Selection of Actions based on Forward Simulations

Bachelorarbeit

zur Erlangung des akademischen Grades
Bachelor of Science (B. Sc.)

HUMBOLDT-UNIVERSITÄT ZU BERLIN
MATHEMATISCH-NATURWISSENSCHAFTLICHE FAKULTÄT
INSTITUT FÜR INFORMATIK

eingereicht von: Benjamin Schlotter
geboren am: 02.07.1992
in: Dresden

Gutachter(innen): Prof. Verena V. Hafner
Prof. Hans-Dieter Burkhard

eingereicht am: ...... verteidigt am: ......

Zusammenfassung

Diese Arbeit beschreibt eine Methode zum Treffen von schnellen Entscheidungen in sich schnell verändernden Situationen. Der vorgestellte Ansatz wird im Rahmen des Entscheidungsproblems im RoboCup-Umfeld untersucht. In dieser Umgebung ist es oft schwierig, die richtige Aktion für eine Aufgabe auszuwählen. Das Ergebnis einer Aktion kann von einer Vielzahl von Umwelteinflüssen abhängen, wie zum Beispiel die Position des Roboters auf dem Feld oder die Positionen von Hindernissen. Außerdem ist die Wahrnehmung heterogen, unsicher und unvollständig. Der Ansatz ist inspiriert von der psychologischen Simulationstheorie. Hierbei werden Vorwärts-Simulationen verwendet, die dem Roboter erlauben seine Aktionen zu simulieren und die Resultate vorherzusagen. Das Ergebnis jeder Aktion wird auf Basis des geschätzten Zustands der Umgebung simuliert.

Die Simulation jeder Aktion wird in eine Reihe von simplen deterministischen Simulationen aufgespaltet, basierend auf der Unsicherheit des geschätzten Zustands und dem Aktions Modell. Jeder *Sample* wird dann separat evaluiert und anschließend addiert. Die zusammengefügten Bewertungen werden dann mit denen anderer Aktionen verglichen, um die beste Aktion zu bestimmen. Dies erlaubt es, schnell unterschiedliche Perzeptionsdaten in die Simulation einzubringen, eine stabile Entscheidung zu berechnen und die Unsicherheit der Entscheidung einzubringen. Dieser Ansatz wurde für die *Kick-Entscheidungen* im RoboCup-SPL-Umfeld implementiert und wird seit 2015 vom *Nao Team Humboldt* eingesetzt. Der Simulationsansatz wird evaluiert in einer abstrakten Simulation, isolierten Experimenten auf dem Roboter und auf Basis realer Spieldaten von RoboCup Meisterschaften.

Abstract

This thesis describes a method for making fast decisions in highly dynamic situations. This approach addresses the decision problem within the RoboCup domain. In this environment, selecting the right action is often a challenging task. The outcome of a particular action may depend on a wide variety of environmental factors, such as the robot's position on the field or the location of obstacles. Also, the perception is often heterogeneous, uncertain, and incomplete. This work is inspired by the simulation theory of cognition. This approach utilizes forward simulations which allow the robot to simulate actions and predict their consequences. The outcome of each possible action is simulated, based on the estimated state of the situation.

The simulation of a single action is split into a number of simple deterministic simulations – *samples* – based on the uncertainties of the estimated state and of the action model. Each of the samples is then evaluated separately, and the evaluations are combined and compared with those of other actions to inform the overall decision. This allows us to effectively combine heterogeneous perceptual data, calculate a stable decision, and reason about its uncertainty. This approach was implemented for the kick selection task in the RoboCup SPL environment and is actively used in competitions since 2015. This work is evaluated in an abstract simulation environment, on isolated experiments on the NAO robot platform and on real game data from the last RoboCup competitions.

Contents

Chapter 1

Introduction

Researching robotics related questions especially when it involves more than one robot tends to be expensive and time consuming. Experiments that require a high number of repetitions might not be feasible on a real robot platform due to the lack of availability, time constraints or possible wear down of the hardware. The cost of physical robots also makes platform-specific research replication particularly difficult. Those problems can be addressed by simulation. In [27] Yuan Xu defines simulation as the imitation of real systems, state of affairs, or processes. The imitation should represent certain behavioral key characteristics of a selected physical or abstract system. The RoboCup initiative's aim is to beat the human World Soccer Champion by 2050 with a fully autonomous team of robots in a football game according to official FIFA rules. When the first test for RoboCup Competitions was first held in 1996 there were no suitable robot platform available therefore a 2D Simulator was used. Over the years more soccer leagues for different kind of robots were introduced as more advanced and affordable robot platforms became available. In 2008 the NAO robot was introduced in the Standard Platform League (SPL), a league in which every team plays with the same robot. The same robot model was introduced in the 3D-Simulation League as well.

Traditionally decision making in RoboCup soccer is mainly researched in the simulation leagues. For example in [5] a ball interception behavior in the 2D simulation league is realized via a mental simulation. An overview over the recent research in simulation leagues can be found in [1]. In the SPL decision making has not received as much attention as in the Simulation leagues due to the fact that localization, ball, goal and line detection are still issues to some degree for most of the participating teams. Also dealing with the limited processing power of the NAO robots remains a challenge. Nonetheless some research exists regarding decision making in the SPL. Some examples are [19] and [15]. The transfer of simulation results to a real robot is challenging due to the lack of accurate models of the environment and robots. In his thesis Yuan Xu explores possibilities to narrow the gap between simulation and reality [27]. His results have been used in the Nao Team Humboldt for developing walking behavior, kick motions and team behavior in a simulator.

The simulation theory of cognition states that humans also simulate dynamic aspects of the environment and their perceived influence on it. This makes simulation an appropriate tool for decision making in robots as well. The simulation theory hypothesizes that thinking utilizes the same cognitive (and neural) processes as interaction with the external environment. When thinking, actions are covert and are assumed to generate, via associative brain mechanisms, the sensory inputs and elicit further actions. In this view, thinking requires building a grounded model of the environment - which is not composed of abstract symbols. Rather it is assumed to re-instantiate and recombine experiences using the brains' system of perception, action, and emotion. The mental model covertly simulates actions and their associated perceptual effects. A good introduction to the simulation theory can be found in [10]. This idea has recently gained attention in *Developmental Robotics*. For an overview see [21].

Internal forward simulation has already been successfully used as an inference method in robotics. In [4] the authors investigate navigation of robots in a dynamic environment. They use a simulation approach to envision movements of other agents and pedestrians to enable avoiding dynamic obstacles while moving towards a goal. In [13] the authors introduce a pancake baking robot which is planning its actions using a full physical simulation of the outcome of possible actions. In [7] the authors use a physics based action selection scheme to generate and select robot actions to maximize the motion of the articulated object and thus learn a better model of the object. [25] introduces Imagination-Augmented Agents to complement a RL algorithm which solves puzzle games like sokoban. To estimate the state after an action a simulation-based approach is used.

In this thesis the kick selection task is used as an example of a decision problem. In the RoboCup SPL league there have been several attempts to implement a kick selection method. In particular [6, 8] and [2] focus on a very similar task – the selection of the optimal kick. In [6] a probabilistic approach is used to describe the kick selection problem, which is then solved by using Monte Carlo simulation. In [8] the kick is chosen to maximize a proposed heuristic *game situation score* which reflects the goodness of the situation. In [2] the authors use an instance based representation for the kick actions and employ Markov decision process as an inference method. In [3] the result of kicks is predicted geometrically. The result depends on the desired distance and the kick angle. They claim that the ball will roll in the direction in which the kicking foot moves before it hits the ball. The desired distance is used to calculate parameters for the foot trajectory. For the evaluation of a kick a region is used, which is defined by the relative goal positions, a sideline offset and an opening angle of the robot which makes sure to always shoot forward. By using the localization uncertainty and the variance in kick execution an angle offset is calculated for each kick. If the ball lands in the valid region by assuming minimum and maximum angle offset respectively the kick is considered valid. If no valid action can be taken, the robot circles around the ball until a valid kick is possible.

1.1 Contribution

For this thesis the proposed method was implemented on the Nao robot platform. The method was evaluated based on video recordings and behavior logs from multiple competitions. The videos were manually synchronized to the behavior logs and labeled. The labeling tool was developed for this purpose. For further analysis the same method was reimplemented in a 2D simulator. The simulation based method improved upon the previous method.

1.2 Outline

This work describes a method for solving the kick selection problem in RoboCup using the idea of *Simulation Theory of Cognition*, a well known concept in psychology. Although the experiments were conducted using the NAO robot, this method could easily be applied to other robots in other situations as well. This approach was first introduced in this form in [17]. In Chapter 2 the mathematical concepts that are needed to understand the following chapters are presented. The following chapters explain the simulation based idea, it's implementation for the kick selection problem as described in my previous work with some improvements and the evaluation thereof. This thesis as well as the paper it is based on was written during my time working with the Nao Team Humboldt.

Chapter 2

Prerequisites

In this chapter some basic concepts are outlined which will be used in later chapters.

2.1 Potential and Gradient Fields

Potential fields are often used to realize reactive path planning approaches. They were originally proposed in [11]. The main idea is that the robot is guided by attracting and repelling forces. In the case of robot soccer attracting forces are the ball and the opponent goal and repelling forces are opponent robots. Potential fields are a very common approach to path planning in mobile robots [12, 22, 20, 16, 26].

The forces are defined by a scalar field which assigns a scalar value called potential to every point in the space. The scalar field for an attractive force is defined such that the potentials increase with increasing distance and for a repeller such that the potentials decrease with increasing distance. Let $U_{att} : \mathbb{R}^n \to \mathbb{R}$ be the scalar field of an target and $U_{rep} : \mathbb{R}^n \to \mathbb{R}$ be the scalar field of an obstacle ∇U_{att} and ∇U_{rep} are the corresponding vector or gradient fields. The vectors of the gradient fields point in the direction of the steepest ascent that means the vectors of ∇U_{att} point away from the attractor and the vectors of ∇U_{rep} point towards the repeller. Switching the sign of each vector gives the desired gradient fields $F_{att} = -\nabla U_{att}$ and $F_{rep} = -\nabla U_{rep}$ are called attractor and repeller field respectively. The resulting force F_G acting on the robot is defined as the sum of all attracting F_{att} and repulsing fields F_{rep}. The sum of all the scalar fields $\sum U$ can be imagined as a mountain range in which the obstacles are the mountains and the valleys are the targets. The robot will move downwards to the valley because of F_G. Local minima of $\sum U$ can pose a problem in this approach and lead to a deadlock since the robot will not move away from it. An example for a potential field is illustrated in Figure 2.1. Here we also see several possible paths leading towards the attractor (e.g. ball) around the repellers.

