Learning Symbols for Hierarchical Control from Interaction: Controllability Control

Arbeit zur Erlangung des Grades
Bachelor of Science
am Fachbereich Mathematik und Informatik
der Freien Universität Berlin

von

Gregor H. W. Gebhardt

Berlin
September 2011

Freie Universität Berlin
Fachbereich Mathematik und Informatik

Institut für Informatik
Machine Learning and Robotics Lab
Prof. Dr. Marc Toussaint

Learning Symbols for Hierarchical Control from Interaction: Controllability Control

Bachelorarbeit

Autor Gregor H. W. Gebhardt

Matrikelnummer 4291126

Eingereicht im September 2011

Erstgutachter Prof. Dr. Marc Toussaint
Zweitgutachter Prof. Dr. Oliver Brock

Acknowledgements

First and foremost I am deeply grateful to my supervisor Prof. Dr. Marc Toussaint for offering this thesis to me, for his support and for the always open door for problems and questions. I also want to emphasize his great lectures on robotics and machine learning, that gave me a fundamental comprehension of the discussed subjects and highly encouraged me to further pursue this topic of computer science.

I am also very grateful to my adviser Dr. Tobias Lang, who gave me so many good advices that always brought me great steps further in writing this thesis. It has always been a pleasure to discuss a question with him.

No less, I want to thank my parents Marianne and Ulrich for their confidence in my decisions and their support that made it possible for me to focus on the study and to have a great time living in Berlin.

I also want to thank my brothers Gunnar and Volker who, besides my parents, taught me to always get to the bottom of things.

Finally, I want to thank my aunt Doris and my cousin Lena for proofreading this thesis.

Abstract

One of the most challenging tasks in robotics is the perception and compact representation of situations in unstructured, natural environments. The severity of this task lies also in the high dimensionality of these highly complex environments. The representation should for instance classify situations in a way that is relevant for action selection and allows for hierarchical decision making. The downscaling of high dimensional states to lower dimensional, potentially symbolic representations is an important step to solve planning and controlling problems within these environments. This thesis uses the concept of controllability states as symbolic representations for the states of an environment. A controllability state specifies, which degrees of freedom are controllable in a given state. The core contribution of this thesis is to show how such symbolic representations of controllability can be extracted based on the data an agent collects during the interaction with a continuous environment. The developed method is demonstrated in a 2-dimensional environment. Based on the learned symbolic representations and dynamic models associated to each symbol I derive the Controllability Control method: a controller that is able to provide a sequence of control instructions, that finally lead to a desired controllability state and hence to the controllability of one or more specific degrees of freedom. The Controllability Control model has successfully been tested for the task of moving an object, that is initially not controllable to a desired position.

Contents

Chapter 1

Introduction

One of the most challenging tasks in robotics is the perception and compact representation of situations in unstructured, natural environments. The severity of this task lies not only in the uncertainty of the informations provided by sensors, but also in the high complexity of these environments that leads to high dimensional state spaces. The representation should for instance classify situations in a way that is relevant for action selection and allows for hierarchical decision making. The downscaling of high dimensional states to lower dimensional, potentially symbolic representations is consequently an important step to solve planning and controlling problems within these environments.

One approach to the reduction of dimensionality is followed by Katz and Brock (2008), who extract kinematic models of objects in the environment by interacting with them and subsequently use those models to obtain actions, that change the kinematic state of these objects to a desired state.

Another approach to this downscaling problem is based on the symbol grounding problem that has its roots in the field of cognitive science (Harnad, 1990) as the problem of connecting words to their meanings. In robotics the symbol grounding problem is the task of mapping a specific situation to an abstract symbol that represents that situation.

The central question is what aspects of the situation such symbols should represent and how they can be learnt from experience. In this thesis I will present a symbolic abstraction of a high dimensional state that is based on a structure which is assumed to be "inherent in natural worlds" (Toussaint, 2011). The following example will give an informal description of this structure: Consider a household-robot that interacts with the human environment. Obviously this robot has many degrees of freedom which it can actuate directly, as its arms, legs and so forth. Though there are many more degrees of freedom in its environment which it can not actuate directly, but which might become manipulable, if the robot is connected to them (Toussaint, 2011).

For example, one task of the robot could be to open the front door when someone rings the bell. To accomplish that task, the robot has to move to that door, then grasp the door handle and lower it in order to finally open the door and let the guests in. (Actually the classification of the people in front of the door as guest or robbers is another problem, that has to be solved.)

In this sequence of actions, there are discrete changes of the degrees of freedom, the robot can affect:

1. At the moment it has grasped the door handle, it can lower the handle.
2. At the moment it has lowered the door handle it can open the door.

Apparently, these changes are important events in the sequence of actions the robot has to execute in order to succeed. This leads to the assumption, that the symbols obtained from this structure could be used to provide a symbolic plan of a complex operation. For the example presented above, the symbolic plan would be:

1. Obtain the ability to actuate the door handle.
2. Obtain the ability to actuate the door.

These symbolic plans could, for example, be provided by a higher level of a hierarchical controller. But they are useless without a translation to the control instructions that can affect the state of the environment in the desired way. Therefore, a control model is necessary that takes the desired symbol as input and generates a sequence of control instructions that eventually lead to a state whose representation is this desired symbol.

The aim of this thesis is to learn symbols that represent such a notion of "ability to actuate" purely from the interaction of an autonomous agent with its environment. From the learned model that maps the state of the environment to its symbolic representation, I will derive a control model that is able to steer the agent in a way that leads to a desired transition in the symbol space. I will use a 2-dimensional toy-world – the *Disc World* – to develop and evaluate the methods and concepts in order to get a proof of concept for more complex environments.

If the methods developed in this thesis could be transferred to more complex environments, the proposed symbolic representations for the high dimensional states might provide a level of abstraction that can be used for hierarchical controlling methods, as subgoals for reinforcement learning problems (McGovern and Barto, 2001) or to describe temporal extended actions (activities) for hierarchical reinforcement learning problems (Barto and Mahadevan, 2003).

Chapter 2

The Disc World

The *Disc World* forms, as a 2-dimensional toy-world for an autonomous agent, the basic scenario of this thesis. The algorithms developed in the following chapters are evaluated using the Disc World as a "benchmark world" (Toussaint, 2011) to get a proof of concept for more complex environments. The simple design of the Disc World emphasizes on the structure, that shall be exploited for symbolic representations of the Disc World state.

In this chapter I will first give a formal definition of the Disc World and describe the basic assumptions I made, regarding the physical behavior of objects. I will then describe the methods and concepts I used to implement a simulator of the Disc World.

2.1 A formal definition of the Disc World

As described by Toussaint (2011), the Disc World is a 2D environment with a finite width $w_{\text{Disc World}}$ and a finite height $h_{\text{Disc World}}$. The Disc World contains a disc shaped agent a with radius r_a and position $p_a = (x_a, y_a) \in \mathbb{R}^2$ as well as n disc shaped objects d_i with radius r_i and position $p_i = (x_i, y_i) \in \mathbb{R}^2$ with $i \in \{1, \ldots, n\}$. Let $\mathcal{W} = d_a, d_1, \ldots, d_n$ be the set of all discs in the Disc World.

The physics of the Disc World are simulated in a kinematic and quasi-static manner. Kinematic means, that velocities are applied to the discs without regarding the forces. The agent can be actuated by setting a wanted velocity $u = (\dot{x}_a, \dot{y}_a)$ as control instruction. This velocity is achieved instantly if there is no obstacle or wall that prevents the movement. The quasi-static model assumes, that the "friction between the disk and its supporting plane is large enough so that there is no motion after pushing ceases." (Agarwal et al., 1997) In other words, discs move only if they are pushed by the agent, either directly or indirectly. As a consequence there is no need for masses or friction-coefficients.

2.2 The Simulator

The simulation of the Disc World has four, successively executed core tasks, that will be described in detail in the following sections. At first a Contact Graph is built, that

stores the connections that exist between the discs in the Disc World. Second, these connections are used to propagate the velocity of the agent to the other discs in the Disc World. The third step is the detection of upcoming collisions, which is dependent on the former computed velocities. The collision detection also calculates the shortest time to collision, which needs to be taken into account by the final task – the performance of the simulation step. These four tasks are triggered by a dynamic timer, that depends on a constant frequency or the time-to-collision, computed by the collision detection.

2.2.1 Computing the Contact Graph

The contact graph is an undirected and unweighted graph, that has the set of discs as vertices and an edge for each pair of discs that is connected. Consider two discs d_i, d_j, with $i, j \in \{a, 1, \ldots, n\}$ and $i \neq j$, positioned at p_i and p_j, respectively. Then a connection between d_i and d_j exists, if the distance

$$\delta_{ij} = |p_i - p_j| \tag{2.1}$$

between the positions of the two discs is less than or equal to ρ_{ij}. Where ρ_{ij} is defined as the the sum of r_i and r_j, the radii of the discs, and a tolerance ε:

$$\rho_{ij} = r_i + r_j + \varepsilon \tag{2.2}$$

This gives the formal definition of the Contact Graph as follows: $G_{\text{con}} = (V_{\text{con}}, E_{\text{con}})$ with

$$V_{\text{con}} = \{d_i \ : \ i = a, 1, \ldots, n\} = \mathcal{W} \tag{2.3}$$

and

$$E_{\text{con}} = \{(d_i, d_j) \ : \ \delta_{ji} \leq \rho_{ij}\} \tag{2.4}$$

2.2.2 Computing the velocities

The computation of the velocities is split into two parts. The first is the forward propagation of the agents velocity to the discs connected to it – either directly or indirectly. The second part is the backward propagation of feedbacks. Feedbacks are pseudo velocities that are used to prevent discs from penetrating walls.

The forward propagation of velocities

The forward propagation of the disc velocities is a breadth first search in the contact graph with these conditions:

- The agent is used as root node.
- Only moving discs (nodes) are visited.
- A disc may be visited multiple times, e.g. if it is pushed by multiple discs.

Figure 2.1: The forward propagation of velocities. In (a) disc d_i is connected to disc d_j in point C. ϑ_{ij} is the connecting vector between d_i and d_j. v_i is the velocity of disc d_i. In (b) v_i is split into a perpendicular component $v_{i\perp}$ and a parallel component $v_{i\parallel}$ with respect to the tangent T in point C.

The transfer of the velocity of a moving disc onto one adjacent disc is discussed in the following paragraphs.

Consider two discs d_i and d_j, as shown in Figure 2.1(a), positioned at p_i and p_j, respectively. The two discs touch each other at point C. Disc d_i has the known velocity v_i. T denotes the tangent and N the normal of d_i and d_j in C. Let $\vartheta_{ij} = p_j - p_i$ be the connecting vector between d_i and d_j.

Disc d_j is only moved by disc d_i, if the latter moves in the direction of the former. This can be simply tested using the dot product of v_i and ϑ_{ij}: if $v_i \cdot \vartheta_{ij} > 0$ holds, d_i moves in the direction of d_j, otherwise they diverge.

