experimental setup was chosen in a way that makes it possible to measure this. Experimental results were presented in Chapter 7. In all of the three experiments, the robots goal-directed behavior changed over the course of the 4 learning phases, and the received rewards increased after each phase, or at least stayed the same within the confidence interval (see Figures 12, 13 and 14).

8.1 Goal-oriented behavior performance

In the "hold distance" experiment for example, the average reward steadily improved after each learning phase. Even after the second learning phase, which used very little data (just 100 time steps), performance improved much compared to the initialized predictor of the first phase. This can be attributed to the way the neural network starts to build a model of the training data. As shown in Section 7.5.1, even after a training set of 50, the neural network is able to start predicting the distance change directions better than a random predictor (see Figure 16 in that section).

The "maximize distance" experiment shows a slightly lower reward result after the second learning phase, which can again be attributed to the generalization properties of the neural network after being trained with just 50 data points (and tested with the other 50). The generalization tests in Section 7.5.2 support this: Figure 18 shows that the neural network has a worse performance after training with 50 data points, than before. The prediction of the distance sensor values seems important for this experiment. This stands in contrast to

the "hold distance" experiment, where it was only important for the predictor to correctly predict the *sign* of the distance change.

Similar to the first experiment, the results from the "keep forward distance" experiment also show steady improvement after each learning phase. We can assume that this is the case due to the similar nature of the experiments: in both cases, a correct prediction of distance change direction was needed to keep the desired distance. The only difference was the number of sensors involved.

From these experiments it can be seen that the initial belief was correct: after each training phase, the system generally achieved better results than before. Also, the improvements became less and less significant after each learning phase, especially in the first two experiments. In Figure 12 we can see that the improvement from the second phase to the third phase was similar to the improvement to the last phase. However, we have to keep in mind that in the last phase we used 5 times more training data than before. Even more convincing are the results from the second experiment in Figure 13: The improvements in the last learning phase are minimal compared to the ones before. Indeed, we can conclude that the neural network predictor reached a point of diminishing returns after training with more than 1000 time steps of sensor data. The following analysis of the neural network predictors also supports this idea.

8.2 Neural network prediction performance

In order to get a sense of what the neural network was able to learn from the data, further tests were conducted in Sections 7.5.1 and 7.5.2, each measuring different prediction characteristics. As was expected and can be seen in Figure 16, the prediction of the *sign* of the distance change improves as the system receives more training data. Maybe more unexpected, after a training set of 50 time steps the prediction of the actual distance values was even worse than the naive prediction, (see Figure 18, and only started to improve with bigger training sets.

We can assume that it is very easy for a neural network to learn some of the basic correlations between motor control and sensor input. Furthermore, the correlations between forward/back motor control and forward/backward sensors should be especially easy to train. Even in a complex environment, the forward distance sensor readings would generally decrease as the forward motor speed is positive. In order to see whether this was indeed picked up by the neural network, a comparison was made between the prediction performance of the forward/backward facing sensors, and the sensors on the side, which can be seen in Figures 17 and 19. Both graphs suggest that front/back facing prediction was much easier to learn than prediction on the left and right side of the robot. Forward sensor prediction of the sign of the distance change was about 19% higher than the prediction percentage of the side sensors (see Figure 17). In a similar fashion, Figure 19 shows that the forward/backward facing distance predictions are much better compared to the side sensor predictions, when using the naive predictions as a baseline.

These differences in prediction quality between the differently facing ultrasonic sensors also show some of the important limitations of this implementation: the robot does not try to build a map of its environment, and has no real memory of its current location. The focus here was to show that a basic motor/sensor model of the robot can be used, in combination with a reinforcement learning architecture, to reach certain goal states.

During the implementation of this architecture, the agent was never explicitly told how to reach the goal state. The robot was able to do that by building the sensomotoric model, and by picking the best action according to the expected reward. This architecture has the advantage, that it is general enough to be useful with different kind of sensors and robotic platforms. Even though our implementation was able to better predict the forward facing distance measurements than the distances on the side, this knowledge was not provided to the system beforehand. In fact, no knowledge about the sensor configuration was represented. If the sensors would be switched around, and the robot be allowed to train everything again from the beginning, the results would be the same.

8.3 Principles of developmental robotics

Section 2.2 explained five principles of developmental robotics: the principles of verification, embodiment, subjectivity, grounding and the principle of incremental development. Have these principles been implemented in this work?

1. *Verification principle:* Our agent cannot actually be sure that it reached a target distance of 1 meter, as in the first experiment, because the ultrasonic sensors are prone to certain inaccuracies. But given the limitations of its own sensors, the robot built a model if its own sensors and motors, and validated it during each learning phase. Therefore, the verification principle has been satisfied.