The term potential field is used for the scalar field and for the gradient field intermittently. A more detailed description of potential fields and their applications in robotics can be found in [9]. Potential fields come with some problems. The attracting and repelling forces can cancel each other out. In path planning it is assumed that the robot can change velocity and direction instantaneously. To approach these problems more elaborate planning techniques are used. By using only the potential field for path planning in RoboCup games, the following two problems may occur: modeling the goalposts and goal borders as repelling forces would "label" parts of the inside of the goal as undesirable and using only one attractor at the opponent goal gives almost equal potentials in and outside of the soccer field. Despite their problems potential fields are useful for realizing robot navigation. Potential fields are easily implemented and visualized and therefore the resulting robot behavior is easy to predict. Since the attracting and repulsing fields are independent of each other, fields can be updated, added or omitted in real time.

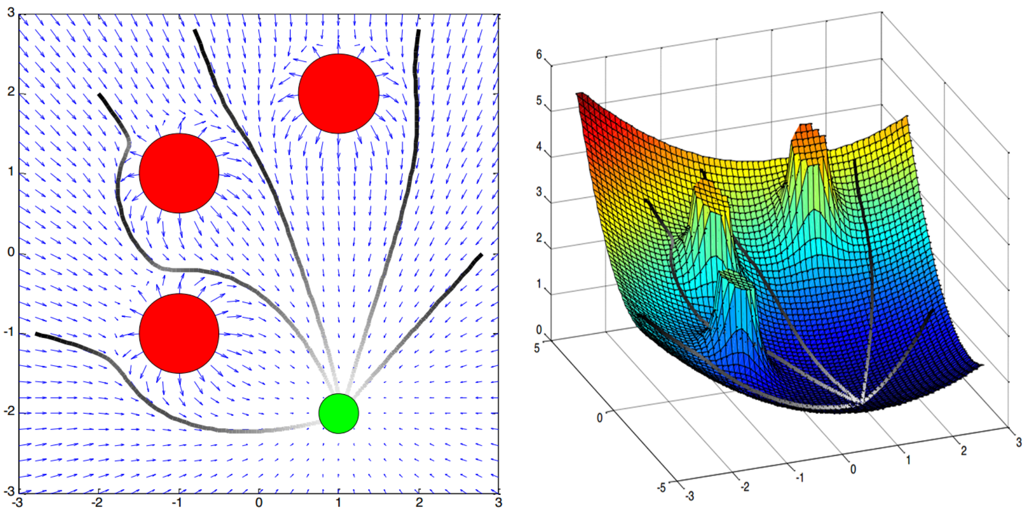


Figure 2.1: *Image was adapted from [9]. Left: a vector field with one attractor (green) and three repellers (red). Right: The corresponding scalar field*

2.2 State Estimation with Bayes Filter

Bayes filters are a general probabilistic approach for estimating an unknown probability density function. In general it calculates the next state estimate based on the previous state, actions (e.g. movements) and observations.

The notation of the equations for the Bayes filter and particle filter are influenced by the book Probabilistic Robotics [24] and the diploma thesis by Markus Scheunemann. A sequence of observations is denoted as:

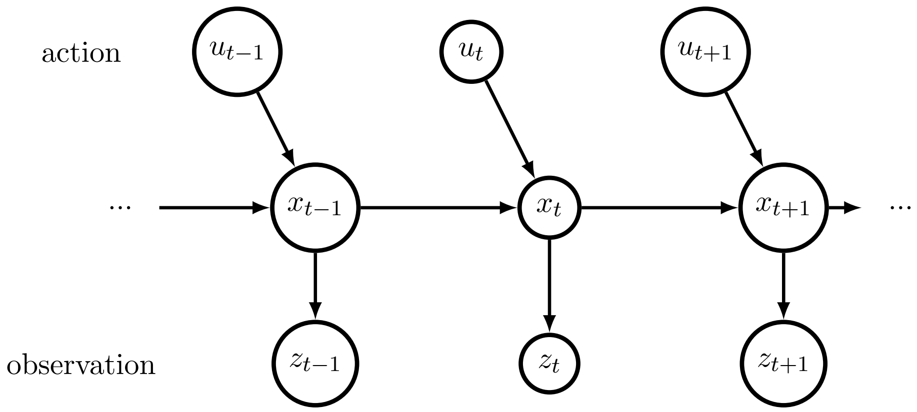


Figure 2.2: *Schematic of a Bayesian network. The modeled object is in state x_t. Executing action u_{t+1} changes the state. By accurately estimating the action the observation z_{t+1} can be predicted.*

$$z_1, ..., z_n = z_{t1}, ..., z_{t_n} \text{ with } t_u \leq t_v \text{ und u} \leq v \in \mathbb{N} \tag{2.1}$$

a sequence of actions u_t and a sequence of states x_t are written in the same fashion. It is assumed that the Markov property holds true. That means that the states x_{t-1} and $x_t + 1$ are stochastically independent. The relation between u_t, x_t and z_t is visualized in Figure 2.2. The goal is to calculate the posterior probability of state x_t at time t after observations $z_1, ...z_t$ and actions $u_1, ..., u_t$. This probability is called Bel for belief.

$$Bel_{x(t)} = P(x_t | z_1, ..., z_t, u_1, ..., u_t) \tag{2.2}$$

The belief prior to observation z_t is called $\widehat{Bel}$ and represents a state prediction. Calculating Bel from $\widehat{Bel}$ is known as sensor update. The theorem of total probability states:

$$P(x|y) = \int P(x|y, z)P(z, y)dz, \text{ with } \int P(z)dz = 1 \tag{2.3}$$

and the Bayes equation is given by:

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}, \text{ with } P(y) > 0 \tag{2.4}$$

Combining Equation (2.3) and Equation (2.4) yields:

$$P(A|B,C) \cdot P(B|C) = P(B|A,C) \cdot P(A|C) \Leftrightarrow P(A|B,C) = \frac{P(B|A,C) \cdot P(A|C)}{P(B|C)} \tag{2.5}$$

A,B and C are random variables. Substituting x_t, z_t and u_t for A, B and C results in:

$$P(\overbrace{x_t}^{A}|\overbrace{z_1,...,z_t,u_1,...,u_t}^{B,C}) = \frac{P(\overbrace{z_t}^{B}|(\overbrace{x_t}^{A},\overbrace{z_1,..,z_{t-1},u_1,..,u_t}^{C})P(\overbrace{x_t}^{A}|\overbrace{z_1,..,z_{t-1},u_1,..,u_t}^{C})}{P(\underbrace{z_t}_{B}|\underbrace{z_1,..,z_{t-1},u_1,..,u_t}_{C})} \tag{2.6}$$

$$= \frac{P(z_t|x_t,z_1,..,z_{t-1},u_1,..,u_t)P(x_t|z_1,..,z_{t-1},u_1,..,u_t)}{P(z_t|z_1,..,z_{t-1},u_1,..,u_t)} \tag{2.7}$$

$$= \frac{P(z_t|x_t)P(x_t|z_1,..,z_{t-1},u_1,..,u_t)}{P(z_t|z_1,..,z_{t-1},u_1,..,u_t)} = Bel(x_t) \tag{2.8}$$

For better readability a normalizing constant is introduced: $\eta = \frac{1}{P(z_t|z_1,..,z_{t-1},u_1,..,u_t)}$

$$Bel(x_t) = \eta \cdot P(z_t|(x_t)P(x_t|z_1,..,z_{t-1},u_1,..,u_t) \tag{2.9}$$

This equation shows that the new state is a product from the previous state and the sensor update. Using the theorem of total probability and the assumed Markov property one can write:

$$\widehat{Bel}(x_t) = P(x_t|z_1,..,z_{t-1},u_1,..,u_t) \tag{2.10}$$

$$= \int P(x_t|x_{t-1},z_{1-1},..,z_t,u_1,..,u_t)P(x_{t-1}|z_1,...,z_{t-1},u_1,...,u_t)\ \mathrm{d}x_{t-1} \tag{2.11}$$

$$= \int P(x_t|x_{t-1},u_t)P(x_{t-1}|z_1,...,z_{t-1},u_1,...,u_t)\ \mathrm{d}x_{t-1} \tag{2.12}$$

$$Bel(x_t) = \eta \cdot P(z_t|x_t) \int P(x_t|x_{t-1},u_t)P(x_{t-1}|z_1,...,z_{t-1},u_1,...,u_t)\mathrm{d}x_{t-1} \tag{2.13}$$

$$= \eta \cdot P(z_t|x_t) \int P(x_t|x_{t-1},u_t) \cdot Bel(x_{t-1}) \tag{2.14}$$

The final part of this section introduces an approximation method, the *Particle Filter*.

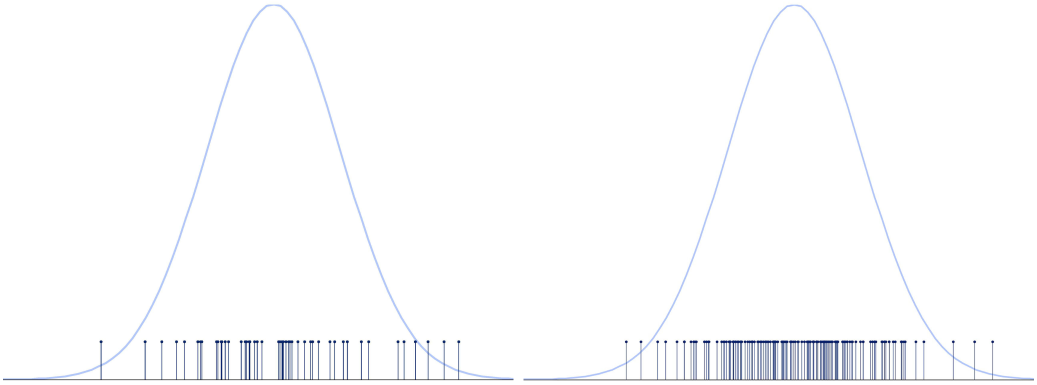


Figure 2.3: *The number of particles M for a particle filter is a crucial design parameter. The number needs to be high enough to represent the potential complex distributions of a state well enough. At the same time the sensor update is calculated for each sample, so time complexity has to be taken into account. In the left image a Gaussian is estimated by the density of 50 particle and in the right the Gaussian is estimated more accurately with 150 particles.*

2.3 Particle Filter

Particle filters are approximations of a Bayes filter. They are often used to estimate dynamic non-linear processes. In the field of robotics, particle filters are used to solve the localization problem. By using particle filters the global localization problem was first solved [23]. A particle filter approximates the posterior probability $Bel(x_t)$ at time t with the set of particles $\mathcal{S}_t$

$$\mathcal{S}_t := s_t^{[1]}, s_t^{[2]}, ..., s_t^{[M]}$$

Each particle $s_t^{[m]}$ represents a hypothesis about the state of the object (that is being subject to the filtering process). M is the number of particles. Since at each time step all the particles are updated by sensor input, the number of particles is the significant factor for computing time. For autonomous robots with limited computing power, like the NAO, it is important to choose M as low as possible but still high enough to be able to approximate the state at all. Since the probability distribution is usually unknown it is a challenge to choose M well enough. Assuming the state follows a normal distribution Figure 2.3 shows how this state can be represented with 50 particles (left) and 150 particles (right). The particle density equals the value of a Gaussian. It's clear that 150 particles represent the gauss function more accurately than 50 particles. The probability of a particle is proportional to the posterior probability of the state.