The velocity v_i, can be decomposed into the parallel component $v_{i\parallel}$ and the perpendicular component $v_{i\perp}$ w.r.t. T, as it can be seen in Figure 2.1(b). Disc d_j is only affected by the perpendicular component, that can be computed as follows:

$$v_{i\perp} = \cos\alpha \cdot |v_i| \cdot \frac{\vartheta_{ij}}{|\vartheta_{ij}|} \tag{2.5}$$

with the angle α, which is formed by v_i and d. The cosine of α can be replaced using the dot product of v_i and ϑ_{ij}:

$$\begin{aligned} v_{i\perp} &= \frac{v_i \cdot \vartheta_{ij}}{|v_i| \cdot |\vartheta_{ij}|} \cdot |v_i| \cdot \frac{\vartheta_{ij}}{|\vartheta_{ij}|} \\ v_{i\perp} &= v_i \cdot \vartheta_{ij} \cdot \frac{\vartheta_{ij}}{|\vartheta_{ij}|^2} \end{aligned} \tag{2.6}$$

Finally, the velocity v_j of d_j is updated, by adding $v_{i\perp}$ to it.

$$v_j^{\text{new}} = v_j^{\text{old}} + v_{i\perp} \tag{2.7}$$

This update can be derived from the law of conservation of momentum, treating the masses to be equal and the speed of the pushing disc to be constant. Due to the quasi-static assumption v_j^{old} is in most cases $\vec{0}$, but there are rare situations in which a disc has more than one pusher.

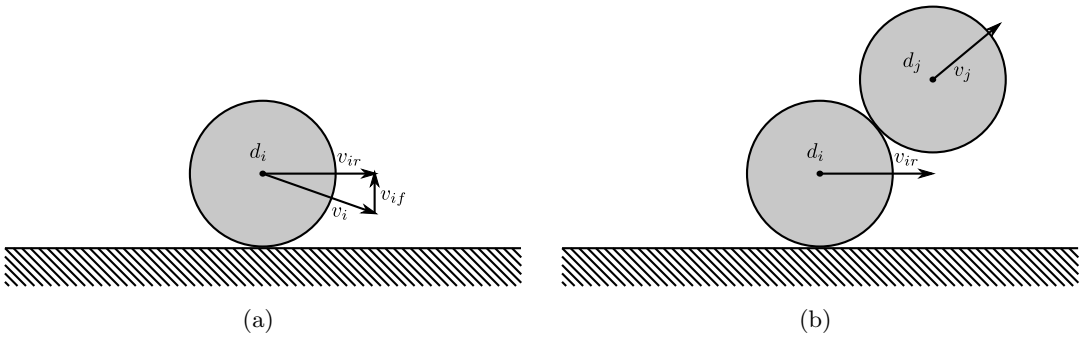

Figure 2.2: Using feedbacks as pseudo velocities to prevent disc from penetrating walls. In (a) disc d_i, with the velocity v_i, is connected to a wall. v_{if} is the feedback, that corrects v_i. The corrected velocity is v_{ir}. In (b) a disc d_j is pushed by d_i. v_j is computed using v_{ir} instead of v_i.

Feedbacks

Consider a disc d_i, connected to a wall, that has the velocity v_i as shown in Figure 2.2(a). If v_i has a perpendicular component toward the wall, the disc would penetrate it during the next step. To avoid this illegal state, a pseudo velocity $v_{i,f}$ – called feedback – is used to correct v_i. The feedback is the inverted perpendicular component of v_i:

$$v_{if} = -v_{i\perp} \tag{2.8}$$

The corrected velocity v_{ir} then is

$$v_{ir} = v_i + v_{if} = v_{i\parallel} \tag{2.9}$$

Since v_{if} is applied instantly to v_i during the forward propagation, the velocities of all subsequently processed discs (as d_j in Figure 2.2(b)) are already based on the corrected velocity v_{ir}.

The backward propagation of feedbacks

However, prior processed discs, as d_h in Figure 2.3(a), still remain unaffected of the feedback v_{if}. To solve this, the feedbacks gathered during the forward propagation of the velocities need to be back propagated to the pushing discs. This is done in a similar way as the forward propagation, by doing a breadth first search on the contact graph. Though, there are some differences:

- The root nodes are the discs, that received a feedback from a wall. (There may be more than one, but this is the same as one pseudo-root with these nodes as children.)
- Adjacent discs, that experienced an already corrected velocity are left out.

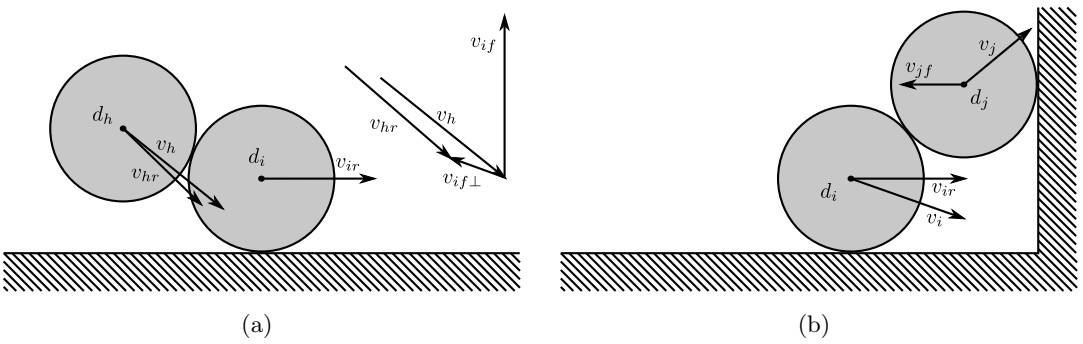

Figure 2.3: The backward propagation of feedbacks (qualitatively only). In (a) disc d_i is pushed by a disc d_h toward a wall and receives a feedback v_{if}. In the back propagation step v_h is corrected to v_{hr} using the perpendicular component $v_{if\perp}$. (b) shows a critical situation, in which two discs d_i and d_j are located near a corner. d_i pushes d_j with v_{ir}. The velocity v_j causes itself a feedback v_{jf}. This results in a loop of feedback passings from d_i to d_j and vice versa. To break this loop, the norm feedbacks needs to be greater than a certain limit in order to be passed to adjacent discs.

- Discs may be visited several times, if feedbacks come from different directions or if a feedback provokes further feedbacks itself – e.g., when discs are positioned near or in a corner, as shown in Figure 2.3(b).

The transfer of a feedback from one disc to another disc is the same as the transfer of velocities (see equations (2.6) and (2.7), replacing v_i with v_{if} and v_j with v_{jf}). If the feedback v_{if} of a disc d_i has not already been applied to v_i during the forward propagation, this is done after the feedback has been transferred to all eligible adjacent discs, by adding the feedback to the velocity:

$$v_i^{\text{new}} = v_i^{\text{old}} + v_{if} \tag{2.10}$$

The updated velocity v_i^{new} could itself provoke a further feedback, which then needs to be propagated again to adjacent discs during a further visit d_i. In critical situations, where two discs d_i and d_j are located near a corner, as shown in Figure 2.3(b), this can lead to loops of passing a feedback from d_i to d_j and vice versa. Those loops are broken by comparing the feedback against a lower limit before it is transferred to adjacent discs. If the feedback is smaller than the lower limit, it is considered negligible. This means it will neither be applied to the velocity of the disc nor will it be transferred onto adjacent discs.

2.2.3 Collision Detection

The collision detection used for the simulator does not detect if a collision happened during the last time-step. Instead it detects, if a collision will occur in the next time step and returns the exact point in time at which this impact will happen. The collision

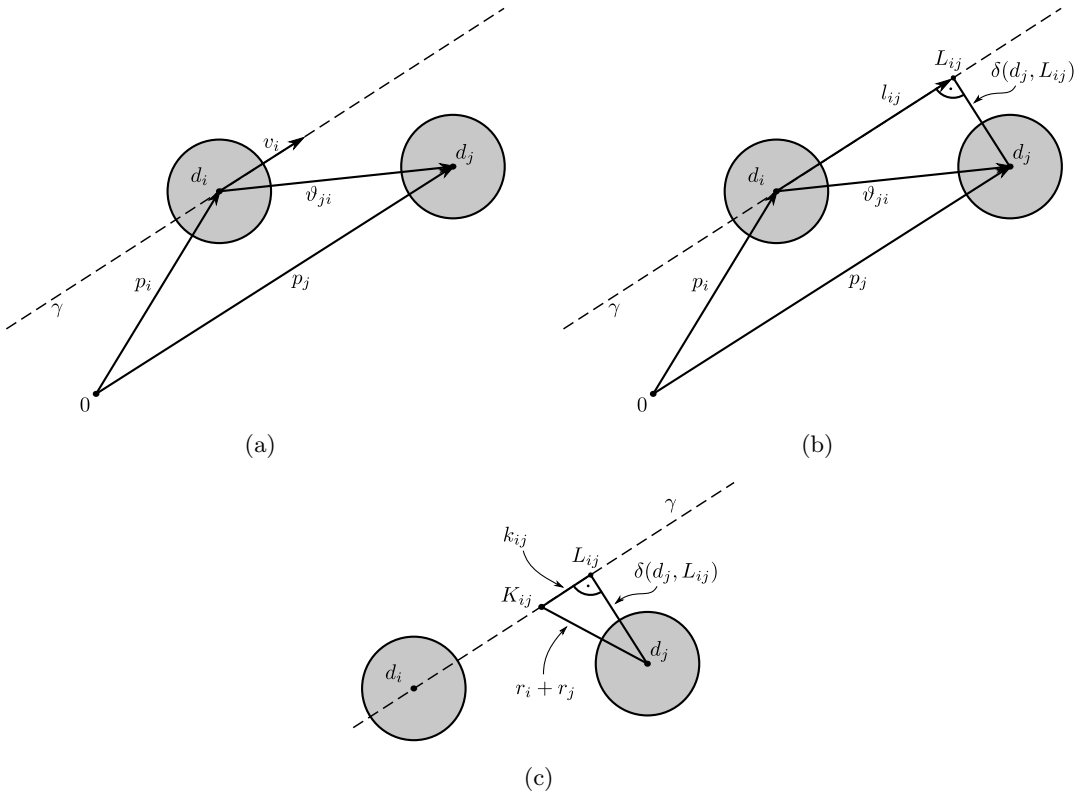

Figure 2.4: Detecting a collision of two discs. In (a) a disc d_i moves along its trajectory γ_i with velocity v_i. Disc d_j could potentially cause a collision with d_i. (b) shows L_{ij}, the closest point of γ to d_j. The distance $\delta(d_i, L_{ij})$ indicates, if a collision is possible. (c) depicts the computation of a potential collision point K_{ij}, assuming that $\delta(d_i, L_{ij})$ is less than $r_i + r_j$.

detection is separated into two parts. One for detecting collisions between two discs and one for detecting collisions of discs and walls. Both are processed for each moving disc. For simplicity, it is assumed, that the simulation time step is infinitely long.

Detecting collisions of two discs

Consider a moving disc d_i, positioned at p_i with the velocity v_i, and an idle disc d_j, positioned at p_j, as shown in Figure 2.4(a). The discs have the radii r_i and r_j and the current trajectory of d_i is

$$\gamma_i = p_i + a \cdot v_i \tag{2.11}$$

Let further be $\vartheta_{ij} = p_j - p_i$ the connecting vector between d_i and d_j.

To skip discs d_j that are behind the moving disc, regarding v_i, the dot product of ϑ_{ij} and v_i is compared to 0. If it is less than or equal to 0, a collision of d_i and d_j is impossible, since d_i does not move toward d_j. Discs d_j for which $\vartheta_{ij} \cdot v_i$ is greater than 0

can further be skipped, if the distance between γ_i and d_j is greater than or equal to the sum of the radii $r_i + r_j$. In this case d_i would pass d_j without colliding with it.