2. *Principle of embodiment:* Since the robot has a body, this principle has been followed.

3. *Principle of subjectivity:* All of the data that has been used by the robot to build its model, has been collected by the robot itself. Therefore, the principle of subjectivity has been implemented. The time constraints of developmental robotics have also been shown implicitly: there is no quick solution to learning. All learning involves real environments and real sensor experience. The trained model can only be used on robots with identical hardware configuration.

4. *Principle of grounding:* Grounding has been achieved through the coupling of the robots actions and their observable outcomes. The repeatability of these actions and outcomes is reflected in the trained sensomotoric model.

5. *Principle of incremental development:* There were some challenges in experimenting with a robot that, in the beginning, was just motor babbling and exploring its environment without any experience, and was in danger of damaging itself. But after the average received reward was evaluated after different time spans of learning, incremental improvements in the robots behavior could be observed.

9 Conclusion

In this work, we have shown how a robot can learn autonomously by exploring its environment and collecting data about it. After learning from the observations, it was able to build a model of the interactions between its actuators and the environment, which were indicated by its sensors. After training, this experience-based model was verified by using its predictions to reach certain sensor states. The learning architecture has been evaluated using three different goals, and the robot successfully used its sensomotoric model to reach them.

During the evaluation, we were able to show a progress in learning, as the robot was able to collect more and more data about its motors and its environment. In the beginning, the robot failed to meet its goals. After several learning phases the robot was able to reach the goal states in the "hold distance", "maximize distance" and "keep forward distance" experiments with consistently good results. In the last two learning stages, it could be seen that the performance did not improve significantly after increasing the amount of sensor data with which the prediction model was trained.

In the earlier stages of training, the robot needed to be allowed to experiment with its actuators, which can result in mechanical damage. This may be one of the reasons why reinforcement learning is difficult to implement in robotics, as many robotic systems are not able to detect such failures or would need expensive repairs after being damaged.

9.1 Future of developmental robotics

The principle of verification is at the heart of developmental robotics, and has far-reaching implications. It follows from it, that the key to *understanding* is the *ability to question*. Robots that use a lot of preprogrammed knowledge cannot question whether a reading from a sensor has a certain meaning, or not. Many assumptions may be built in from the beginning, and cannot be falsified. If assumptions are wrong in some cases, the robot

cannot learn from that failure. Experimentation and allowing for failure is crucial for systems which need to be able to verify their own actions.

It will also allow building robots that are more robust to changes in their environment, or even programming robots with different sensor and motor configurations. We have shown this by implementing a system which did not have any prior knowledge about the direction and configuration of its sensors and motors.

However there are also a lot of drawbacks when developmental robotic solutions are directly compared to classical top-down robotics. Developmental robotics may only in a few cases be already practical, in the sense that solutions perform better than their non-developmental counterparts. If the principle of subjectivity is taken seriously, the robots will have to spend a lot more time with learning about their environment, compared to their non-developmental counterparts.

But developmental robotics it is a very young and promising field, and progress in robotics will depend on research in many related fields, including cognitive science. Developmental robotics is in a position to combine some of the promising ideas of fields such as developmental psychology, in order to find a way to solve the problem of scaling out in robotics.

9.2 Outlook

In contrast to previous developmental robotics research on the robot platform *Corvid*, which evaluated an intrinsic motivation system, we showed how an autonomously learned model can be used to induce goal-oriented behavior. The next challenge would be to implement a system that combines the learning of affordances in an architecture that allows to define goals for the robot. This would also make it easier to evaluate these acquired affordances. But related work with affordance learning has shown that there are

still new approaches needed for this to work on a mobile robot, because of the potential vastness of the exploration space.

Learning autonomously becomes more and more important in robotics, and it is our belief that research from developmental psychology will play an important role here. It is still very challenging to develop robots that learn about their environment without the direct aid and explicit programming by humans. We believe it is possible to leverage some of the knowledge of both fields, robotics and developmental psychology, and to push the boundaries of what robots are able to learn autonomously in the near future.

List of Figures

References

[BO09] Adrien Baranès and Pierre-Yves Oudeyer. R-IAC: Robust Intrinsically Motivated Exploration and Active Learning. *IEEE Transactions on Autonomous Mental Development*, 1(3):155–169, 2009.

[GVH03] B. Gerkey, R. T. Vaughan, and A. Howard. The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems. In *Proceedings of the 11th International Conference on Advanced Robotics (ICAR'03)*, pages 317–323, 2003.