$$s_t^{[m]} \sim P(x_t|z_1, ..., z_t, u_1, ..., u_t) = Bel(x_t)$$

This equation states that it's more likely that the filter represents the true state when the density of the particles is higher. The particle filter creates the set of particles $\mathcal{S}_t$ from the previous particle set $\mathcal{S}_{t-1}$, an action and a sensor model. The algorithm 1

Algorithmus 1 : Particle Filter with *Importance Sampling*

Input : $\mathcal{S}_{t-1}$, u_t, z_t
Data : $M \leftarrow |\mathcal{S}_{t-1}|$, $\bar{\mathcal{S}} \leftarrow \emptyset$, $\mathcal{S}_t \leftarrow \emptyset$
Output : $\mathcal{S}_t$
foreach $m \in M$ **do**
    Estimation $s_t^{[m]} \sim P(s_t | u_t, s_{t-1}^{[m]})$
    $w_t^{[m]} \leftarrow P(z_t | s_t^{[m]})$
    $\bar{\mathcal{S}} \leftarrow \bar{\mathcal{S}} + \langle s_t^{[m]}, w_t^{[m]} \rangle$
end
foreach $m \in M$ **do**
    with probability $\propto w_t^{[m]}$ draw $s_t^{[m]}$ (tuple from $\bar{\mathcal{S}}$)
    add $s_t^{[m]}$ to $\mathcal{S}_t$
end

---

shows a simple version of a particle filter with Importance Sampling. First the action model is applied. After that each particle of the set $\mathcal{S}_{t-1}$ is weighted with w and is then stored in set $\bar{\mathcal{S}}$. The weighting is done based on sensor measurements z_t. In the next step particles are taken from $\bar{\mathcal{S}}$ and stored in $\mathcal{S}_t$. The chance that a particle is selected is proportional to its weight. $\mathcal{S}_t$ represents the posterior probability $Bel(x_t)$ of state x_t.

Chapter 3

Simulation Based Action Selection

In the RoboCup environment, selecting the right action is often a challenging task. The outcome of a particular action may depend on a wide variety of environmental factors, such as the robot's position on the field or the location of obstacles. In addition, the perception is often heterogeneous, uncertain, and incomplete.

Here simulation based decisions are used for the selection of actions. The intuition behind simulation based decisions is to simulate what would happen as the result of the execution of a particular action and then choose the action with the best simulated outcome.

Although there are more aspects to action selection in RoboCup e.g. positioning, the focus here is only on the selection of kicks. A core part of this chapter was already presented in [17]. The work regarding kick selection was motivated by the high number of kicks that resulted in the ball going off field using a non probabilistic method.

This non probabilistic method used a potential field which determined the direction of a kick. For positions close to the opponent goal the potential field was no longer used, but a set of rules mapped the robots position to a sequence of actions e.g.: turn towards the goal and then kick forward. The kick distance and any uncertainty was not considered. In contrast to the simulation based method only the gradients of the potential field were used.

The task of the simulation process is to predict the state of the world after the execution of a given kick. Instead of simulating the kick motion, the effect of kick motion on the ball is simulated. To be able to select the best kick, the robot needs an estimation of the world state and an action model. In case of the kick selection problem, the state consists of the robot's position on the field, the position of the ball

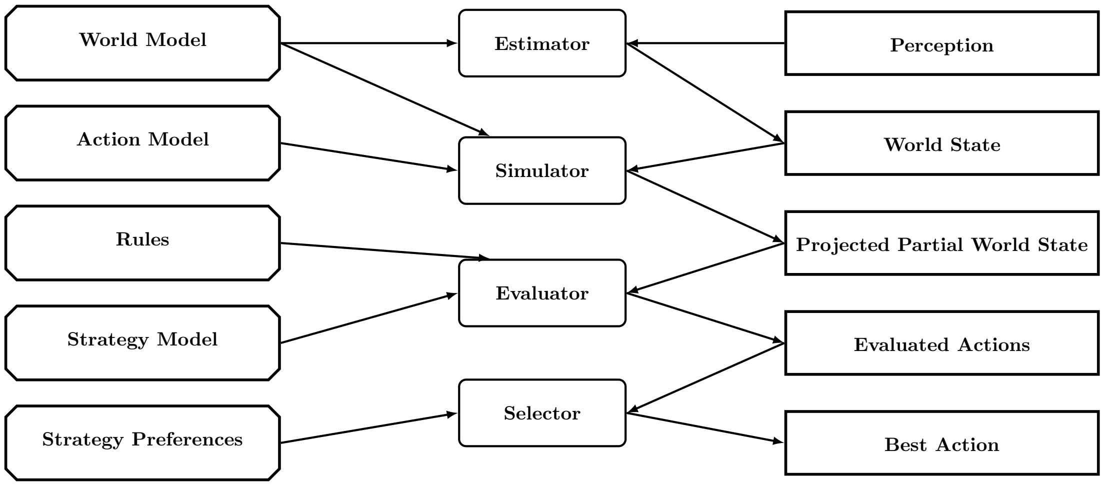


Figure 3.1: *Visualization of the simulation based decision algorithm. The left boxes are assumptions that are used at different stages in the algorithm, represented by the boxes in the middle. On the right the calculated intermediate results and input and output are shown.*

relative to the robot, positions of the teammates and opponents. These particular aspects are usually estimated using various filtering techniques. In our case different independent probabilistic filters are involved, in particular, a particle filter for self-localization and a multi-hypotheses extended Kalman filter for the ball position [18]. The action model comprises the set of possible actions and models which describe the interaction with the environment. For example a collision model. To simulate the outcome of a kick, the interactions between the executing robot and the ball, the dynamics of the ball motion and its possible interactions with the environment need to be modeled.

The structure of the simulation based decision method is visualized in Figure 3.1. The *Simulator* uses the estimated world state to envision multiple future world states for each action. The *Evaluator* evaluates the future world states based on a set of rules and a strategy model. In the case of the kick selection a rule is: the ball should stay inside the field and a strategy would be: the ball should go inside the opponent goal. The *Selector* defines the best action according to the evaluated actions and strategy preferences. Preferences are for example: If at least one projection of an action scores a goal select this action.

In the following section, these models will be discussed in detail. The Section 3.1 explains how a single projected world state is calculated. In Section 3.2 the evaluation approach based on potential fields used in [17] is presented as well as some improvements and in Section 3.3 the decision algorithm is explained.

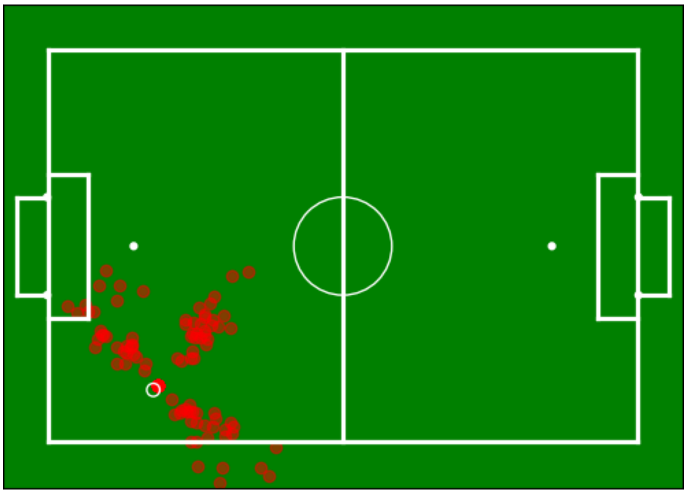


Figure 3.2: *The objective is to get the ball away from the own goal and towards the opponent goal. This objective is encoded in the potential field. From this perspective, the decision should be a sidekick to the right. But due to the uncertainty in the actions' execution, the ball might go outside the field. To deal with this uncertainty, samples are simulated*

3.1 Physical Simulation

In general, an exhaustive physical simulation is a complicated and resource consuming process. One has to carefully consider the aspects that are relevant for the simulation. To reduce complexity, several simplifying assumptions were taken. The focus is only on simulating aspects involved in the action, i.e., the ball motion and its potential collision with obstacles and goals. That means, it is assumed that all objects excluding the ball remain static. Though this is obviously not true, the velocity of the ball is usually much higher than that of the robots, which makes it a viable assumption in this case. To model collisions with obstacles, especially goal borders, a perfectly inelastic collision is assumed, where the ball's trajectory ends at the point of contact. With these assumptions, the *dynamic model of the ball* and the *model for the effect of the kick on the ball* are defined, which is discussed in the following two subsections. The simulation mentioned in this section refers to the deterministic simulation of one *sample*. In Figure 3.2 the simulation of multiple samples per action is visualized.

3.1.1 Ball Dynamics

To describe the dynamics of the ball motion, a simple *rolling resistance* model is used which leads to the following motion equation:

$$d(t) = -\frac{1}{2} \cdot g \cdot c_R \cdot t^2 + v_0 \cdot t \tag{3.1}$$

Where $d(t)$ is the distance the ball has rolled after the time $t > 0$, c_R is the rolling resistance coefficient, and v_0 is the initial velocity of the ball after the kick. By solving $d'(t) = 0$ and putting the result in eq. (3.1) the maximal rolling distance, i.e., the stopping distance of the ball, can readily be determined as

$$d_{max} = \frac{v_0^2}{2c_R \cdot g}. \tag{3.2}$$

The parameters v_0 and c_R of this model have to be determined experimentally. It should be noted that v_0 depends mainly on the particular kick motion and c_R relies primarily on the particular carpet of the field since the ball remains the same. Thus, v_0 has to be estimated once for each kick motion and c_R once for each carpet and ball pair.