The distance between γ_i and d_j is the distance between d_j and L_{ij} – the closest point on γ_i to d_j:

$$\delta(d_j, L_{ij}) = |p_j - L_{ij}| \tag{2.12}$$

L_{ij} can be obtained using l_{ij} – the orthogonal projection of ϑ_{ij} onto v_i:

$$l_{ij} = \vartheta_{ij} \cdot v_i \cdot \frac{v_i}{|v_i|^2} \tag{2.13}$$

L_{ij} then is the sum of l_{ij} and p_i (see Figure 2.4(b)):

$$L_{ij} = p_i + l_{ij} = p_i + \vartheta_{ij} \cdot v_i \cdot \frac{v_i}{|v_i|^2} \tag{2.14}$$

If the inequality $\delta(d_j, L_{ij}) \leq r_i + r_j$ holds for d_j, then the next step is the calculation of the collision point K_{ij}. To get K_{ij} the distance k_{ij} between K_{ij} and L_{ij} is first calculated using the Pythagorean equation:

$$k_{ij} = \sqrt{(r_i + r_j)^2 - \delta(d_j, L_{ij})^2} \tag{2.15}$$

K_{ij} then is located k_{ij} times the normalized velocity v_i before L_{ij}:

$$K_{ij} = L_{ij} - k_{ij} \cdot \frac{v_i}{|v_i|} \tag{2.16}$$

Finally, using K_{ij}, the time to collision t_{ij} can be computed as follows:

$$\begin{aligned} K_{ij} &= p_i + v_i \cdot t_{ij} \\ K_{ij} - p_i &= v_i \cdot t_{ij} \\ |K_{ij} - p_i| &= |v_i| \cdot t_{ij} \\ t_{ij} &= \frac{|K_{ij} - p_i|}{|v_i|} \end{aligned} \tag{2.17}$$

Since the only collision of interest is the next collision, we have to compare t_{ij} to the current minimal time $t_{\min}$ and replace it as necessary.

Detecting collisions of a disc and a wall

Since there are only vertical and horizontal walls in the Disc World, the detection of collisions of discs with walls is a simpler task than the detection of collisions of two discs.

Consider a disc d_i located at p_i with a velocity v_i and a vertical wall w with its normal n_w, as shown in Figure 2.5(a). Let w_x be the x-value of any point on that wall. (I.e. that would be 0 if it is the left wall or the width of the Disc World, if it is the right wall.) The distance δ_{iw} between the wall and the disc can then be computed as

$$\delta_{iw} = w_x + (n_w \cdot r_i)_x - (p_i)_x \tag{2.18}$$

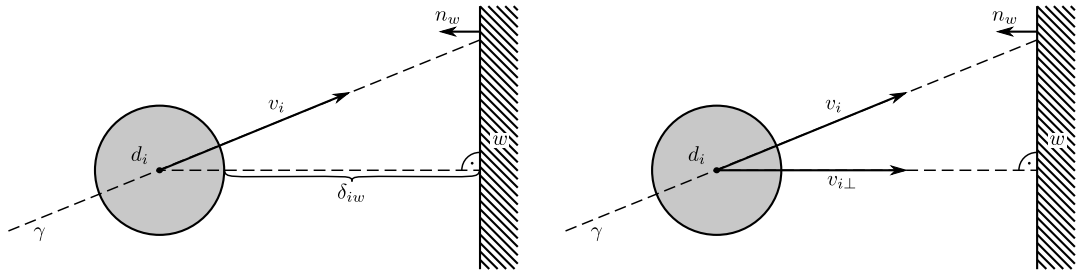

Figure 2.5: Detecting a collision of a disc and a wall. (a) shows the disc d_i that has a velocity v_i toward a wall w. δ_{iw} is the distance between d_i and w. n_w is the normal of w. (b) shows the perpendicular component $v_{i\perp}$ of v_i, with respect to w. $v_{i\perp}$ is used to calculate the time to collision of d_i and w.

where $(a)_x$ is the x-component of the vector a.

The time to collision is the quotient of δ_{iw} and the perpendicular component of v_i w.r.t. the wall (which is shown in Figure 2.5(b)):

$$t_{iw} = \frac{\delta_{iw}}{|v_{i\perp}|} = \frac{\delta_{iw}}{(v_i)_x} \tag{2.19}$$

Horizontal walls are processed analogous.

Finally, this value needs again to be compared to the current minimal time $t_{\min}$ and replaced as necessary, as in the previous section.

2.2.4 Performing a step

The last task of the simulator is the performance of the simulation step. That is moving the discs according to the prior calculated velocities. For each disc – including the agent – the position at time $t + 1$ is computed by applying the speed v_t to the position x_t, using the Euler method:

$$x_{t+1} = x_t + v_t \cdot \Delta t \tag{2.20}$$

Here $\Delta t = \min(t_{\text{default}}, t_{\text{collision}})$, the minimum of the default time step and the time step computed by the collision detection.

Chapter 3

Controllability Control

In this chapter I will give a definition of *controllability* and *controllability states* in the Disc World. The controllability states are a symbolic representation for the state of the Disc World, that can be used in higher levels of hierarchical planning mechanisms. I will then describe the *controllability control model*, a low level controller for these hierarchical planning mechanisms that executes the high level plans in the state space of the environment. But before going into detail about controllability, controllability states and controllability control, I will give a formal definition of the states and transitions of the Disc World.

3.1 Representations of States and Actions

Since the Disc World is simulated in discrete time steps, the states and transitions of the Disc World can also be described for discrete time steps.

Let s_t be the state of the Disc World at time t, which is defined as a vector containing the positions of the agent a and the discs $d_1, \ldots, d_n$:

$$s_t = (p_a, p_1, \ldots, p_n) = (x_a, y_a, x_1, y_1, \ldots, x_n, y_n) \in \mathbb{R}^{2n+2}, \tag{3.1}$$

where n is the number of discs (without the agent) in the Disc World. The single entries of s_t are denoted as $s_{t,i}$.

The actions in the Disc World are the control instructions passed to the agent. The control instruction of the agent at time t is defined as:

$$u_t = (\dot{x}_a^*, \dot{y}_a^*) \in \mathbb{R}^2. \tag{3.2}$$

Note that this is not the actual velocity of the agent but the wanted velocity set by the controller (denoted by an asterisk).

A transition of the Disc World from the state s_t at time t to the state s_{t+1} at time $t+1$ caused by the control instruction u_t at time t can be described by the triplet

$$\tau_t = (s_t, u_t, s_{t+1}) \in \mathbb{R}^{4n+6}. \tag{3.3}$$

The transitions of the Disc World for T time steps starting at time $t_0 = 1$ compose the set $\mathcal{T}$:

$$\mathcal{T} = \{\tau_t\}_{t=1}^{T} = \{(s_t, u_t, s_{t+1})\}_{t=1}^{T} \in \mathbb{R}^{T \times 4n+6} \tag{3.4}$$

$\mathcal{S}$ is the set of states s_t corresponding to the starting points of the transitions in $\mathcal{T}$:

$$\mathcal{S} = \{s_t\}_{t=1}^{T} \in \mathbb{R}^{T \times 2n+2} \tag{3.5}$$

3.2 A Definition of Controllability in the Disc World

Controllability was first defined by Kálmán (1960), as the ability to "'generate' a control function $u(x)$ (...), which causes x [the state of the system] to be 'transferred' to the equilibrium state."

The classical definition of controllability is the following: Given a system, that can be described by the equation $\dot{x} = A(x)x + B(x)u$, where x is the state of the system and u is the control instruction or input to the system. Then the system is controllable, if and only if the matrix

$$C = [B \; AB \; A^2B \; \ldots \; A^{n-1}B] \in \mathbb{R}^{n \times mn}$$

has full rank n.

Since these specifications are a more general ones, the following definition of controllability is more specific to the Disc World environment and forms the basis for the definitions in the following sections.

Definition 1: Given a state s_t at time t as defined in Equation (3.1):

$$s_t = (x_a, y_a, x_1, y_1, \ldots, x_n, y_n) \in \mathbb{R}^{2n+2},$$

that contains the positions $p_{i,t} = (x_i, y_i)$ for each disc $d_i \in \mathcal{W}$ at time t. A disc d_i in state s_t is *controllable*, if there exits a control instruction u_t at time t, that transitions the state s_t to a state s_{t+1} at time $t+1$, such that the position $p_{i,t}$ of d_i at time t is not equal to the position $p_{i,t+1}$ of d_i at time $t+1$: $p_{i,t} \neq p_{i,t+1}$.

This definition relies on the fact, that there are no external forces affecting the Disc World. That means, all movements in the Disc World arise from a control instruction to the agent.

The definition above implies also the definition of *controlling* a disc d_i as applying a control instruction, that actually changes the position p_i of d_i.

3.3 Controllability States

Toussaint (2011) defined a *controllability state (c-state)* as "the set of DoFs that are controllable". Based on the definition of controllability above, the definition of the c-state below is more specific to the Disc World and forms as well the base for the definitions in the following sections.

Definition 2: Consider a function $I : d_i, s \longmapsto c_i \in \{0, 1\}$ that specifies if the disc d_i is controllable in state s:

$$I(d, s) = \begin{cases} 1 & \text{if disc } d \text{ is controllable in state s} \\ 0 & \text{if disc } d \text{ is not controllable in state s} \end{cases} . \tag{3.6}$$

The *controllability state (c-state)* $c(s)$ of state s is the vector $(I(d_i, s))_{i=1,\ldots,n} \in \{0, 1\}^n$, with $c_i = 1$ if disc d_i is controllable or $c_i = 0$ if disc d_i is not controllable.

The agent is not regarded, since it is obviously always controllable.

Let further $C = \{c(s) : s \in S\}$ be the finite set of all possible c-states, with S the infinite set of all possible states.

3.4 Controllability Control

The idea behind controllability control is, to steer the agent "in sense of a hybrid control" (Toussaint, 2011) approach, in which the c-states provide a level of abstraction of the Disc World state for hierarchical planning. The *controllability control model* is used on a lower level to obtain control instructions which – successively executed – cause a transition of the Disc World state in a way that leads to a transition of the c-state.

Consider a state s_t at time t with its c-state $c(s_t)$ and a desired c-state c^* that is not equal to $c(s_t)$. The controllability control model

$$\psi : (s_t, c^*) \longmapsto u_t \in \mathbb{R}^2 \tag{3.7}$$

maps the state s_t and the desired c-state c^* to a control instruction u_t, that causes a transition from s_t to s_{t+1}, such that s_{t+1} is closer to any state s^* with $c(s^*) = c^*$ than s_t:

$$|s_{t+1} - s^*| < |s_t - s^*| \quad \forall \quad s^* \in \{s \mid s \in S \ \wedge \ c(s) = c^*\} . \tag{3.8}$$

To obtain a proper function for the controllability control model ψ, it can be decomposed into the following two models:

1. The *inverse c-state model* χ, that maps the state s_t at time t and the desired c-state c^* to the transition δs of the states s_t to a successive state s_{t+1}. Where δs fulfills (3.8):
$$\chi : s_t, c^* \longmapsto \delta s \tag{3.9}$$
2. The *inverse motion model* ω, that maps the transition δs and the current state s_t to a control instruction u, which – if executed in state s_t – causes δs:
$$\omega : s_t, \delta s \longmapsto u \tag{3.10}$$

A composition of the inverse c-state model χ and the inverse motion model ω then would be a solution to the controllability control model ψ:

$$\psi = \omega \circ \chi : s_t, c^* \longmapsto u_t = \omega(s_t, \chi(s_t, c^*)) \tag{3.11}$$

The functions that realize the inverse c-state model χ and the inverse motion model ω will be derived in the following sections from their non-inverse counterparts.