[KLM96] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[LCJ10] Xiuzhi Li, Wei Cui, and Songmin Jia. Range scan matching and Particle Filter based mobile robot SLAM. In *IEEE International Conference on Robotics and Biomimetics (ROBIO) 2010*, pages 779–784, 2010.

[Län10] Christoph Längauer. *Lernen von Affordance für die Roboter Navigation*. PhD thesis, Fachhochschule Technikum Wien, Austria, 2010.

[Moz95] Michael C. Mozer. *A focused backpropagation algorithm for temporal pattern recognition*, pages 137–169. L. Erlbaum Associates Inc., 1995.

[OK04] P-Y. Oudeyer and F. Kaplan. Intelligent Adaptive Curiosity: a source of Self-Development. In *Proceedings of the Fourth International Workshop on Epigenetic Robotics*, 2004.

[OKH07] P-Y. Oudeyer, F. Kaplan, and V. Hafner. Intrinsic Motivation Systems for Autonomous Mental Development. *IEEE Transactions on Evolutionary Computation*, 11(2):265–286, 2007.

[RHW86] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning internal representations by error propagation*, chapter 8, pages 318–362. MIT Press, Cambridge, MA, USA, 1986.

[Rie94] Martin Riedmiller. Advanced supervised learning in multi-layer perceptrons – from backpropagation to adaptive learning algorithms. *Computer Standards & Interfaces*, 16(3):265 – 278, 1994.

[SK11] Dietmar Schreiner and Jens Knoop. iRIS - Towards a Robotic Immune System. In *In Proceedings of the Austrian Robotics Workshop 2011, UMIT - Lecture Notes in Biomedical Computer Science and Mechatronics*, pages 22–35, 2011.

[SMS09] Ryo Saegusa, Giorgio Metta, and Giulio Sandini. Active learning for multiple sensorimotor coordination based on state confidence. In *Proceedings of the 2009 IEEE/RSJ international conference on Intelligent robots and systems*, IROS'09, pages 2598–2603. IEEE Press, 2009.

[Sto09] A. Stoytchev. Some basic principles of developmental robotics. *Autonomous Mental Development, IEEE Transactions on*, 1(2):122–130, 2009.

[SWGG07] Jürgen Schmidhuber, Daan Wierstra, Matteo Gagliolo, and Faustino Gomez. Training Recurrent Networks by Evolino. *Neural Computation*, 19(3):757–779, 2007.

[ZB01] J. Zlatev and C. Balkenius. Introduction: Why epigenetic robotics? In *Proceedings of the First International Workshop on Epigenetic Robotics*, 2001.

[ZBKL10] Michael Zillich, Michael Baumann, Wolfgang Knefel, and Christoph Längauer. Corvid: A Versatile Platform for Exploring Mobile Manipulation. In Jackie Chappell, Susannah Thorpe, Nick Hawes, and Aaron Sloman, editors, *Symposium on AI-Inspired Biology (AIIB)*, 2010.

[ZPMV11] Michael Zillich, Johann Prankl, Thomas Mörwald, and Markus Vincze. Knowing your limits - self-evaluation and prediction in object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 813–820, 2011.

Curriculum Vitae

Personal Information

First name / Surname	Christian Papauschek
Address	Zeillergasse 13/28, 1170 Vienna, Austria
E-mail	christian@papauschek.at
Date of birth	January 2nd, 1985

Education

2009 - 2012	*Middle European interdisciplinary master programme in Cognitive Science* at University of Vienna. ERASMUS mobility semester at Comenius University of Bratislava.
2003 - 2009	Bachelor of Science (B.Sc.) in *Software & Information Engineering* at University of Technology Vienna.
1994 - 2003	Matura (General University Maturity) at Kollegium Kalksburg, Realgymnasium, Vienna.

Work experience

October 2010 – now	Software developer at *runIT* (IT infrastructure, business intelligence software development) Gutheil-Schoder-Gasse 8-12, 1100 Vienna
Sept. 2007 – now	Self-employed IT service provider and freelance software developer

Other projects and activities

December 2010 Speaker at the IEEE ROBIO 2010 conference in Beijing, China (see publication below)

August 2009 Attended European Forum Alpbach 2009, "Trust" (received a return scholarship)

August 2008 Attended European Forum Alpbach 2008, "Perception and decision" (received a return scholarship)

August 2007 Attended European Forum Alpbach 2007, "Emergence" (received a scholarship from Europäisches Forum Alpbach gemeinnützige Privatstiftung, Vienna)

Publications

December 2010 Christian Papauschek, Michael Zillich. Biologically inspired navigation on a mobile robot. In *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 519–524, 2010.