Derivation of rolling resistance model

The basic friction model is:

$$F_r = c_R \cdot N \tag{3.3}$$

c_R is the roll friction coefficient. In this case the normal force N is earth's gravitational pull, so $N = g \cdot m$ with m being the mass of the ball. Newton's second law of motion says that force is equal to mass times acceleration: $F_r = a_r \cdot m$. This yields:

$$c_r \cdot N = c_R \cdot g \cdot m = a_r \cdot m \Leftrightarrow c_r \cdot g = a_r \tag{3.4}$$

a_r is constant. After the kick a_r is the only acceleration that acts upon the ball. So the acceleration of the ball for time t ≥ 0 can be written as

$$a(t) = -c_R \cdot g \tag{3.5}$$

The acceleration is negative because it acts in the opposite direction of the ball movement. This results in the following differential equation for the velocity of the ball.

$$v'(t) = a(t) = -c_R \cdot g \tag{3.6}$$

And solve it:

$$v(t) = v_0 - t \cdot c_R \cdot g \tag{3.7}$$

with initial velocity greater or equal zero. In order to calculate the distance the ball rolls $v(t)$ is interpreted as a derivative.

$$d'(t) = v(t) = v_0 - t \cdot c_R \cdot g \tag{3.8}$$

The general solution for d is:

$$d(t) = d_0 + t \cdot v_0 - \frac{1}{2} \cdot t^2 \cdot c_R \cdot g \tag{3.9}$$

The start distance d_0 is assumed to be zero since for this work only the distance the ball rolled after the kick is of any relevance.

$$d(t) = t \cdot v_0 - \frac{1}{2} \cdot t^2 \cdot c_R \cdot g \tag{3.10}$$

This equation is equal to Equation (3.1). It is important to note that this model only makes sense as long $v(t) \geq 0$. The stopping time point T can be calculated by:

$$v(T) = v_0 - T \cdot c_R \cdot g = 0 \Leftrightarrow T = \frac{v_0}{(c_R \cdot g)} \tag{3.11}$$

and therefore the distance the ball rolls is calculated as follows:

$$\begin{aligned}
d_{max} = d(T) &= T \cdot v_0 - \frac{1}{2} \cdot T^2 \cdot c_R \cdot g &(3.12)\\
&= \frac{v_0}{c_R \cdot g} \cdot v_0 - \frac{1}{2} \cdot (\frac{v_0}{c_R \cdot g})^2 \cdot c_R \cdot g &(3.13)\\
&= \frac{v_0^2}{c_R \cdot g} - \frac{1}{2} \cdot \frac{v_0^2}{c_R \cdot g} &(3.14)\\
&= \frac{1}{2} \cdot \frac{v_0^2}{c_R \cdot g} &(3.15)
\end{aligned}$$

3.1.2 Kick-Action Model

The result of a kick can be described by the likelihood of the final ball location after its execution, i.e., positions in which the ball is expected to come to a halt eventually. These positions can be estimated based on the dynamic model of the ball as described in Section 3.1.1 and the intended direction of the kick. It is assumed that the direction of the ball motion α and the initial velocity v_0 of the kick behaving according to a Gaussian distribution. With this, the outcome of a kick action can be

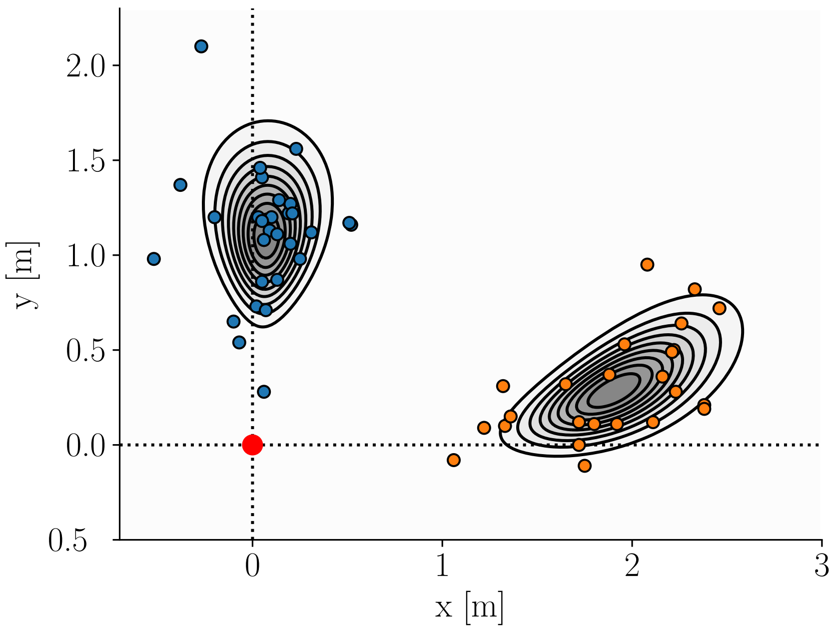


Figure 3.3: *Kick action model: distributions of the possible ball positions after a sidekick and the long kick forward with the right foot assuming a robot rotation of 0°. Blue dots illustrate experimental data.*

described as a tuple of initial velocity v_0, direction α, and corresponding standard deviations σ_v and σ_α:

$$a = (v_0, \alpha, \sigma_v, \sigma_\alpha) \in \mathbb{R}_+ \times [-\pi, \pi) \times \mathbb{R}_+ \times [-\pi, \pi) \tag{3.16}$$

The outcome of an action is predicted by sampling from the Gaussian distributions:

$$predict(a) := (d_{max}(\epsilon_v), \epsilon_\alpha) \in \mathbb{R}_+ \times [-\pi, \pi) \tag{3.17}$$

where $\epsilon_v \sim N(v, \sigma_v)$ and $\epsilon_\alpha \sim N(\alpha, \sigma_\alpha)$. Note that the function $predict(\cdot)$ is non-deterministic. Figure 3.3 illustrates the resulting likelihood for the final ball positions for a kick forward and a sidekick left. The four parameters for each kick are determined as described in Section A.1

3.1.3 Obstacle Models

In [17] a simplistic obstacle model was implemented which shortened the trajectories of all samples from the kick short action if the sonar sensor detected an obstacle in front of itself. Since the sonar measurements are too imprecise and only applicable

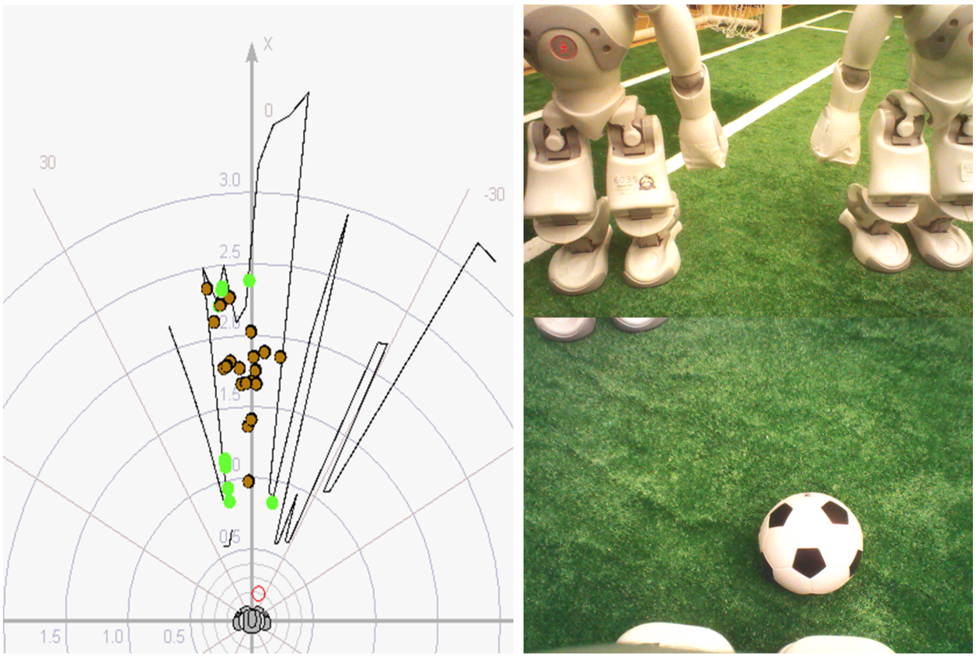


Figure 3.4: *The sample which are influenced by the opponent robots are marked green. The other samples pass between the robots. In this case the forward kick would be chosen.*

for the forward kicks, a visual approach is used. Sometimes during games, the sonar sensors detected the ground in front of the robot as an obstacle which then resulted in the wrong action. Another shortcoming of sonars is the range. A visual approach can easily deal with this.

As described in [18] a scan line algorithm is used for detecting the endpoints of the green field in the image. Those endpoints are projected on the field, and a rough border is calculated by interpolating between the projected points. If the line between the current ball position and a sample intersects the border, the sample is reset to the intersection point minus the ball radius. In Figure 3.4 illustrates an example of this method is clearly visible. Despite the two opponent robots in front of the ball, our robot would still shoot the ball forward since it's likely for the ball to go between the two robots and closer to the goal in the background. The trajectories of all samples are also shortened by collision with goal borders.

3.2 Evaluation

This step evaluates the multiple realizations of an action, called *samples*. This is referred to as simulating an action. A *hypothesis* $\mathcal{H}_a$ for the action $a \in \mathcal{A}$ is defined as a set of $n \in \mathbb{N}$ samples drawn from the model distribution of an action a as described in Section 3.1.2.

$$\mathcal{H}_a := \{p_i | p_i = predict(a), i = 1 \ldots n\} \subset \mathbb{R}_+ \times [-\pi, \pi) \tag{3.18}$$

The samples of each hypothesis are individually evaluated by two different systems. First, each sample $h \in \mathcal{H}_a$ is assigned a label

$$label(h) \in \mathcal{L} := \{\text{INFIELD}, \text{OUT}, \text{GOALOPP}, \text{GOALOWN}\} \tag{3.19}$$

based on where on the field it is, e.g., inside the field, inside the own goal, outside the field, etc. These labels reflect the corresponding discrete rules of the game. Each sample can only have one label. In the second step, all samples labeled $INFIELD$ are evaluated by a scalar potential field encoding the team strategy. The potential field used in the simulation based action selection algorithm is described closer in Section 3.2.1.

3.2.1 Potential Field

A potential field as described in Section 2.1 assigns a value to each position of the ball inside the field. The values reflect the static strategy of the game and are used to compare possible ball positions in terms of their strategic value. For instance, a position in front of the opponent goal is intuitively better than one in front of the own goal. From 2015 on the *Nao Team Humboldt* used the following potential field during competitions:

$$P(x) = \underbrace{x^T \cdot \nu_{\text{opp}}}_{\text{linear slope}} - \underbrace{N(x|\mu_{\text{opp}}, \Sigma_{\text{opp}})}_{\text{opponent goal attractor}} + \underbrace{N(x|\mu_{\text{own}}, \Sigma_{\text{own}})}_{\text{own goal repeller}}. \tag{3.20}$$

It consists of three different parts: the linear slope points from the own goal towards the opponent goal and is modeling the general direction of attack. The two exponential fields $N(x|\mu_{\text{own}}, \Sigma_{\text{own}})$ and $N(x|\mu_{\text{opp}}, \Sigma_{\text{opp}})$ create a repulsor around the own and an attractor around the opponent goal respectively. The basic function is defined as

$$N(x|\mu, \Sigma) = \exp\left[-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right]. \tag{3.21}$$

The configuration currently used for SPL Games is

$$\nu_{\text{opp}} = (-1/x_{\text{opp}}, 0)^T \tag{3.22}$$

with $x_{\text{opp}} = 4.5$ being the x-position of the opponent goal and

$$\mu_{\text{opp}} = (4.5, 0), \quad \Sigma_{\text{own}} = \begin{pmatrix} 3.375^2 & 0 \\ 0 & 1.2^2 \end{pmatrix}, \tag{3.23}$$

$$\mu_{\text{opp}} = (-4.5, 0), \quad \Sigma_{\text{opp}} = \begin{pmatrix} 2.25^2 & 0 \\ 0 & 1.2^2 \end{pmatrix} \tag{3.24}$$

for the repeller and attractor respectively. All parameters are of unit m. The potential field described in equations 3.20 to 3.24 is visualized in figure 3.5. The lines indicate points with the same potential value.