3.4.1 The C-State Model

The inverse c-state model χ can be derived from the *c-state model* γ, that maps a state s_t at time t to its c-state $c(s_t)$:

$$\gamma : s \longmapsto c(s) \in C. \tag{3.12}$$

As the transitions between c-states are discontinuous – either a disc is controllable or not – this function will as well be discontinuous. But assuming a gradient could be derived with respect to s as

$$G(s) = \frac{\partial}{\partial s}\ \gamma(s), \tag{3.13}$$

with

$$\begin{aligned} (\gamma(s + \delta s) - \gamma(s)) &= G(s)\ \delta s \\ \delta c &= G(s)\ \delta s \end{aligned} \tag{3.14}$$

it would be possible to get the inverse c-state model as follows:

$$\chi : s_t, c^* \longmapsto \delta s = G^\sharp(s_t)\ \delta c. \tag{3.15}$$

Where $G^\sharp(s)$ is the pseudoinverse of $G(s)$ and δc is $c^* - c(s_t)$.

A solution to the c-state model γ, that is both, continuous and differentiable, is a slightly modified c-state model that uses a separate decision function for each c-state $c \in C$ of the form

$$\gamma_c : S \longrightarrow [0, 1]. \tag{3.16}$$

Where $\gamma_c(s_t)$ shall be near or equal to 1, if c is the c-state of s_t, and near or equal to 0 otherwise. The gradient $G(s_t)$ and its inverse $G^\sharp(s_t)$ can be replaced by the gradient $G_{c^*}(s_t)$ and its inverse $G^\sharp_{c^*}(s_t)$, respectively. Where G_{c^*} is the gradient of γ_{c^*}, the decision function γ_c that belongs to the desired c-state c^*. The inverse c-state model χ then takes the form

$$\chi : s_t, c^* \longmapsto \delta s = G^\sharp_{c^*}(s_t)\ \left[1 - \gamma_{c^*}(s_t)\right]. \tag{3.17}$$

The function γ_c can be learned for each c-state separately using two-class logistic regression with ridge estimators (le Cessie and van Houwelingen, 1992):

$$\gamma_c(s) = \sigma\left(\mathbf{w}_c^\mathsf{T} \phi_\gamma(s)\right) \tag{3.18}$$

Here σ is the logistic sigmoid function

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.19}$$

and $\phi_\gamma(s)$ is the feature vector defined below. $\mathbf{w}_c$ is the learned parameter vector.

In fact, $\gamma_c(s)$ can even be seen as $P(c = c(s)|s)$ the conditional probability that c is the c-state of the given state s.

The Feature Vector

The c-state model is supposed to map a given state s to its c-state $c(s)$ that specifies, which of the discs are controllable in s. The controllability of a disc mainly depends on the following two questions:

(1) Is the disc somehow connected to the agent?

(2) Is the disc connected to a wall?

In contrast, the absolute positions of the discs are rather unimportant and are therefore not included in the feature vector. Additionally, the features have to fulfill the following two conditions:

- The feature must not use any data, that cannot be assimilated by the agent, as the c-states shall be learned solely from interaction of the agent with the Disc World.
- The feature function, that maps the state s to the feature vector has to be differentiable, in order to obtain the gradient G.

The following two feature types hold these conditions and satisfy one of the above posed questions, respectively.

The first feature type – the existence of a direct or indirect connection between a disc and the agent – can be represented by the squared distance $\delta_{i,j}(s)$ between the positions p_i and p_j for each pair of discs d_i and d_j with $i, j \in \{a, 1, \ldots, n\}$ and $i < j$:

$$\phi^{(1)}_{i,j}(s) = \delta^2_{i,j}(s) \tag{3.20}$$

with

$$\delta_{i,j}(s_t) = |p_i - p_j| = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}\,. \tag{3.21}$$

The second feature type – the connection to a wall – can be represented by the difference between an element $s_{t,i}$ of state s_t and the minimum and maximum observed value for this element, respectively:

$$\phi^{(2)}_i(s_t) = \left[\left(s_{t,i} - \min_t s_{t,i}\right), \left(\max_t s_{t,i} - s_{t,i}\right)\right] \tag{3.22}$$

where $\min_t s_{t,i}$ is the minimum observed value in $\{s_{i,t} : s_t \in \mathcal{S}\}$ and $\max_t s_{t,i}$ is the maximum observed value in $\{s_{i,t} : s_t \in \mathcal{S}\}$.

The overall feature vector $\phi_\gamma(s)$ is consequently the combination of (3.20) and (3.22) with an augmented 1 for the intercept parameter:

$$\phi_\gamma(s) = \left(\phi^{(1)}_{a,1}(s), \phi^{(1)}_{a,2}(s), \ldots, \phi^{(1)}_{n-1,n}(s), \phi^{(2)}_1(s), \ldots, \phi^{(2)}_{2n+2}(s), 1\right)^\mathsf{T} \in \mathbb{R}^H \tag{3.23}$$

The Gradient and the Inverse Gradient of γ_c

With the feature vector $\phi_\gamma(s)$ defined above, the gradient of γ_c with respect to s takes the following form:

$$G_c(s) = \frac{\partial}{\partial s}\gamma_c(s)$$

$$G_c(s) = \sigma\left(\mathbf{w}_c^\mathsf{T}\phi_\gamma(s)\right)\ \left(1 - \sigma\left(\mathbf{w}_c^\mathsf{T}\phi_\gamma(s)\right)\right)\frac{\partial}{\partial s}\phi_\gamma(s) \tag{3.24}$$

The partial derivative of $\phi_\gamma(s)$ with respect to s is described in detail in Appendix A.1. As G_c is a vector, its pseudo inverse $G_c^\sharp$ has the form

$$G_c^\sharp(s) = \frac{G_c^\mathsf{T}(s)}{G_c^\mathsf{T}(s) \cdot G_c(s)} \tag{3.25}$$

3.4.2 The Motion Model

The inverse motion model ω can be derived from the *motion model* μ_c, that maps a given state s_t and a control instruction u_t at time t to the successive state s_{t+1} at time $t+1$:

$$\mu_c : s_t, u_t \longmapsto s_{t+1}. \tag{3.26}$$

Since the motions in the Disc World are dependent on which discs are controllable in state s, this mapping is specific to the c-state $c(s)$ of state s. Thus there is a separate model μ_c for each c-state $c \in C$. The appropriate model for state s_t is chosen beforehand by the c-state model discussed in Chapter 3.4.1.

As the motions in the Disc World are continuous, there is a gradient for each motion model μ_c of the form

$$M_c(s) = \frac{\partial}{\partial u}\ \mu_c(s, u) \tag{3.27}$$

with

$$\begin{aligned}(\mu_c(s_t, u_t) - s_t) &= M_c(s_t)\ u_t \\ \delta s &= M_c(s_t)\ u_t\end{aligned} \tag{3.28}$$

With the pseudo inverse $M_c^\sharp(s)$ of the gradient $M_c(s)$, the inverse motion model ω, that maps a transition of state δs and the current state s_t to a control instruction u, as defined in Equation (3.10), becomes

$$\omega : s_t, \delta s \longmapsto u = M_{c(s_t)}^\sharp(s)\ \delta s \tag{3.29}$$

The function for each motion model μ_c can be learned as a linear mapping from the feature vector $\phi_\mu(s_t, u_t)$ – which is described below – to the successive state s_{t+1}, using linear ridge regression (Bishop, 2009; Hastie et al., 2009). The motion model then takes the form

$$\mu_c(s_t, u_t) = \mathbf{v}_c^\mathsf{T} \cdot \phi_\mu(s_t, u_t) \tag{3.30}$$

with the learned parameters $\mathbf{v}_c$. (The learning of the parameters is described in Section 4.3.2.)

The Feature Vector

The transition from a state s_t at time t, with a specific c-state $c(s_t)$, to the successive state s_{t+1} at time $t+1$ depends on s_t and the control instruction u_t. Furthermore, it also depends on the orthogonal projection of the control instruction u_t on the connecting vector $p_j - p_i$ for each pair of discs d_i, d_j with $i, j \in \{a, 1, \ldots, n\}$ and $i < j$, since this is the component of the velocity of a disc, that is transferred to the adjacent discs. These three feature types compose together with an augmented 1 for the intercept parameter the feature vector $\phi_\mu(s, u)$:

$$\phi_\mu(s_t, u_t) = \begin{pmatrix} 1 \\ s_t^\mathsf{T} \\ u_t^\mathsf{T} \\ \vartheta_{a,1}\,(u_t \cdot \vartheta_{a,1}) \\ \vartheta_{a,2}\,(u_t \cdot \vartheta_{a,2}) \\ \vdots \\ \vartheta_{n-1,n}\,(u_t \cdot \vartheta_{n-1,n}) \end{pmatrix} \in \mathbb{R}^W \tag{3.31}$$

where $\vartheta_{i,j} = p_j - p_i$ is the connecting vector between d_i and d_j, and $p_i = (x_i, y_i)^\mathsf{T}$ is the position of disc d_i.

The Gradient and the Inverse Gradient of $\mu_\mathbf{c}$

With the feature vector $\phi_\mu(s, u)$, it is now possible to derive the gradient of μ_c with respect to the control instruction u:

$$M_c(s) = \frac{\partial}{\partial u}\mu_c(s, u) \tag{3.32}$$

$$M_c(s) = \mathbf{v}_c^\mathsf{T}\frac{\partial}{\partial u}\phi_\mu(s, u) \tag{3.33}$$

The partial derivative of $\phi_\mu(s)$ is described in detail in Appendix A.2. Since M_c is a matrix, the inverse gradient $M_c^\sharp$ is:

$$M_c^\sharp(s) = \left(M_c^\mathsf{T}(s) M_c(s)\right)^{-1} M_c^\mathsf{T}(s) \tag{3.34}$$

3.4.3 The Controllability Control Model

Reconsider the controllability control model defined in (3.11):

$$\psi = \omega \circ \chi : s_t, c^* \longmapsto u_t = \omega(s_t, \chi(s_t, c^*))$$

Using the above proposed solutions to the inverse c-state model and the inverse motion model, the controllability control model takes the form:

$$\psi(s_t, c^*) = M^\sharp_{\gamma(s_t)}(s_t)\; G^\sharp_{c^*}(s_t)\; [1 - \gamma_{c^*}(s_t)] \tag{3.35}$$

Since $M^{\sharp}_{\gamma(s_t)}(s_t)$ and $G^{\sharp}_{c^*}(s_t)$ are derived from local linearizations around s_t, the control instruction u_t, that is gained by the controllability control model ψ is only applicable for small steps from s_t toward the desired state s^* with $c(s^*) = c^*$. To get these small steps, the result is scaled by a small constant factor β to an appropriate norm. This leads to the following final form of the controllability control model:

$$\psi(s_t, c^*) = \beta \; M^{\sharp}_{\gamma(s_t)}(s_t) \; G^{\sharp}_{c^*}(s_t) \; [1 - \gamma_{c^*}(s_t)] \tag{3.36}$$

3.5 Control within a Controllability State

Consider the following task: A specific disc d_i, to which the agent is not connected, shall be moved by the agent from its current position $p_{t,i}$ to a target position p_i^*. This task can be separated into two subtasks. The first subtask is to establish the controllability of d_i – which can be achieved using the controllability control model ψ described above. When the agent has reached a c-state in which d_i is controllable, the second subtask is to generate a series of control instructions for the agent such that it pushes d_i from $p_{t,i}$ to p_i^*.