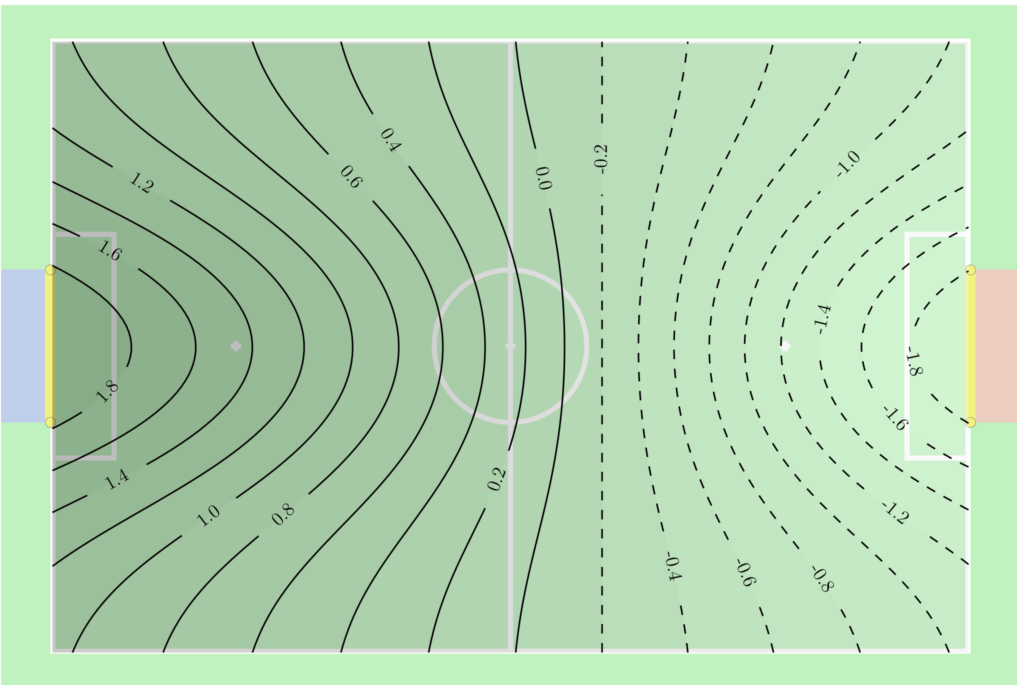


Figure 3.5: *Strategic potential field evaluating ball positions. Own goal is on the left (blue).*

3.2.2 Grounding the Potential Field

The potential field used in [17] was based on a heuristic such that positions close to the own goal get high values and close positions to the opponent goal get low values.

Thus the robot would move the ball from its own goal more towards the sidelines, while being in the opponent half, it would bring the ball back into the middle of the field. This strategy is known in football as wing play. The potential field function is mentioned in Section 3.2.1. The potential field has been observed to perform poorly in certain game situations. For example next to an opponent goal post. I propose to substitute this field with one learned using reinforcement learning methods.

Although the handcrafted version of the potential field implemented the desired strategy, it lacked grounding. Evaluating a position on the empty field according to the time that it takes to score a goal from that position results in a similar potential field. The time has to be calculated for all possible robot rotations for a given position. The minimum of those times is the potential and the robot-rotation that lead to the minimum time defines the corresponding vector. This is visualized in the left image in Figure 3.6. Mirroring the potential field and subtracting it from the original yields the image on the right. Thus the grounded potential field can be expressed as:

$$P_{(x,y)} = T_{goal}(x, y) - T_{goal}(-x, y) \tag{3.25}$$

where $T_{goal}(x, y)$ refers to the time the robot takes to score a goal from position x,y.

The grounded potential field was calculated in an abstract 2D Simulator developed for this purpose. For calculating the times we assumed the robot walks forward with a velocity of 200mm per second and a rotation velocity of 60°per second. It is further assumed that the robot always rotates towards the ball and then walks a straight line towards it and immediately performs the best action, which doesn't take any time unless it's another rotation. The future ball position is assumed to be the mean of the distribution of the best kick. The time to score a goal is now

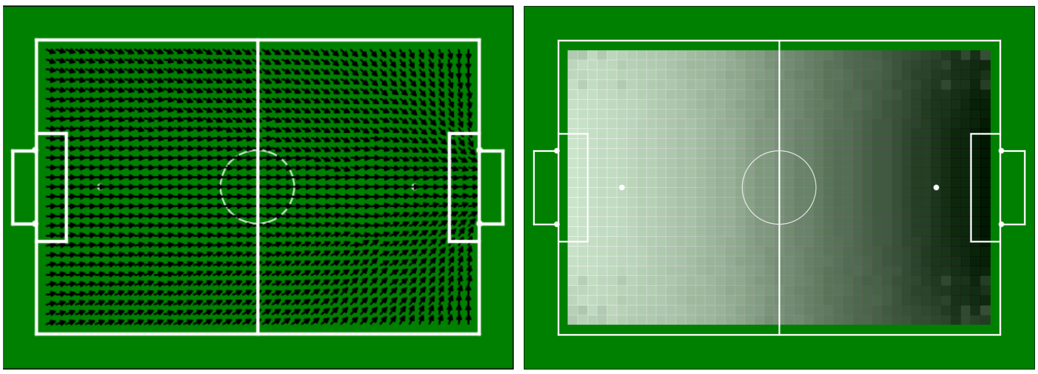


Figure 3.6: *Left: Visualization of best direction as defined as the shortest time to goal. Right: Grounded potential field as defined by equation 3.25*

the potential at the starting point. This has the benefit that dynamic aspects of the game, e.g., team members or opponent players can be more naturally represented in the potential field by modeling their effects on the time to score a goal. An example is described in Section 3.2.3.

3.2.3 Region of Influence

In Section 3.2.2 the concept of a potential field was grounded by treating the time it takes from one position to score a goal as the potential of this position. Now dynamic aspects can be encoded in the potential field by modeling the effects on the time to score a goal. The dynamic objects in a robot soccer game are mainly the robots themselves. The human referees are not considered here. Each robot is modeled with an influence region which represents the space in which it has an influence on the ball's total time to the goal. The influence region is modeled as the set of positions that can be reached in less than max_reach seconds. In local coordinates x,y this can be represented as

$$time = \frac{arctan(y,x)}{rot\_vel} + \frac{\sqrt{x^2+y^2}}{walk\_vel} \tag{3.26}$$

The influence potential $infl_p$ can be calculated as follows:

$$infl_p = \begin{cases} time - max\_reach & time < max\_reach \\ 0 & time \geq max\_reach \end{cases} \tag{3.27}$$

To get the final potential field for each team member the corresponding influence region is added to the existing field. The resulting potential field is depicted in Figure 3.7

3.3 Decision

The overall decision has to take into account the strategy preferences, meaning the trade-off between possible risks, e.g., ball leaving the field, and possible gains, e.g., scoring a goal, weighted by the chances of their occurrence. The estimation of those risks and gains can be done based on the individual ratings of the particular simulation results, i.e., samples. The likelihood of the occurrence of an event marked by a label $\lambda \in \mathcal{L}$ within a hypothesis $\mathcal{H}_a$ can be estimated as

$$p(\lambda|a) := \frac{|\{h \in \mathcal{H}_a | label(h) = \lambda\}|}{|\mathcal{H}_a|}. \tag{3.28}$$

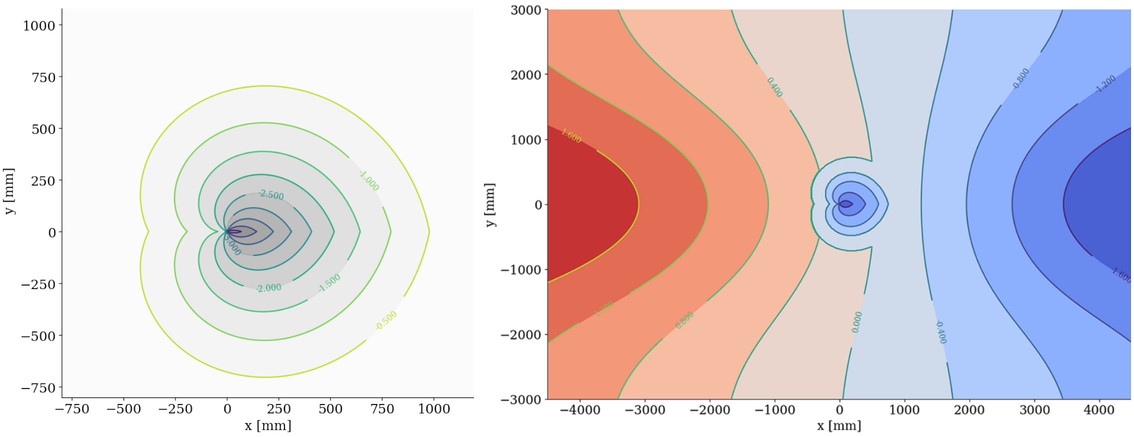


Figure 3.7: *The left image shows the influence region of a robot in local coordinates. The right image shows the applied influence region of a robot standing in the middle of the field.*

For instance, the likelihood for scoring a goal with the action a can be written as $p(\text{GOALOPP}|a)$. A two step decision process is used, whereby the actions that are *too risky* are discarded in the first step, and the one with the highest gain is selected in the second. More precisely, an action is called *too risky* if there is a high chance for kicking the ball out of the field or scoring own goal. The set of actions with acceptable risk can be defined as:

$$\mathcal{A}_{acc} := \{a \in \mathcal{A} | p(\text{INFIELD} \cup \text{GOALOPP}|a) \geq T_0 \land p(\text{GOALOWN}|a) \leq T_1\} \quad (3.29)$$

with fixed thresholds T_0 and T_1 (in our experiments $T_0 = 0.85$ and $T_1 = 0$ were used). From this set the actions with the highest likelihood of scoring a goal are selected

$$\mathcal{A}_{goal} := \operatorname{argmax} \{p(\text{GOALOPP}|a) | a \in \mathcal{A}_{acc}\} . \quad (3.30)$$

In case that $\mathcal{A}_{acc}$ is empty the default action is always to turn around the ball. In case $\mathcal{A}_{goal}$ contains more than one possible action, the best action is selected randomly from the set of actions with the maximal strategic value based on the potential field

$$a_0 \in \operatorname{argmax}\{value(a) | a \in \mathcal{A}_{goal}\} \quad (3.31)$$

with strategic values defined as

$$value(a) := \int_{\Omega} p(x|a) \cdot V(x) \, \mathrm{d}x = \frac{1}{n} \sum_{i=0}^{n} V(x_i) \quad (3.32)$$

where V is the strategic value of a position given by the potential field. In case that $\mathcal{A}_{goal}$ is empty the action with the maximum strategic value is selected from

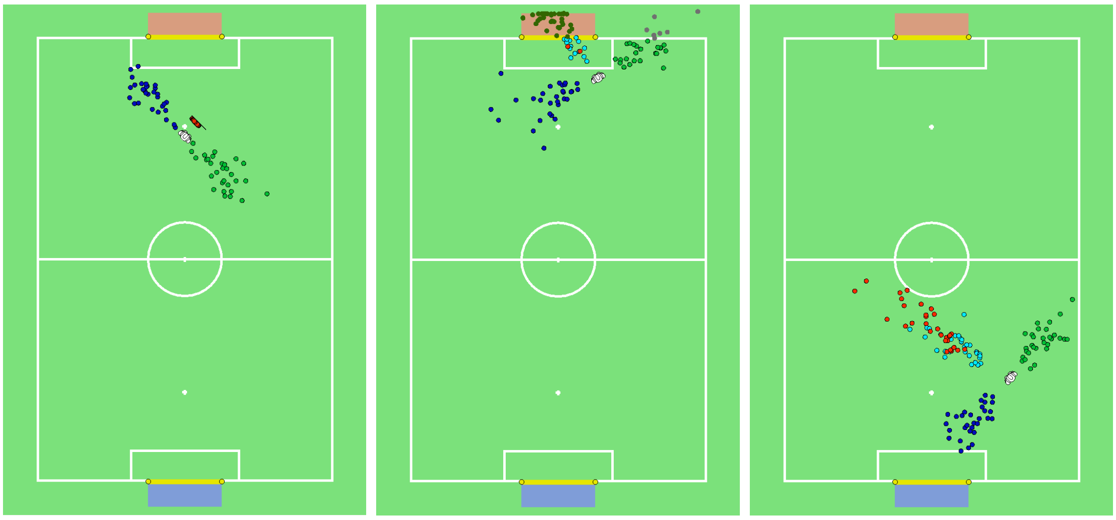


Figure 3.8: *Three examples for kick simulations. Each possible kick direction is simulated with 30 samples (different colors correspond to different kicks). Left: the short and long kicks are shortened due to collision with an obstacle. Middle: long kick is selected as the best action since it has the most samples result in a goal. Right: the best action is sidekick to the right – the other kicks are more likely to end up in a dangerous position for the own goal according to the potential field.*

the set of acceptable actions. The evaluation process is described by algorithm 2. Figure 3.8 illustrates several situations with the corresponding simulated hypotheses and their evaluations.

3.3.1 Turn Strategy

In [17] 4 different kicks were evaluated and the best one executed, except when the current position of the ball got a better potential value than the estimated future position of all other kicks. In this case the robot would choose to turn around the ball. This was motivated by the need to keep the ball moving and therefore making it harder for the opponent to have enough time to prepare for a long kick. However, in some situations it would be beneficial to turn around the ball to some degree and then execute a kick. With the simulation based decision algorithm described previously such a decision could not be made. A naive solution to this shortcoming would be to simulate all actions for every rotation. Due to resource constraints this is not possible. However executing the simulations for a small number of rotations is possible. To iteratively compute the best direction a particle filter as described in Section 2.3 is used. Each kick action is evaluated for n robot rotations. These rotations are the particles for the particle filter. The evaluation of the kicks for each particle is done as described in Section 3.2. The particles' likelihood is determined by the positions of the individual samples of a kick simulation, e.g. $INFIELD$ and $OPPGOAL$. The resampling is done as shown in algorithm 1. The process is repeated 10 times. The mean of the samples after 10 resamplings, here meaning robot rotations, is then used as the value the robot needs to turn in order to achieve the best result. The action that needs the least rotation is executed after the appropriate rotation. This approach is in a sense the opposite to the previous one. Here the robot always turns the best way it can and then shoots. In order to improve on the time the robot spends on turning around the ball a lower threshold for the turn angle can be implemented. Also it might make sense to set a maximum time allowed to turn around the ball. A comparison between the original strategy and the *turn strategy* is presented in Section 4.2.

Algorithmus 2 : Decision

Input : action set
Data : $\mathcal{A}_{acc} \leftarrow \{\}$, $\mathcal{A}_{goal} \leftarrow \{\}$
Output : best action

foreach $a \in$ *action set* **do**
    if $p(GOALOWN|a) \leq T_1$ **then**
        $\mathcal{A}_{acc} \leftarrow \mathcal{A}_{acc} \cup a$
    end
    if $p(INFIELD \cup GOALOPP|a) \geq T_0$ **then**
        $\mathcal{A}_{acc} \leftarrow \mathcal{A}_{acc} \cup a$
        if $p(GOALOPP|a) \geq 1$ **then**
            $\mathcal{A}_{goal} \leftarrow \mathcal{A}_{goal} \cup a$
        end
    end
end
if $|\mathcal{A}_{acc}| = 0$ **then**
    return turn action
end
if $|\mathcal{A}_{acc}| = 1$ **then**
    return $\mathcal{A}_{acc}\{0\}$
end
if $|\mathcal{A}_{goal}| = 0$ **then**
    return $argmax(value(a)|a \in \mathcal{A}_{acc})$
end
if $|\mathcal{A}_{goal}| = 1$ **then**
    return $\mathcal{A}_{goal}\{0\}$
end
return $argmax(value(a)|a \in \mathcal{A}_{goal})$

Chapter 4

Experimental Evaluation

For this thesis a simple 2D Simulator was developed for evaluating how the action selection algorithm performs with different initial conditions. For experiments that require the simulation of consecutive actions the mean of the samples is assumed as the future robot position. The process is shown in Figure 4.1.

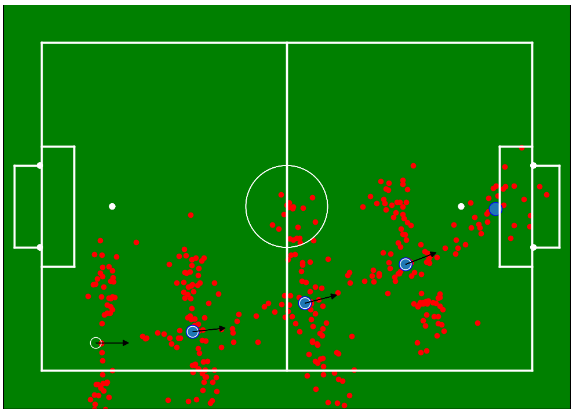


Figure 4.1: *Robot is represented as white circle, the arrow indicates the robots rotation on the field. The red circles represent particles and the blue circle is the mean of the particles of the best action*

4.1 Estimating the Number of Simulations

A central aspect of the simulation based decision approach is the evaluation of samples and not the full underlying probability density functions. To make this algorithm usable on the NAO robot platform only a small number of samples can be used. To test if using more samples results in a significant qualitative improvement, multiple experiments were conducted using an abstract 2D simulator. For every position (x, y, rotation) on the field the decision was calculated 100 times. From the resulting histogram for every position the highest column represents the most likely decision. The number of positions where the highest column has less than 50 % of the 100 decisions is shown in Figure 4.2. The field resolution in x and y was 200mm and a rotation step of 20° which resulted in 22032 evaluated states. Using 100 or more samples to estimate a kick would result in a significant better result, which has less uncertainty. Unfortunately this is not feasible on the robot platform. In

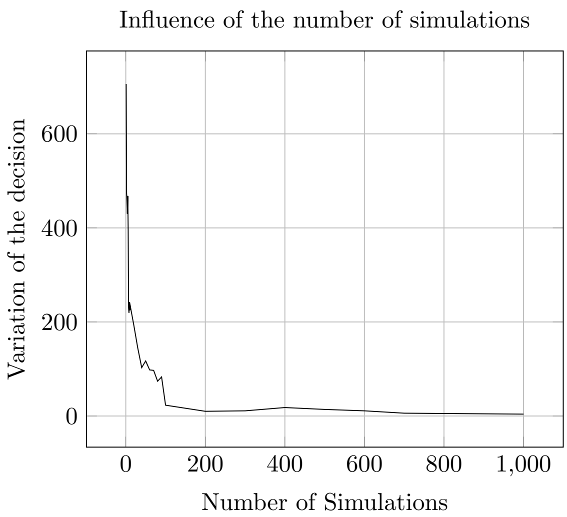


Figure 4.2: *Plot of the variation in decision depending on the number of simulations used to make a decision.*

[17] 30 samples were used. This analysis shows that using 30 samples to estimate the result of a kick is appropriate as increasing the number of samples has only a small effect on the uncertainty. Even by using more samples it can still happen that at a particular position the decision changes from one simulation to the next. It is just less likely to happen. To improve the quality of decision making a histogram of decisions like the one used in this analysis could be introduced. This would prevent executing spurious decisions. However the histogram must be recalculated as soon as the robot moves or when opponent players are part of the simulation whenever they move as well.

4.2 Strategy Comparison

In [17] the kick selection would always prefer to execute a kick and only turn around the ball when no acceptable kick could be performed. This strategy is shown in algorithm 2. In Section 3.3.1 another strategy, which prefers to turn first, is introduced. In this section a comparison between the two approaches are presented. To estimate how well a strategy works, I estimated how fast the strategy can score a goal on an empty field. This experiment was conducted in an abstract 2D Simulator which is described in Section 3.2.2. Figure 4.3 shows how the time that the robot needs from

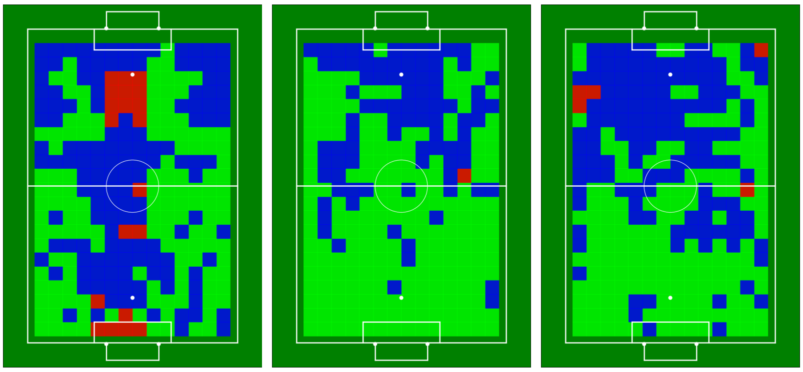


Figure 4.3: *The start positions are indicated by the colored patches. Each image corresponds to a specific rotation at the start position. Left: 0°, Middle: r-90°, Right: 180°. Green Patches indicate that the particle filter approach was at least 5 seconds faster, red patches indicate that the simulation based algorithm was 5 seconds faster or more. Blue patches indicate that the time difference was less then 5 seconds.*

a certain position to score a goal compares between the strategies. Positions marked green represent positions from which the turn strategy was significantly faster (10 seconds or more). For blue positions there are no significant differences. At positions marked red the normal simulation based algorithm as described in Chapter 3 was faster. Each simulation was repeated 100 times. In Figure 4.4 the disadvantage of the turn strategy is visible. In this strategy the robot will turn a little even if the current rotation is near perfect. Since the rotation speed of the robot is comparable slow, this influences the total time in a significant way. The turn strategy approach is seems better when the robot is not already turned towards the goal. The same analysis is done for the number of kicks until a goal was scored. Overall the turn strategy seems to be an improvement.