For this second subtask an additional model, the ζ-model, is necessary that provides a gradient from the current state s_t to a state s^* in which the position of d_i is p_i^*. This ζ-model can be obtained from inverse kinematic control. The jacobian for disc d_i then becomes:

$$\begin{aligned} J(d_i) &= \frac{\partial}{\partial p_i} s_t \\ &= \begin{pmatrix} 0 & 0 & \cdots & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 1 & \cdots & 0 & 0 \end{pmatrix} \end{aligned} \tag{3.37}$$

where all components are 0 except for the components at $(1, 2i)$ and $(2, 2i + 1)$ which are 1. As the jacobian $J(d_i)$ provides δp_i the change of the position of d_i for a given δs. The pseudo inverse $J^{\sharp}(d_i)$ of the jacobian $J(d_i)$ will provide the transition δs that leads to the transition δp_i. Due to the simple structure of $J(d_i)$, the pseudo inverse is the transposed of the jacobian: $J^{\sharp}(d_i) = J(d_i)^{\mathsf{T}}$. The ζ-model then is defined as

$$\zeta : \delta p_i \longmapsto \delta s = J^{\sharp}(d_i) \; \delta p_i \tag{3.38}$$

Subsequently, the gradient δs can be used with the inverse motion model ω defined in (3.29) to obtain a set of control instructions u_t that leads to the desired transition δp_i of the coordinates of d_i:

$$u_t = \alpha \; \omega(s_t, \zeta(\delta p_i)) \tag{3.39}$$

with a factor alpha that scales the control instruction to a small length, as the inverse motion model ω is a local linearization around s_t.

Chapter 4

Evaluation

To evaluate the above presented models, an instance of the Disc World with a width of 600 units and a height of 500 units has been used. This Disc World contains the agent and three discs. The discs have a radius of 30 units and the agent has half the radius of 15 units. Note that the agent is not able to detach a disc from a wall, if its diameter is not smaller than the diameter of the discs. In this case the component of the agent's velocity that is transferred to an adjacent disc cannot have a perpendicular component with respect to the wall the adjacent disc is attached to that points off that wall.

To learn the c-state models γ_c and the motion models μ_c, the agent needs to explore the Disc World to experience the different c-states as well as the movements in these c-states. The data gathered in this exploration can then be used as training sets for the learning algorithms that are used for the c-state models and the motion models. The exploration methods used to obtain a data set in which all c-states are covered in a sufficient amount are described in Section 4.1.

The data set gained from the exploration of the Disc World needs to be transformed to appropriate training data sets for the c-state models and the motion models, respectively. This transformation, as well as the results of the learning process are presented in Section 4.2 for the c-state model and in Section 4.3 for the motion model.

Finally the controllability control model ψ is demonstrated with two experiments in specific scenarios. These demonstrations and their results are described in Section 4.4.

In the implementation I encoded the c-states as integers using bit i as indicator for the controllability of disc d_i. Hence the c-state $(1, 1, 0)$ becomes 011, which is 3. I will use this encoding also in the following sections to get less cluttered diagrams and equations.

4.1 Exploration of the Disc World

The exploration of the Disc World is the process of moving the agent through its environment and in doing so, assemble a data set for the learning algorithms of the c-state models γ_c and the motion models μ_c. Both of these models are trained for each c-state separately so that each c-state has to occur in an sufficient amount in the collected data set. To obtain an appropriate data set an *Exploration Controller* has been implemented

that steers the agent based on one of several heuristics. The controller and the five heuristics will be described in detail in the following sections. Note that this implementation of the Exploration Controller is specific to the evaluation setup of the Disc World with three disc besides the agent. If other setups are used, it would be necessary to adjust this implementation.

4.1.1 The Exploration Controller

The purpose of the Exploration Controller is to assemble a data set $\mathcal{T}$, containing T transitions τ_t, with $t = 1, \ldots, T$, as specified in (3.4). As noted above, in this data set all possible c-states have to be covered by a subset of data points that is large enough to act as a standalone data base for the separate learned motion models and c-state models.

To gather this data set, the controller is triggered in discrete time steps to set a new control instruction u_t for the agent and to save the current state s_t and the control instruction u_t at time t together with the state s_{t+1} at time $t + 1$ after u_t has been applied as transition τ_t in $\mathcal{T}$.

The heading of the control instruction is based on one of five different heuristics, called strategies, which will be explained in-depth in the following paragraphs. Additionally the heading is biased by a random angle that is chosen from the interval $[-\frac{1}{2}\pi, \frac{1}{2}\pi]$ in every 50th time step. Finally a constant speed is applied to the heading to obtain the control instruction that is passed to the agent. The strategy that is taken to determine the heading is randomly picked every 1500th step.

The random strategy

The first strategy is the most random of the five strategies. It is intended for the generation of complete random movements of the agent. The heading of the agent is rotated by a random angle of the interval $\left[-\frac{1}{16}\pi, \frac{1}{16}\pi\right]$ in every step. In practice, this leads mainly to situations in which the agent does not push any disc.

The follow-disc-strategy

The second strategy has the purpose to generate movements where the agent pushes at least one disc. To realize this, the agent always moves toward the position of a target disc, that is randomly chosen from the set of discs in the Disc World. A new target disc is picked every 500th step. This strategy mostly leads to situations in which the agent moves a disc along a wall and gets eventually stuck in a corner of the Disc World.

The push-one-disc-strategy

The third strategy has the purpose to produce movements where the agent a pushes a disc d_i that is positioned at p_i without the guidance of a wall. To implement this, the heading of the agent is always set to a target point T_a at the side of the disc that opposes

the center of the Disc World $c_{\text{Disc World}}$:

$$T_a = p_i + \frac{\varrho}{|p_i - c_{\text{Disc World}}|} \cdot (p_i - c_{\text{Disc World}}) \tag{4.1}$$

Where ϱ is an arbitrary factor greater than 0 and less than the sum of the radii r_a of the agent and r_i of disc d_i. The disc to be pushed is chosen every 500th step.

The push-two-discs-strategy

The fourth strategy is an extension to the push-one-disc-strategy, with the intention to generate situations in which two discs d_i, positioned at p_i, and d_j, positioned at p_j are pushed by the agent without the guidance of walls. To implement this, an additional target point T_i for for the first disc d_i is needed, which is defined analogously to the target point of the previous strategy:

$$T_i = p_j + \frac{\varrho_{i,j}}{|p_j - c_{\text{Disc World}}|} \cdot (p_j - c_{\text{Disc World}}) \tag{4.2}$$

Subsequently the target point for the agent is

$$T_a = p_i + \frac{\varrho_{a,i}}{|p_i - T_i|} \cdot (p_i - T_i) \tag{4.3}$$

Again, $\varrho_{i,j}$ and $\varrho_{a,i}$ are arbitrary factors greater than 0 and less than the sum of the radii r_i of disc d_i and r_j of disc d_j and r_a of the agent and r_i, respectively.

In the resulting situations, d_i, which is pushed by the agent, collides with d_j and subsequently d_j is pushed indirectly by the agent a. The two discs are chosen every 500th step.

The push-three-discs-strategy

The fifth strategy is a further extension of the push-two-discs-strategy, that has been implemented after the first evaluations of the data generated with only the four strategies above have shown that the c-state, in which all three discs d_j, d_j and d_k are controllable, is heavily underrepresented. To generate these situations, this strategy uses – besides the two targets T_a and T_i – a third target point T_j that is defined analogous as follows:

$$T_j = p_k + \frac{\varrho_{j,k}}{|p_k - c_{\text{Disc World}}|} \cdot (p_k - c_{\text{Disc World}}) \tag{4.4}$$

The second target point T_i then takes the form

$$T_i = p_j + \frac{\varrho_{i,j}}{|p_j - T_j|} \cdot (p_j - T_j) \tag{4.5}$$

The target for the agent T_a is the same as in Equation (4.3).

4.2 Learning and Evaluation of the C-State Model

To learn the separate c-state models γ_c, the data set $\mathcal{T}$ gathered by the Exploration Controller has to be transformed into a training data set $\mathcal{T}_{\gamma,c}$ for each c-state $c \in C$. This transformation is described in Section 4.2.1. With these training data sets $\mathcal{T}_{\gamma,c}$, the parameters $\mathbf{w}_c$ of the c-state models have been learned with two-class logistic regression with ridge estimators (see le Cessie and van Houwelingen, 1992), using the data mining tool WEKA (Hall et al., 2009). The results of this learning are presented in sections 4.2.2 and 4.2.3.

4.2.1 The Training Data

Each training set $\mathcal{T}_{\gamma,c_i}$ contains the feature vector $\phi_\gamma(s_t)$ for each state $s_t \in \mathcal{S}$ as well as a label l that specifies if c_i is the c-state of s or not. The set $\mathcal{S}$ can be obtained from the data set $\mathcal{T}$ as $\mathcal{S} = \{s_t : (s_t, u_t, s_{t+1}) \in \mathcal{T}\}$.

The label l shall be 1 if $c(s) = c_i$ or 0 otherwise. As the c-state model is supposed to be learned by the agent while it explores the Disc World, these labels have to be observed by the agent itself rather than obtained from an analytic function.

I therefore propose the *k-state*, which is the *c-state observed by the agent*, as the label l. The k-state is a vector specifying the *observed controllability* for each disc d_i in the Disc World between a state s_t at time t and its successive state s_{t+1} at time $t+1$. It is defined as

$$k(s, s') = \big(I_k(d_i, s, s')\big)_{i \in \{1,\dots,n\}} \in \mathbb{R}^n, \tag{4.6}$$

with

$$I_k(d, s, s') = \begin{cases} 1 & \text{if the position of disc } d \text{ has changed from } s \text{ to } s' \\ 0 & \text{if the position of disc } d \text{ has not changed from } s \text{ to } s' \end{cases} \tag{4.7}$$

Using the feature vector $\phi_\gamma(s)$ defined in (3.23) and as label the k-state defined in (4.6), the training data set takes the form

$$\mathcal{T}_{\gamma,c} = (\phi_\gamma(s_t), I(k(s_t, s_{t+1}) = c))_{t=1}^{T} \tag{4.8}$$

Where $I(\textit{proposition})$ is 1 if *proposition* is true and 0 otherwise.

The principle idea behind the k-state is that in state s_t the agent applies a control instruction u_t and thereby causes a transition from s_t to s_{t+1}. Afterward the agent can decide, which discs presumably have been controllable in state s.

I think this is also the way, in which humans or rather animals label their "training data". For example, if you grasp an object like a door handle for the first time, you do not know if this door is openable or not. You have to try to open it and thereafter you memorize if it actually has been openable or not.

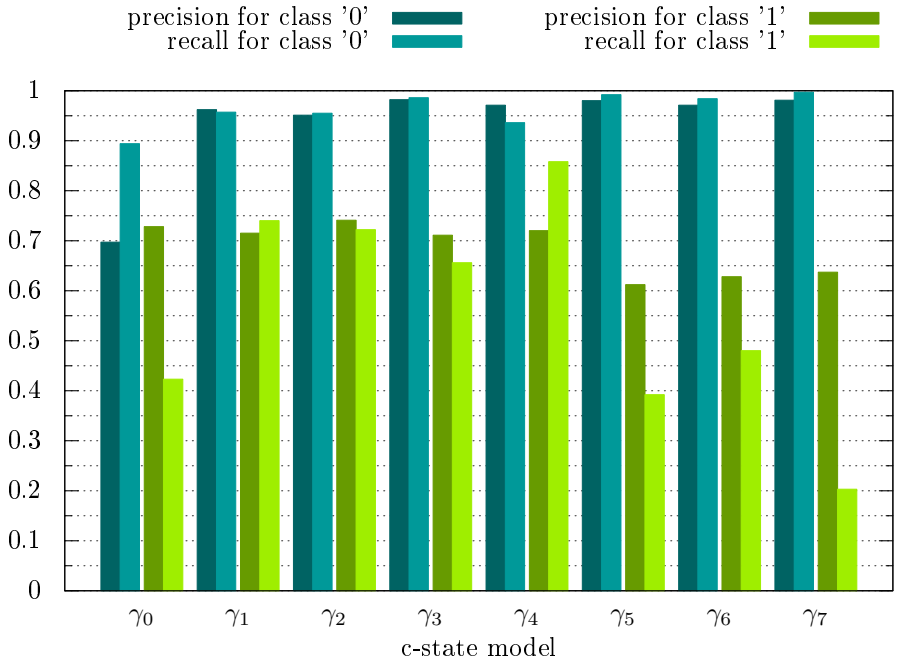

Figure 4.1: Precisions for each c-state model. The models have been learned using the original training data sets $\mathcal{T}_{\gamma,c}$. Class '1' denotes the class of states, that are in c-state c, class '0' denotes the class of states, that are not in c-state c.