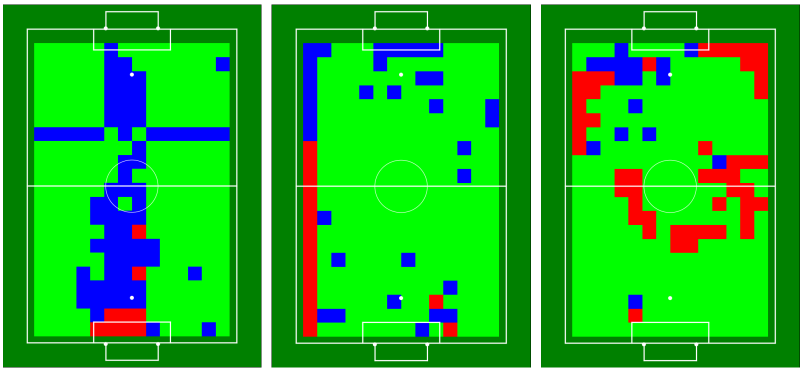


Figure 4.4: *The start positions are indicated by the colored patches. Each image corresponds to a specific rotation at the start position. Left: 0°, Middle: r-90°, Right: 180°. Green Patches indicate that the particle filter approach took less kicks motions, red patches indicate that the simulation based algorithm took less kick motions. Blue patches both approaches took the same amount of kicks to score a goal.*

4.3 Quantitative Analysis in Real Game Situations

As mentioned in Chapter 3 the original motivation behind the simulation based decision approach was to decrease the times the robot shoots outside the field. In [17] games with the old approach were compared against games with the simulation bases method. For this purpose videos overlooking the whole field of the games were recorded during RoboCup competitions in 2015 alongside with log files recorded by each of the robots. Video recordings provide a ground truth of the situation while log data recorded by the robots provides the corresponding internal state. The log files contain perceptions and the behavior decision tree for every cognition cycle (33 ms). This allows the extraction of situations in which the robot took the decision to kick. The logs have been synchronized with the corresponding video files and the extracted kick actions were labeled manually. The labeling procedure has been performed with the help of the interface, which had been designed specifically for this purpose. Figure 4.5 illustrates an example of a labeling session for the first half of the game with the team *NaoDevils* at the RoboCup 2015. The labeling criteria consist of 15 distinct boolean labels in three categories: technical execution of the kick, e.g., robot did miss the ball; situation model (was the estimation of robots position on the field and the ball correct?); result of the action and strategic improvement of the situation (ball left the field, was moved closer to the opponent goal etc.).

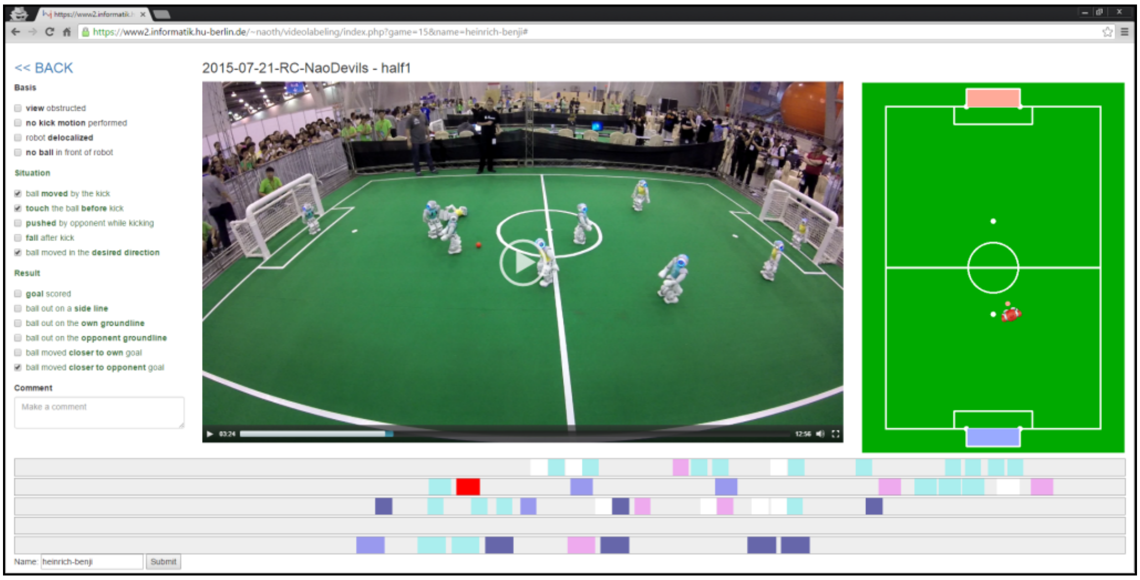


Figure 4.5: *Illustration of the labeling interface used to collect data regarding the quality of the kicks. At the bottom are time lines for each of the robots. Different actions are represented by buttons on the time line with different colors. On the right the robots estimated state is visualized, i.e., estimation of its position, ball model and obstacles. On the left are three categories of labels capturing the quality of the action.*

4.3.1 Data Set

For the analysis games that the Nao Team Humboldt has played in two different competitions in 2015 – the *German Open 2015* (GO15) and *RoboCup 2015* (RC15) were considered. In both competitions the NaoTH robots performed well and reached the third place at the German Open and quarter finals at the RoboCup. At the GO15 the previous solution mentioned in Chapter 3 were used while at the RC15 the presented simulation based approach had been employed. From GO15 a total of five game halves have been analyzed with: *ZKnipsers* (two halves, preliminaries); *HULKS* (first half, preliminaries); and *Nao Devils* (two halves, game for the 3rd place). And from RC15 three complete games were analyzed with: *RoboCanes* (two halves, preliminaries); *Nao Devils* (two halves, intermediate round); and *HTWK* (two halves, quarter finals). The selection of the analyzed games depended largely on the availability of the videos and log data. The names *ZKnipsers*, *HULKS*, *Nao Devils*, *RoboCanes* and *HTWK* refer to SPL teams.

Algorithm	New		Old	
Total number of kicks	163		196	
Robot was localized	150	(92.02 %)	165	(84.18 %)
Successful execution	93	(57.06 %)	153	(78.06 %)
Failed execution	70		43	
Failed: opponent interference	33	(47.14 %)	14	(32.56 %)
Failed: technical failure	37	(52.86 %)	29	(67.44 %)
Successful execution + Localized	86	(52.76 %)	131	(66.84 %)
+1	67	(77.91 %)	88	(67.18 %)
0	15	(17.44 %)	39	(29.77 %)
-1	4	(4.65 %)	4	(3.05 %)
Out at opponent goal line	1	(1.16 %)	8	(6.11 %)

Table 4.1: *Analysis results of video material. The new algorithm shows a higher rate of strategic improvements (+1) and a lower rate of mediocre kicks (0). It is also about 5 times less likely to kick out at the opponent field line.*

4.3.2 Results

To single out the effect of the kick selection only kicks in which the robot was well *localized* (so it knew what it was doing) and the kicks that were executed successfully, i.e., the ball went in the intended direction and did not collide with opponent, are considered. In short: *successful* kicks are the ones which comply with our action model as described in section 3.1.2. The top part of the table 4.1 illustrates the numbers of the successful and failed kicks. Our analysis has also revealed that a high percentage of the actions fail due to various reasons. The main reasons appear to be failure in the technical execution, e.g., the robot trips and doesn't kick the ball properly and interference by opponent players. Both aspects are not part of the simulation and require further investigation. The table 4.1 (**Failed execution**) summarizes the rates of the failed kicks split in these two cases. The higher opponent interference in the case of the new approach can be explained by the more challenging opponent teams at the RoboCup 2015. In the lower part of the table 4.1 the evaluation of the kick results according to the strategic improvement of the ball position as described above are summarized. The separation used here is very rough: +1 corresponds to the cases in which the strategic position of the ball was clearly improved by the action, e.g., it was moved closer towards the opponent goal; -1 was given when the ball moved towards own goal or away from the opponent goal; and 0 when no improvement was visible, e.g., ball moved along the middle line. The results show that the new approach results in a higher rate of improvements

(+1) and a lower rate of mediocre kicks (0), while the rate of cases in which the position of the ball worsened (-1) remained at a comparable level. Another important factor is the *number of times the ball leaves the field*, because it results in a tactical disadvantage as the ball position is reset to the disadvantage of the team which shoot the ball out. The penalty is especially severe when the ball leaves on the opponent goal line, since the ball is then reset to the middle line. In this case a significant improvement with the new approach can be seen, as only one kick (1.16 %) left the field at the opponent goal line in contrast to more than 6 % (8 kicks) with the old solution. In summary, the data shows that the new approach performs more robustly than our previous solution. The new algorithm is about 5 times less likely to kick out at the opponent field line (decrease by 81 %) and 16 % more likely to kick towards the opponent goal.

Chapter 5

Discussion and Future Work

A method for fast decision making was presented. While the simulation based idea was introduced and evaluated in the specific domain of the kick selection problem in *RoboCup*, I believe this approach can be used in other domains as well. The core idea of the simulation based decision making method is to approximate a complex state with a number of simple deterministic simulations. The models used for those simulations depend on the task. Here I argue that simple models are sufficient enough to compare outcomes of possible ball kicks. It is important to note that estimating the exact future ball position is not needed in order to compare the kicks. Experimental data collected in real RoboCup games has shown that the algorithm performs very well and is an improvement over the algorithm used by the *Nao Team Humboldt* before.

A significant advantage of the simulation based decision making approach is the modularity. Every part can be improved and changed without breaking the overall behavior. The simulation can be easily changed to incorporate models for team members and opponents. Those models might be complex but their integration in the proposed decision making process is not since only the influence on one sample needs to be explicitly modeled. Following I want to highlight some possible future improvements.

5.1 Positioning Behavior

So far the presented approach only captures decisions that the robot next to the ball needs to take. This leaves the question of how to deal with situation in which the robot is not in possession of the ball. Supporting robots can simulate the action of the robot that is currently closest to the ball (striker) by executing the decision making algorithm as the striker would do. That means using the position of the striker instead of their own. This gives the supporting players the information where the ball will be and when. The passive player can then go to this place in anticipation of a pass. Note, that the striker does not need to know that it is about to pass the ball to another player. The situation is more difficult, when the opponent player has the ball. One can't simulate their kicks since the behavior is unknown. But assumptions can be made about the opponents decision making. The easiest way is to assume that they behave exactly as our robots do, but play towards our goal, maximizing the potential value instead of minimizing it. In that case, given the opponents positions and the ball position is known, the defending robot can calculate the future ball position and its probability distribution. Usually, the behavior of the opponent differs a lot from our robots' behavior, so a model of the opponent kick behavior has to be learned. It might be possible to learn this from videos of previous games since overall team behavior tends to change slowly from year to year. Kick events from opponents can be labeled similar to what is explained in Section 4.3. In [14] an approach for learning a model of a human soccer teams is presented.

5.2 Simulating Foot Steps

Currently the parameters for a kick are experimentally determined. A kick is a special foot trajectory. That means that changes effecting the trajectory (stabilization, walking engine) also effect the kick. Those changes are not part of the simulation process. It is possible to simulate whether a step, special or not, effects the ball by simulating the contact point between ball and foot and then use a simple physics simulation again to calculate the resulting endpoints of the ball trajectory.