4.2.2 The C-State Model as Classifier

The performances of the c-state models as classifiers for the c-state $c(s_t)$ of a given state s_t are shown in Figure 4.1 and Figure 4.2 as the precision and recall values for both classes of each trained c-state model γ_c.

The precision of a classifier for a class x is the ratio of the number instances, that are correctly classified to the total number of instances classified as x, whereas the recall value of this classifier is the ratio of the number of correctly classified instances to the number of all instances of class x.

In figures 4.1 and 4.2 class '1' denotes the class of states that are in c-state c_i and class '0' denotes the class of states that are not in c-state c_i, for all $c_i \in C$. The precision and recall values have been determined using 10-fold cross validation.

The models underlying the results shown in Figure 4.1 have been trained with the original training data sets $\mathcal{T}_{\gamma,c}$. Whereas the results shown in Figure 4.2 are obtained from models that have been learned from training data sets which are resampled with repetition from $\mathcal{T}_{\gamma,c}$. These resampled sets are equally distributed with respect to the class attribute $I(k(s_t, s_{t+1}) = c)$ (see (4.8)).

Figure 4.1 shows, that the c-state models, learned from the original training data sets $\mathcal{T}_{\gamma,c}$, only give poor results, especially for the c-state models γ_0, γ_5, γ_6 and γ_7, where the recall value for class '1' lies under 0.5, which means, that most of the instances of class '1' are not correctly classified. But also the precision value for class '1', that is between 0.6 and 0.75 for all c-states is not a good result, compared to the results of a

Figure 4.2: Precisions for each c-state model. The models have been learned using resampled equally distributed training data sets. Class '1' denotes the class of states, that are in c-state c, class '0' denotes the class of states, that are not in c-state c.

decision tree presented below.

In contrary, the c-state models learned from the resampled data have much better precision and recall values, as shown in Figure 4.2. The only exception is the c-state model γ_0, where the resampling of the training data has only little (for class '1') or even a negative effect (for class '0'). This latter effect is due to the fact, that the training data set for the c-state model γ_0 is already nearly equally distributed w.r.t. its class attribute. Additionally, I think, the classification of the c-state 0 is also the most difficult one, as there are many situations, in which a disc is stuck in a corner of the Disc World and the agent is connected to it but does not observe a controllability, when it pushes the disc toward the corner.

Summarized, the c-state models γ_c compose only a mediocre classifier for the c-state $c(s_t)$ of a given state s_t. Experiments showed, that especially states in which, the agent is near a disc d_i but not already connected to d_i are classified as a c-state in which d_i is controllable. But, as the classification of the c-state is not the primary task of the c-state model, it can be replaced easily by a classifier that gives better results. E.g., much better performances can be achieved using decision trees as shown in Table 4.1. Here a training data set with the feature vector ϕ_γ and the k-state as class attribute has been used to train a C4.5 decision tree (Quinlan, 1993) (again with the data mining tool WEKA (Hall et al., 2009)).

c-state	0	1	2	3	4	5	6	7
precision	0.997	0.990	0.993	0.992	0.993	0.988	0.987	0.978
recall	0.993	0.993	0.997	0.992	0.996	0.979	0.989	0.987

Table 4.1: The precision and recall values of a C4.5 decision tree as classifier for the c-state $c(s_t)$ of a given state s_t. The features are the same as those of c-state models. The class attribute is the k-state.

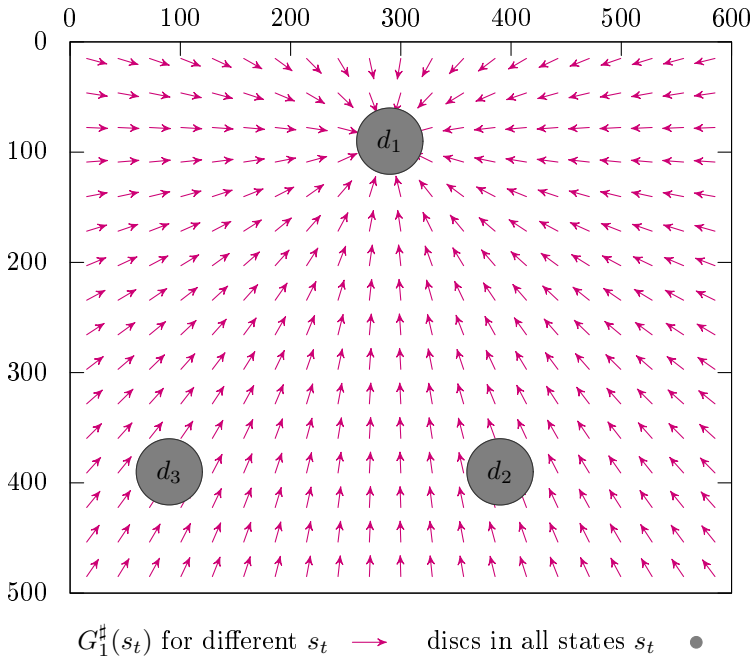

Figure 4.3: The first two elements of the gradient $G_1^\sharp(s_t)$ plotted as arrows for different states s_t. The positions of the discs d_1, d_2 and d_3 are constant, whereas the position of the agent is variable. The origins of the arrows are the different positions of the agent. The arrows are scaled to a constant length.

4.2.3 The Inverse C-State Model

The core intention of the c-state model γ is the derivation of the inverse c-state model χ. Hence, the performance of the inverse c-state model plays a major role compared to the performance of the c-state model as classifier. Given a desired c-state c^* and a state s_t at time t, the inverse c-state model χ provides a gradient $\delta s = G_{c^*}^\sharp(s_t)$ that points toward the state s^* which has the highest probability of being in the desired c-state c^*. To evaluate this gradient, its first two elements have been plotted at different states of the Disc World for different desired c-states c^*. The first two elements of the gradient are the transition of the coordinates of the agent toward the state s^*. These two elements would also be extracted as control instruction by the inverse motion model ω if the Disc World is in c-state 0.

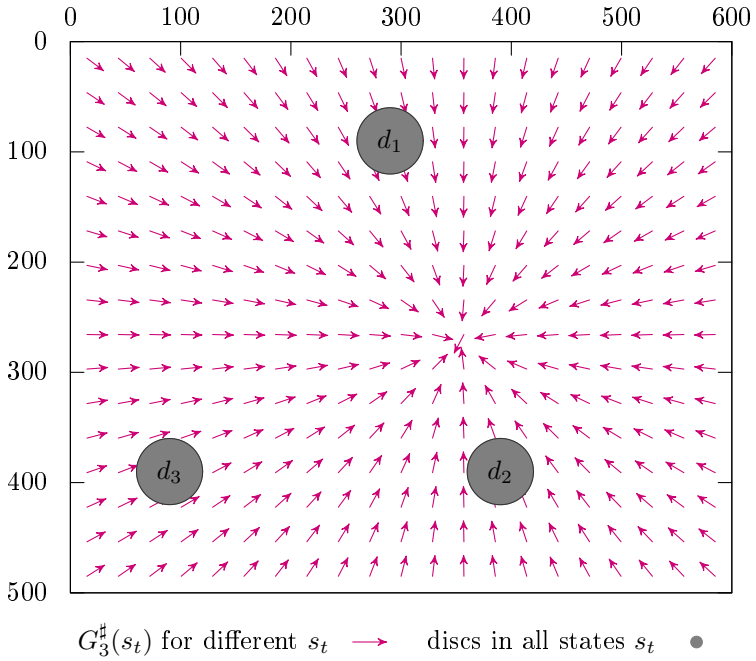

Figure 4.4: The first two elements of the gradient $G_3^\sharp(s_t)$ plotted as arrows for different states s_t. The positions of the discs d_1, d_2 and d_3 are constant, whereas the position of the agent is variable. The origins of the arrows are the different positions of the agent. The arrows are scaled to a constant length.

In Figure 4.3 and Figure 4.4 the first two elements of the gradient have been plotted for different states s_i of the Disc World. In these states s_i the position of the agent is variable, where the positions of the other discs in the Disc World are constant. In Figure 4.3 the desired c-state c^* is 1 in which only d_1 is controllable. For the c-states 2 and 4 in which only d_2 and d_3, respectively, are controllable, analogous results have been achieved. In Figure 4.4 the desired c-state c^* is 3 in which the discs d_1 and d_2 are controllable. Also, for the other c-states 5, 6, and 7 in which multiple discs are controllable, the results are comparable.

Figure 4.3 shows the expected results that would lead to a movement of the agent toward the disc d_1 and thus to a transition from the current c-state 0 to the desired c-state 1. In contrary, as Figure 4.4 shows, the Gradient for a desired c-state in which multiple discs are controllable, starting from a state s_t in which no disc is connected to another, does not lead to a movement of the agent toward one of the discs, that should be controllable, but to a movement toward some point between these discs. In consequence, the transition from, for example, c-state 0 to c-state 3, if discs d_1 and d_2 are not connected, has to be composed of two tasks. The first task is to establish the controllability of one of the two discs. The second task then is to move this disc in a way to gain finally the controllability of both discs.

Figure 4.5 shows that the inverse c-state model χ returns anyhow valid gradients for a state s_t and a desired c-state c^*, where in state s_t no disc is connected to another disc

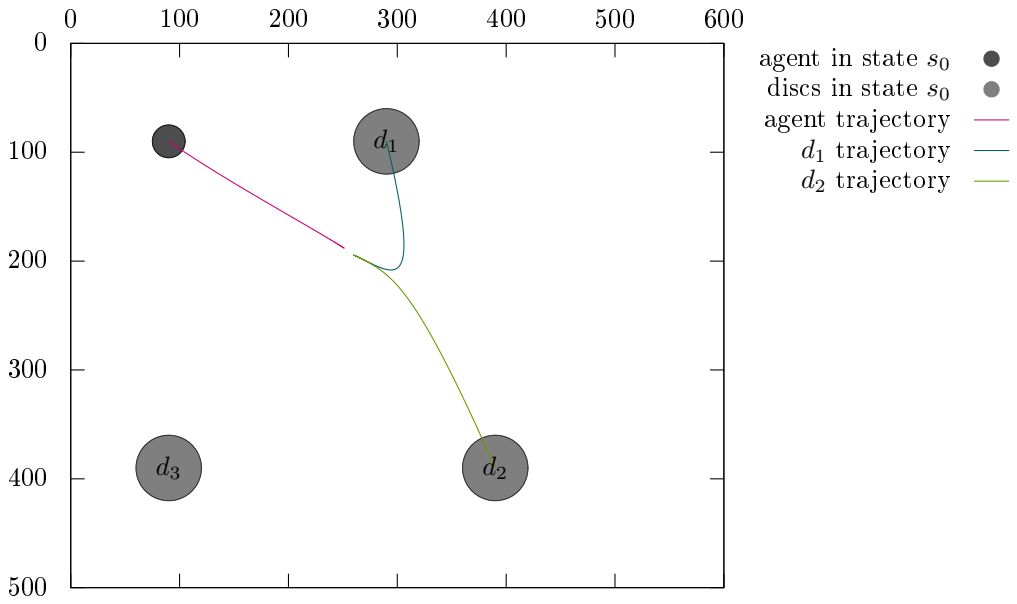

Figure 4.5: In the visualized experiment, the gradient $G_3^\sharp(s_t)$ is added to the state s_t to obtain the successive state s_{t+1} for 2000 steps. The experiment shows, that the inverse c-state model χ delivers valid gradients for situations in which two unconnected discs shall become controllable.

and in c-state c^* multiple discs are controllable. The visualized experiment starts in a initial state s_0 and runs for 2000 time steps up to state s_{2000}. Note that these states are not generated by the simulator and are also not each valid states, as there are obviously states in which discs are overlapping. The state s_{t+1} is generated by adding the the gradient $G_3^\sharp(s_t)$ to the state s_t. As it can be seen clearly, the gradient provided by the inverse c-state model χ would lead to the desired c-state c^* if all discs in the Disc World could be actuated directly.