Appendix A

Parameter Estimation

A.1 Estimation of Kick Action Parameters

The four kick parameters, initial velocity of the ball, kick direction and their corresponding standard deviations were estimated experimentally. To estimate the rolling resistance coefficient for a particular surface and ball, multiple experiments with an inclined plane were performed. The ball started to roll down from different heights and the distance the ball rolled on the surface was measured. From the height and length of the plane the velocity of the ball at the time when it hits the ground υ_p can be estimated with:

$$\upsilon_p = \frac{g \cdot h}{l} \cdot t_p \tag{A.1}$$

where t_p is the time the ball needs to roll down the plane. We then measured the distance in multiple experiments. By transposing the rolling distance formula, the rolling resistance coefficient can be calculated.

$$c_R = \frac{1}{2} \cdot \frac{\upsilon_p^2}{g \cdot d} \tag{A.2}$$

where υ_p is the starting velocity, g the gravitational constant, and d the total distance the ball traveled on the surface. The mean of the calculated coefficients is used as the rolling resistance coefficient for the next calculations. To calculate the initial velocity of a kick, the distance the ball rolled after a particular kick was measured in an experiment. By using the stopping distance formula, the initial velocity of one kick can be calculated by

$$\upsilon_0 = \sqrt{d \cdot 2c_R \cdot g} \tag{A.3}$$

where υ is the initial velocity of the ball. c_R the rolling resistance coefficient and g the gravitational constant. The mean of υ_0 of multiple repetitions approximates the initial velocity of this action. The standard deviation of the repetitions is used as the standard deviation of the velocity of the kick.

Appendix B

RoboCup Initiative

B.1 RoboCup

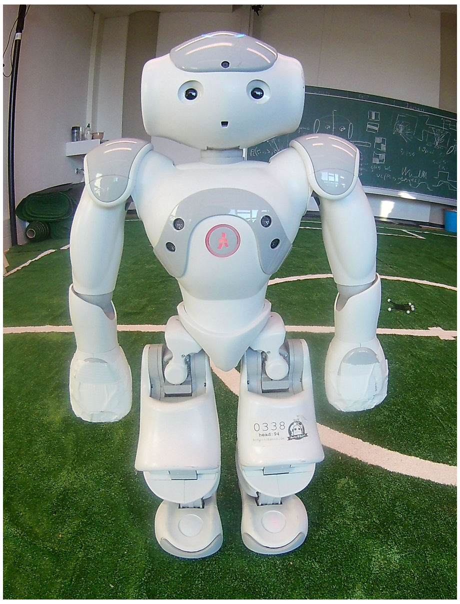
Since 1997 the RoboCup World championship is held yearly with the goal foster the development of a robot soccer team which can beat the FIFA World Champion by 2050. Since then different leagues have been established with different goals like RoboCup Rescue, @Home and Junior. In RoboCup Soccer there are different leagues, each with it's own unique research focus. There are the Humanoid, Small Size, Middle Size, 2D, 3D Simulation and the Standard Platform League. The SPL stands out as the only league where all robots are the same model. So the robots differ only by the programmed behavior. Since 2008 the NAO robots by Aldebaran are used as the Standard Platform. In the humanoid leagues the robots hardware is developed by individual teams themselves. They are divided in three sub leagues corresponding to the height of the robots. The Middle size and small size league are non humanoid.

B.2 NAO Robot

The NAO robot is produced by the french company Aldebaran. They produced 5 different versions so far. In the following table an excerpt of the specifications for the v5 model is listed. The specifications are taken from the official documentation provided by Aldebaran. Older models are allowed to be used in the SPL.

Table B.1: *Overview of Nao v5*

Operating System	
Name	NAOqi (Embeded Linux)
Version	2.1.4.13
CPU	
Model	Intel Atom Z530
Architecture	Silverthorne, x86
Clock Speed	1.6 GHz, 533 MHz FSB
Cores	1
Cache	512 kB
Memory	
RAM	1 GB
External	2 GB (Flash), 8 GB (SDHC)
Measurements	
Dimensions	573mm $\times$ 311mm $\times$ 275mm (Hight $\times$ Depth $\times$ Width)
Weight	5.4 kg
Degrees of Freedom	25
Sensors	
Cameras	2 á 1280 Pixel $\times$ 960 Pixel, 30 FPS 60,9° horizontal opening angle 47,6° vertical opening angle
Microphones	4 a 20 mV/Pa $\pm$ 3dB sensitivity Frequency range: 150Hz bis 12kHz
Ultrasound	2 $\times$ 2 (1cm – 4cm resolution, frequency: 40kHz, 60° opening angle)
Inertia Sensors	3-axis Gyro and 3-axis acceleration sensor
Proprioceptive sensors	perception of joint positions with Hall sensors (ca. 0,1° resolution)



Bibliography

[1] Publications related to the robocup soccer simulation league. `http://www.cs.utexas.edu/~pstone/tmp/sim-league-research.pdf`. Accessed: 2017-08-24.

[2] Mazda Ahmadi and Peter Stone. *RoboCup 2007: Robot Soccer World Cup XI*, chapter Instance-Based Action Models for Fast Action Planning, pages 1–16. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[3] Samuel Barrett, Katie Genter, Todd Hester, Michael Quinlan, and Peter Stone. Controlled kicking under uncertainty. In *The Fifth Workshop on Humanoid Soccer Robots at Humanoids 2010*, Nashville, TN, 2010.

[4] A. Bordallo, F. Previtali, N. Nardelli, and S. Ramamoorthy. Counterfactual reasoning about intent for interactive navigation in dynamic environments. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 2943–2950, Sept 2015.

[5] Hans-Dieter Burkhard, Markus Hannebauer, and Jan Wendler. *AT Humboldt — Development, practice and theory*, pages 357–372. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[6] R. Dodds, P.A. Vallejos, and J. Ruiz-del Solar. Probabilistic kick selection in robot soccer. In *Robotics Symposium, 2006. LARS '06. IEEE 3rd Latin American*, pages 137–140, Oct 2006.

[7] Clemens Eppner, Roberto Martín-Martín, and Oliver Brock. Physics-based selection of actions that maximize motion for interactive perception. In *RSS workshop: Revisiting Contact - Turning a problem into a solution*, 2017.

[8] Pablo Guerrero, Javier Ruiz-Del-Solar, and Gonzalo Díaz. Robocup 2007: Robot soccer world cup xi. chapter Probabilistic Decision Making in Robot Soccer, pages 29–40. Springer-Verlag, Berlin, Heidelberg, 2008.

[9] T. Hellström. Robot navigation with potential fields. Uminf-11.18, Department of Computing Science, Umeå University, 2011.

[10] Germund Hesslow. Conscious thought as simulation of behaviour and perception. *Trends in Cognitive Sciences*, 6(6):242 – 247, 2002.

[11] O Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. Rob. Res.*, 5(1):90–98, April 1986.

[12] Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, pages 1398–1404 vol.2, Apr 1991.

[13] Lars Kunze and Michael Beetz. Envisioning the qualitative effects of robot manipulation actions using simulation-based projections. *Artificial Intelligence*, pages –, 2015.

[14] Hoang M Le, Peter Carr, Yisong Yue, and Patrick Lucey. Data-driven ghosting using deep imitation learning. In *MIT Sloan Sports Analytics Conference 2017*, 2017.

[15] David L. Leottau, Javier Ruiz-Del-Solar, Patrick Macalpine, and Peter Stone. A study of layered learning strategies applied to individual behaviors in robot soccer. In *RoboCup 2015: Robot World Cup XIX on RoboCup 2015: Robot World Cup XIX - Volume 9513*, pages 290–302, New York, NY, USA, 2015. Springer-Verlag New York, Inc.

[16] M. Macktoobian and S. A. A. Moosavian. Time-variant artificial potential fields: A new power-saving strategy for navigation of autonomous mobile robots. In *2013 First RSI/ISM International Conference on Robotics and Mechatronics (ICRoM)*, pages 121–127, Feb 2013.

[17] Heinrich Mellmann, Benjamin Schlotter, and Christian Blum. *Simulation Based Selection of Actions for a Humanoid Soccer-Robot*, pages 193–205. Springer International Publishing, Cham, 2017.

[18] Heinrich Mellmann, Benjamin Schlotter, Steffen Kaden, Philipp Strobel, Thomas Krause, Claas-Norman Ritter, Tobias Hübner, and Schahin Tofangchi. Berlin united - nao team humboldt team report 2016. techreport, Adaptive Systeme, Institut für Informatik, Humboldt-Universität zu Berlin, Unter den Linden 6, 10099 Berlin, Germany, 2016.

[19] M. Moradi, M. A. Ardestani, and M. Moradi. Learning decision making for soccer robots: A crowdsourcing-based approach. In *2016 Artificial Intelligence and Robotics (IRANOPEN)*, pages 25–29, April 2016.

[20] R. Osorio-Comparán, I. López-Juárez, A. Reyes-Acosta, M. Pena-Cabrera, M. Bustamante, and G. Lefranc. Mobile robot navigation using potential fields

and lma. In *2016 IEEE International Conference on Automatica (ICA-ACCA)*, pages 1–7, Oct 2016.

[21] Guido Schillaci, Verena V. Hafner, and Bruno Lara. Exploration behaviors, body representations, and simulation processes for the development of cognition in artificial agents. *Frontiers in Robotics and AI*, 3:39, 2016.

[22] Shingo Shimoda, Yoji Kuroda, and Karl Iagnemma. High-speed navigation of unmanned ground vehicles on uneven terrain using potential fields. *Robotica*, 25(4):409–424, July 2007.

[23] Sebastian Thrun. Particle filters in robotics. In *in Proceedings of the 17th Annual Conference on Uncertainty in AI (UAI*, 2002.

[24] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[25] T. Weber, S. Racanière, D. P. Reichert, L. Buesing, A. Guez, D. Jimenez Rezende, A. Puigdomènech Badia, O. Vinyals, N. Heess, Y. Li, R. Pascanu, P. Battaglia, D. Silver, and D. Wierstra. Imagination-Augmented Agents for Deep Reinforcement Learning. *ArXiv e-prints*, July 2017.

[26] T. Weerakoon, K. Ishii, and A. A. F. Nassiraei. Dead-lock free mobile robot navigation using modified artificial potential field. In *2014 Joint 7th International Conference on Soft Computing and Intelligent Systems (SCIS) and 15th International Symposium on Advanced Intelligent Systems (ISIS)*, pages 259–264, Dec 2014.

[27] Yuan Xu. *From simulation to reality*. PhD thesis, Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät II, 2014.

Selbständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe. Weiterhin erkläre ich, eine Bachelorarbeit in diesem Studiengebiet erstmalig einzureichen.

Berlin, den 1st December 2017 ....

Statement of authorship

I declare that I completed this thesis on my own and that information which has been directly or indirectly taken from other sources has been noted as such. Neither this nor a similar work has been presented to an examination committee.

Berlin, 1st December 2017 ....