4.3 Learning and Evaluation of the Motion Model

As for the c-state models γ_c, to learn the motion models μ_c, the data set $\mathcal{T}$ has to be transformed into separate training data sets $\mathcal{T}_{\mu,c}$ for each c-state $c \in C$. This transformation is described in Section 4.3.1. With these separate training sets $\mathcal{T}_{\mu,c}$, the parameters $\mathbf{v}_c$ for each separate motion model μ_c have been learned with linear ridge regression (Bishop, 2009; Hastie et al., 2009). The results of this learning are evaluated in Section 4.3.2.

4.3.1 The Training Data

To build the separate training data set $\mathcal{T}_{\mu,c}$, the data set $\mathcal{T}$, gathered using the Exploration Controller, is at first partitioned into $|C|$ subsets $\mathcal{T}_c$. Each subset $\mathcal{T}_c$ contains only transitions τ_t in which the discs that are controllable according to the c-state c, have been controlled by the agent:

$$\mathcal{T}_c = \{(s_t, u_t, s_{t+1}) : (s_t, u_t, s_{t+1}) \in \mathcal{T} \wedge k(s_t, s_{t+1}) = c\}\,. \tag{4.9}$$

The transitions τ_t are filtered using the k-state defined in (4.6).

Based on these separate data sets $\mathcal{T}_c$, the training data sets $\mathcal{T}_{\mu,c}$ can be composed of the feature vector for the motion model $\phi_\mu(s_t, u_t)$ and the successive state s_{t+1} for each tuple $(s_t, u_t, s_{t+1}) \in \mathcal{T}_c$:

$$\mathcal{T}_{\mu,c} = \{(\phi_\mu(s_t, u_t), s_{t+1}) : (s_t, u_t, s_{t+1}) \in \mathcal{T}_c\}\,. \tag{4.10}$$

The successive state s_{t+1} will act as the target for the ridge regression.

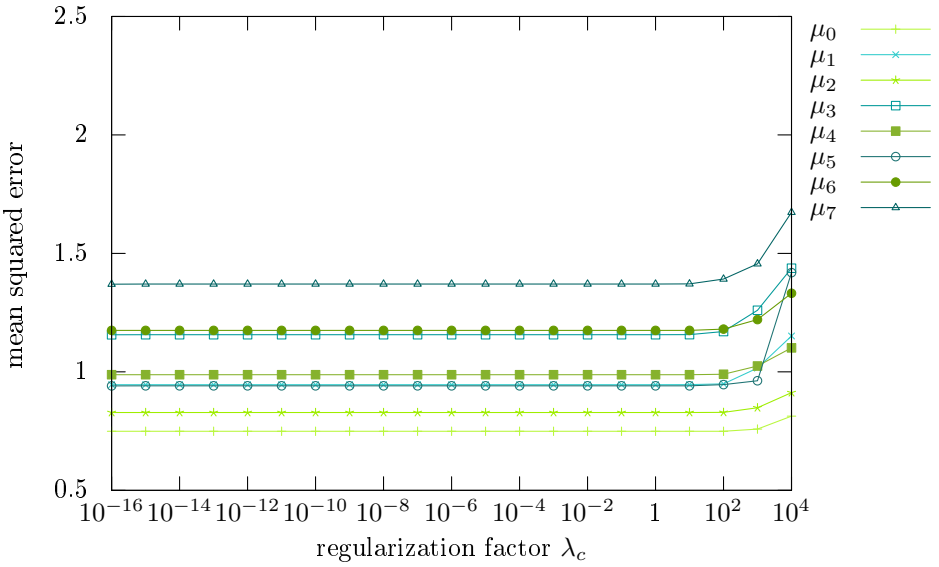

Figure 4.6: Mean squared error versus regularization factor λ_c for each motion model μ_c. The regularization factor λ_c goes from 10^{-16} to 10^4. The exponent is incremented by 1 in each measurement.

4.3.2 Learning the Motion Model

As noted above, the motion models μ_c have been learned with linear ridge regression. The parameters $\mathbf{v}_c$ then are gained by the following equation:

$$\mathbf{v}_c = (\lambda_c \mathbf{I}_M + \mathbf{\Phi}_c^\mathsf{T} \mathbf{\Phi}_c)^{-1} \mathbf{\Phi}_c^\mathsf{T} \mathbf{t}_c \tag{4.11}$$

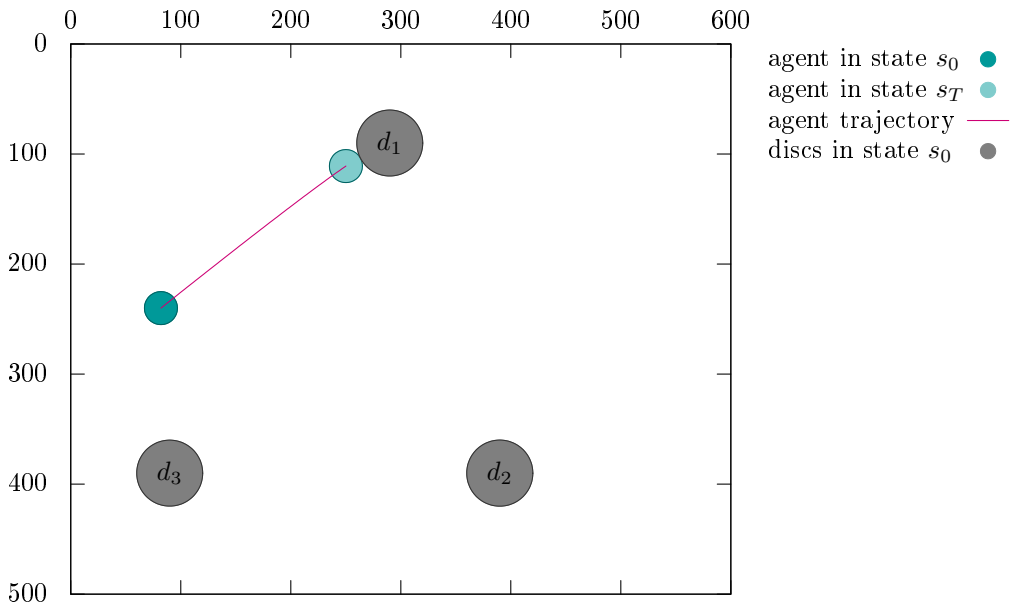

Figure 4.7: Demonstration of the controllability control model ψ. The initial c-state c_0 is 0 and the desired c-state c^* is 1. The agent has been moved toward disc d_1 using the control instructions gained from ψ. The experiment stopped after T time steps in state s_T when the desired c-state c^* has been reached.

Here λ_c is the regularization coefficient and $\mathbf{I}_M$ is the $M \times M$ identity matrix, where M is the dimensionality of the feature vector ϕ_μ. $\mathbf{\Phi}_c$ is the $|\mathcal{T}_{\mu,c}| \times M$ matrix that contains all feature vectors in $\{\phi_\mu(s_t, u_t) : (\phi_\mu(s_t, u_t), s_{t+1}) \in \mathcal{T}_{\mu,c}\}$ as row vectors and $\mathbf{t}$ is the $|\mathcal{T}_{\mu,c}| \times 2n+2$ target matrix that contains all successive states in $\{s_{t+1} : (\phi_\mu(s_t, u_t), s_{t+1}) \in \mathcal{T}_{\mu,c}\}$ also as row vectors.

Figure 4.6 shows the mean squared error versus the regularization factor λ_c for all motion models μ_c. The mean squared errors have been determined using 10-fold cross validation. It can be seen that the learning of the motion models μ_c requires no regularization, as the mean squared error remains approximately constant between $\lambda_c = 10^{-16}$ and $\lambda_c = 10$ for all models μ_c. Experiments with $\lambda_c = 0$ gave the same results.

4.4 Demonstration of the Controllability Control Model

The first demonstration of the controllability control model ψ is shown in Figure 4.7. In the initial state s_0 no disc is connected to another disc thus the initial c-state c_0 is 0. As the agent shall obtain the controllability of disc d_1, the desired c-state c^* is 1. It can be seen, that the control instructions provided by the controllability control model ψ lead to a motion of the agent toward the disc d_1. The motion is stopped after T time steps when the agent is connected to d_1 the c-state c_T is 1. This stopping criterion has not

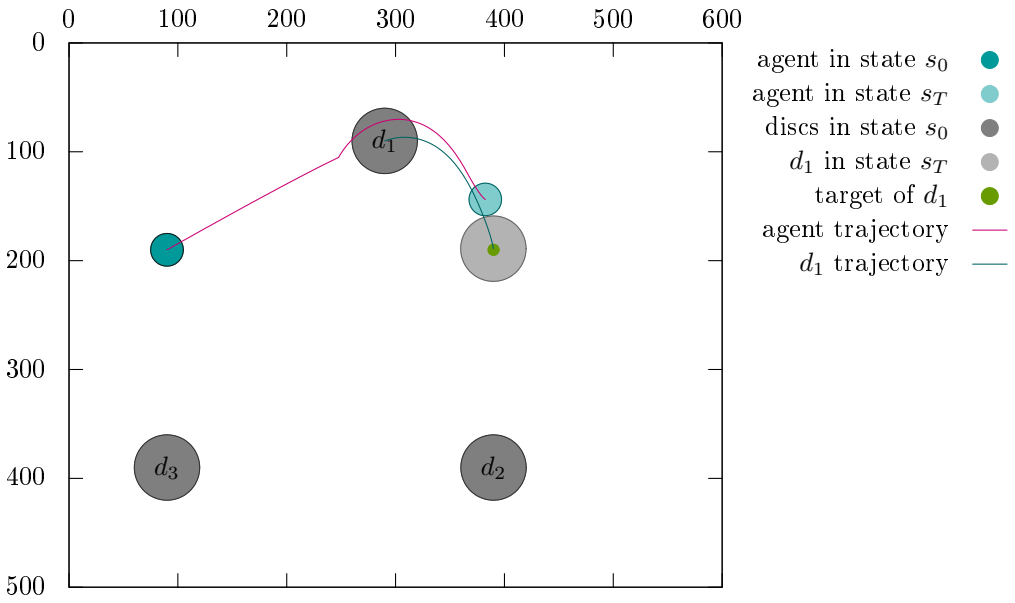

Figure 4.8: Demonstration of an appliance of the controllability control model ψ in a hierarchical control mechanism. The task is to move disc d_1 to the desired position p_1^*. A top level controller uses at first the controllability control model ψ to obtain the controllability of d_1. When d_1 is controllable, the top level controller uses the ζ-model to push d_1 toward p_1^*.

been tested with the c-state model γ but by comparing the distance between the agent and d_1 with the sum of their radii, similar to the classification of the current c-state using a decision tree (compare to Section 4.2.2).

The second experiment with the controllability control model ψ demonstrates the appliance of ψ in a hierarchical control mechanism. This experiment is adopted from the task depicted in Section 3.5: Disc d_1, to which the agent is not connected, shall be moved by the agent from its initial position $p_{0,1}$ to a desired position p_1^*. This experiment starts in the initial state s_0 and stops in the final state s_T in which d_1 reached the desired position p_1^*. As noted above, this task has to be separated into two subtasks, where the first task is to gain the controllability of disc d_1 and the second task is to push d_1 toward the target position p_1^*.

The first task equates to the experiment above and can be solved using the controllability control model ψ with the desired c-state c^*.

The second task can be solved using the gradient obtained from the ζ-model, as described in Section 3.5. The problem for the second task is, that the motion model has been learned for instantaneous motions. These motions are simulated using the Euler method with the perpendicular component of the pushing discs velocity with respect to the tangent between the pusher and the pushed disc. Consequently the learned motions are always along this perpendicular component, regardless of the agents heading. To

solve this problem, the motion model has to be learned with a different data set $\mathcal{T}$, where each transition τ_t is recorded over multiple simulation steps between the state s_t and its successive state s_{t+1}. But since the inverse motion model ω is not the principal problem of this thesis, I used a heuristic as a shortcut to map the gradients δs given by the ζ-model to a control instruction u_t. This heuristic is similar to the push-one-disc-strategy of the Exploration Controller except that the target T_a for the agent is computed using the two components of the gradient δs corresponding to d_1 as follows:

$$T_a = p_1 - \alpha \frac{\varrho}{|\delta p_1|} \cdot \delta p_1 \tag{4.12}$$

Here α is a scaling factor between r_1, the radius of d_1, and $r_1 + r_a$, the sum of the radii of d_1 and the agent. α is dependent on the distance between the current position $p_{t,1}$ and the desired position p_1^* of disc d_1.

A top level controller monitors the current state s_t of the Disc World and decides which model is used to generate the control instruction u_t. This decision is based on the current c-state c_t that is classified as in the first demonstration above, by comparing the distance between the agent and d_1 to the sum of their radii.

Figure 4.8 shows the initial state s_0 and the final state s_T of this experiment, as well as the trajectory of the agent and disc d_1 between s_0 and s_T. It can be seen, that the hierarchical control mechanism successfully controls the agent in a way that leads to a transition of d_1 from the initial position $p_{0,1}$ to the desired position p_1^*.

Chapter 5

Conclusions

The first main problem that has been addressed in this thesis is the exploitation of a controllability structure that is inherent to natural worlds in order to obtain symbolic representations for the high dimensional states of an environment. This has been achieved by learning the c-state model γ (see Section 3.4.1) using the perceptions of an agent that interacts with the 2-dimensional Disc World. The evaluations of the the inverse c-state model χ in Section 4.2.3 show that the controllability states provide a valid abstraction from the state of the Disc World.

The second main problem of this thesis was to plan on the abstract level of the controllability states and then using a control method that translates the transitions in these plans to sequences of control instructions that lead to a specific transition in the state space with the effect of the desired transition in the symbol space. This problem has been solved for the Disc World environment with the controllability control model ψ. The demonstrations in Section 4.4 show that the controllability control model is capable of generating a set of control instructions that lead to a transition of the controllability state and hence makes a specific disc controllable that can consequently be pushed to a desired position.

This thesis shows that plans on the level of controllability states in combination with the controllability control model provide a useful abstraction from the state space of the Disc World for appliances in hybrid control mechanisms. However, in future research the models have to be expanded to more sophisticated environments or evaluated for more complex tasks to obtain a better grading of the usability of the controllability structure. If the methods also succeed these problems a further objective could be their employment as activities in hierarchical reinforcement learning problems (Barto and Mahadevan, 2003) or as subgoals for reinforcement learning agents (McGovern and Barto, 2001).

Appendix A

Differentiations

A.1 The Partial Derivative of ϕ_γ

The function:

$$\phi_\gamma(s) = \left(\phi^{(1)}_{a,1}(s), \ldots, \phi^{(1)}_{n-1,n}(s), \phi^{(2)}_1(s), \ldots, \phi^{(2)}_{2n+2}(s)\right)^\mathsf{T}$$

The partial derivative with respect to s:

$$\frac{\partial}{\partial s}\phi_\gamma(s) = \begin{pmatrix} \frac{\mathrm{d}}{\mathrm{d}s_1}\phi_\gamma(s) \\ \vdots \\ \vdots \\ \frac{\mathrm{d}}{\mathrm{d}s_{2n+2}}\phi_\gamma(s) \end{pmatrix}$$

Where $\frac{\mathrm{d}}{\mathrm{d}s_i}\phi_\gamma(s)$ is:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}s_i}\phi_\gamma(s) &= \frac{\mathrm{d}}{\mathrm{d}s_i}\left(\phi^{(1)}_{a,1}(s), \ldots, \phi^{(1)}_{n-1,n}(s)\phi^{(2)}_1(s), \ldots, \phi^{(2)}_{2n+2}(s)\right) \\ &= \left(\frac{\mathrm{d}}{\mathrm{d}s_i}\phi^{(1)}_{a,1}(s), \ldots, \frac{\mathrm{d}}{\mathrm{d}s_i}\phi^{(1)}_{n-1,n}(s), \frac{\mathrm{d}}{\mathrm{d}s_i}\phi^{(2)}_1(s), \ldots, \frac{\mathrm{d}}{\mathrm{d}s_i}\phi^{(2)}_{2n+2}(s), \right) \end{aligned}$$

Where $\frac{\mathrm{d}}{\mathrm{d}s_i}\phi^{(1)}_{f,g}(s)$ is:

$$\frac{\mathrm{d}}{\mathrm{d}s_i}\phi^{(1)}_{f,g}(s) = \frac{\mathrm{d}}{\mathrm{d}s_i}\left((x_f - x_g)^2 + (y_f - y_g)^2\right)$$

$$\frac{\mathrm{d}}{\mathrm{d}s_i}\phi^{(1)}_{f,g}(s) = \begin{cases} 2 \cdot (x_f - x_g) & s_i = x_f \\ 2 \cdot (y_f - y_g) & s_i = y_f \\ -2 \cdot (x_f - x_g) & s_i = x_g \\ -2 \cdot (y_f - y_g) & s_i = y_g \\ 0 & \text{else} \end{cases}$$

And $\frac{\mathrm{d}}{\mathrm{d}s_i}\phi_j^{(2)}(s)$ is:

$$\frac{\mathrm{d}}{\mathrm{d}s_i}\phi_j^{(2)}(s) = \left(\frac{\mathrm{d}}{\mathrm{d}s_i}\left(s_j - \min_t s_j\right), \frac{\mathrm{d}}{\mathrm{d}s_i}\left(\max_t s_j - s_j\right)\right)$$

with

$$\frac{\mathrm{d}}{\mathrm{d}s_i}\left(s_j - \min_t s_j\right) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

$$\frac{\mathrm{d}}{\mathrm{d}s_i}\left(\max_t s_j - s_j\right) = \begin{cases} -1 & i = j \\ 0 & i \neq j \end{cases}$$

A.2 The Partial Derivative of ϕ_μ

The function:

$$\phi_\mu(s,u) = \begin{pmatrix} 1 \\ s^\mathsf{T} \\ u^\mathsf{T} \\ \vartheta_{a,1}\left(u \cdot \vartheta_{a,1}\right) \\ \vartheta_{a,2}\left(u \cdot \vartheta_{a,2}\right) \\ \vdots \\ \vartheta_{n-1,n}\left(u \cdot \vartheta_{n-1,n}\right) \end{pmatrix} \in \mathbb{R}^W$$

The partial derivative with respect to u:

$$\begin{aligned} \frac{\partial}{\partial u}\phi_\mu(s,u) &= \left(\frac{\mathrm{d}}{\mathrm{d}u_x}\phi_\mu(s,u), \frac{\mathrm{d}}{\mathrm{d}u_y}\phi_\mu(s,u)\right)^\mathsf{T} \\ &= \begin{pmatrix} 0 & \cdots & 0 & 1 & 0 & \frac{\mathrm{d}}{\mathrm{d}u_x}\left[\vartheta_{a,1}\left(u \cdot \vartheta_{a,1}\right)\right] & \cdots & \frac{\mathrm{d}}{\mathrm{d}u_x}\left[\vartheta_{n-1,n}\left(u \cdot \vartheta_{n-1,n}\right)\right] \\ 0 & \cdots & 0 & 0 & 1 & \frac{\mathrm{d}}{\mathrm{d}u_y}\left[\vartheta_{a,1}\left(u \cdot \vartheta_{a,1}\right)\right] & \cdots & \frac{\mathrm{d}}{\mathrm{d}u_y}\left[\vartheta_{n-1,n}\left(u \cdot \vartheta_{n-1,n}\right)\right] \end{pmatrix} \end{aligned}$$

With

$$\frac{\mathrm{d}}{\mathrm{d}u_x}\left[\vartheta_{i,j}\left(u \cdot \vartheta_{i,j}\right)\right] = \left[\left(x_j - x_i\right)^2, \left(x_j - x_i\right)\left(y_j - y_i\right)\right]$$

And

$$\frac{\mathrm{d}}{\mathrm{d}u_y}\left[\vartheta_{i,j}\left(u \cdot \vartheta_{i,j}\right)\right] = \left[\left(y_j - y_i\right)\left(x_j - x_i\right), \left(y_j - y_i\right)^2\right]$$

Literature

P. K. Agarwal, J.-C. Latombe, R. Motwani, and P. Raghavan. Nonholonomic path planning for pushing a disk among obstacles. In *Proceedings on the 1997 IEEE International Conference on Robotics and Automation*, volume 4, pages 3124–3129, 1997.

A. G. Barto and S. Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13:341–379, 2003. ISSN 0924-6703. URL `http://dx.doi.org/10.1023/A:1025696116075`. 10.1023/A:1025696116075.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer Science+Business Media, LLC, 233 Spring Street, New York, NY 10013, USA, 2009. ISBN 978-0387310732.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1), 2009.

S. Harnad. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42: 335–346, 1990.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Science+Business Media, LLC, 233 Spring Street, New York, NY 10013, USA, second edition, 2009. ISBN 978-0387848570.

R. E. Kálmán. Contributions to the theory of optimal control, 1960.

D. Katz and O. Brock. Extracting planar kinematic models using interactive perception. (8):11–23, May 2008. ISSN 1876-1100. doi: 10.1007/978-0-387-75523-6.

S. le Cessie and J. C. van Houwelingen. Ridge estimators in logistic regression. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 41(1):191–201, 1992.

A. McGovern and A. G. Barto. Automatic discovery of subgoals in reinforcement learning using diverse density. In *Proc. of the 18th Int. Conf. on Machine Learning (ICML 2001)*, pages 361–368, 2001. URL `http://www-anw.cs.umass.edu/People/barto/barto.html`.

J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufman Publishers, Inc., 2929 Campus Drive, Suite 260, San Mateo, CA 94403, USA, first edition, 1993. ISBN 1-55860-238-0.

M. Toussaint. Active controllability control in worlds with discontinuous controllability switches, 2011.

Erklärung

Hiermit bestätige ich, dass ich die vorliegende Arbeit selbständig angefertigt habe. Ich versichere, dass ich ausschließlich die angegebenen Quellen und Hilfen in Anspruch genommen habe.

Gregor Gebhardt

Berlin, 29. September 2011