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

Design and Empirical Analysis of Motion and Behavior Primitives for Humanoid Soccer Robots

Bachelorarbeit

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

eingereicht von:	Johan Hartung
geboren am:	08.06.2001
geboren in:	Haan
Gutachter/innen:	Prof. Dr. Verena Hafner Prof. Dr. Thomas Kosch

eingereicht am: ………… verteidigt am: …………

Abstract. Kicking is one of the most central aspects in humanoid robotic soccer. In this thesis, we will study robotic kicking as motion and behavior primitive. We start by discussing the kick behavior primitive with reference to the Berlin United kicking process. To improve the kick distance and prepare for hot-swappable kick actions, we evaluate the different motion variables of the kick motion primitive and their influence on the resulting kick. With the experiment-driven optimizations and introduced techniques, we are able to increase the mean absolute kicking distance, by 37.98 %, compared to the previous kick configuration. Furthermore, we explore possibilities to enhance the robot's predictive action model by evaluating the robot's capability to verify it. Also, we propose procedures to improve its predictive value and introduce a time-based and a probabilistic approach to decide, when to kick the ball.

Contents

1 Introduction 1
1.1 Contributions . . . 2
1.2 Related Work . . . 3

2 Kick Behavior Primitive 5
2.1 Overview . . . 5
  2.1.1 Architectural Foundation . . . 6
  2.1.2 Path Request . . . 6
  2.1.3 Step Buffer . . . 6
  2.1.4 Kick Types . . . 7
2.2 Four Phases of a Kick . . . 7
  2.2.1 Alignment Phase . . . 8
  2.2.2 Configuration Phase . . . 9
  2.2.3 Execution Phase . . . 10
  2.2.4 Ballistic Phase . . . 10

3 Parameterless Kicking 11
3.1 Dynamic ZMP-Offset . . . 11
3.2 Methodology . . . 12
  3.2.1 Experiment Setup . . . 13
  3.2.2 Terminology . . . 14
  3.2.3 Evaluation of the Results . . . 14
  3.2.4 Limitations and Scientific Value . . . 15
3.3 Velocity . . . 15
3.4 Kick Height . . . 16
3.5 Kick Length and Adaptivity . . . 17
3.6 Kick Target Selection . . . 18
3.7 Ball Distance . . . 22
3.8 Kick Actions . . . 24
3.9 Summary . . . 24

4 Target Prediction and Simulation Correction 27
4.1 Integration of Kicking in Decision Making . . . 27
4.2 Autonomous Simulation Adjustment . . . 29
  4.2.1 Online Kick Distance Measurement . . . 29
  4.2.2 Reliability of the Online Measurement . . . 31
4.3 Refined Action Prediction . . . 32
4.4 Prediction Based Kick Decision . . . 33
  4.4.1 Kick Prediction . . . 33
  4.4.2 Evaluation . . . 35
4.5 Time-Based Kick Decision . . . 37
  4.5.1 Motivation . . . 37

4.5.2 Functionality 38
4.5.3 Implementation 40
4.5.4 Evaluation 40
4.6 Kick Action Extension 41

5 Discussion and Outlook **43**
5.1 Future Work 44

Appendix **v**

Acronyms

k-NN	k-nearest neighbor
APM	action prediction model
KTO	Kick Target Offset
MAE	mean absolute error
MLP	multilayer perceptron
PDC	Progressive Deviation Control
SPL	Standard Platform League
ZMP	Zero Moment Point

1 Introduction

In humanoid robotic soccer, kicking is one of the most crucial aspects. It is also a very complex task, as it is strongly dependent on nearly all major abilities of the robot. A successful kick requires cognitive abilities, such as a robust and accurate localization of both the ball relative to the robot and its own position in the field, a good path planning or a fast and certain decision making, as well as motoric abilities like reliable stabilization and precise movement control. Without being able to localize the ball, the robot will not be able position itself and kick it properly. Without being able to localize itself on the field, the robot will not be able to intentionally kick the ball in a desired direction. A good path planning is needed to bring the robot into an adequate position; stabilization and movement control are vital for the kick motion to function. However, these are just the enablers, the foundation on which the kicking behavior primitive is built. The kick motion primitive involves many motion variables, each of which can greatly affect the overall outcome of the kick. Oftentimes, the kick motion primitive or single kick motion variables are studied as isolated events, without considering the other factors. In this work we will study robotic kicking as motion primitive, i.e., the kick motion, as well as behavior primitive, i.e., the full kicking process, including the approach, to gain a more wholistic impression, of what makes a good kick. Furthermore, we will explore possibilities to improve the robot's decision making.

To prepare for the following work, we will first assess kick behavior primitives (Section 2), with reference to the Berlin United kicking process. The way kicking is currently implemented in the Berlin United codebase, it is highly reliant many, manually set, interdependent parameters. This drastically limits the adaptivity and makes the inclusion of new kick motions or the adjustments to new environments a complex and time consuming task. To combat this, in *Parameterless Kicking* (Section 3), we aim to leave behind the rigid structure of hardcoded parameters in our current kicking process, to allow for more dynamic and adaptive kicking. We will investigate the effects of different motion variables, such as velocity or kick height on the kick's performance, to gain a deeper understanding of their influence. We will then use this knowledge to form a `KickAction`, containing all relevant information, specific to each kick behavior primitive. With this, we work towards a low-threshold way to include new kick actions, that allow for a quick and easy changeability in runtime, while integrating in the existing best action selection. In the second main part, *Target Prediction and Simulation Correction* (Section 4), we will analyze whether the robot's own ball model can be used to verify and adjust the action prediction model and explore how this model can be improved in accuracy and features. To decide on a good moment to execute a kick, we will propose both a time-based and a probabilistic approach.

Figure 1: The Berlin United robots during a soccer match at the *Robotic Hamburg Open Workshop (RoHOW)* 2023. Photo taken by Murat Kirtay.

1.1 Contributions

This is an overview of our contributions, featured in this work.

- Discussion of kick behavior primitives, with reference to the Berlin United kicking process.
- Introduction of a model, consisting of four consecutive phases to describe the kick behavior primitive. Each phases decreases the robot's influence on the resulting kick.
- Evaluation of the robot's stability for different kick durations.
- Design and implementation of a function that dynamically assigns each kick a suitable ZMP kick offset, to assure the robot's stability.
- Installation of a top down camera as an optical measuring system, to evaluate the robot's kicking performance.
- Development of a program that captures images, automatically annotates the ball and computes its position on the soccer field.
- Empirical evaluation of different motion variables and their impact on the kick.
- Improvement of the robot's kicking performance through experiment-driven optimizations and introduced techniques such as an offset to the kick target.
- Discussion of how kicking is integrated in the decision making process, at the example of the Berlin United action selection.

- Assessment of the noise in the speed perception of the robot's ball model.
- Implementation of the ability, for the robot to recognize the end of a kick.
- Evaluation of the robot's ball model in terms of accuracy in distance measurement.
- Exploration of possible ways to improve the predictive action model.
- Design and development of a model to predict the kick' outcome based on the robot's current position, to probabilistically decide when to execute a kick.
- Comparison of different supervised learning approaches, to make such predictions.
- Design and implementation of a novel, time-based kick moment control, by progressively increasing the allowed deviation from the robot's target position.
- Identification and selection of important parameters, to be included in a `KickAction` module.

1.2 Related Work

When thinking about motion in humanoid robots, a common approach is to examine human movement [11, 22, 26]. This is mostly done by using motion capture data, which can be processed and adapted to the robot's kinematics. In the context of humanoid robot soccer, this approach comes with strong advantages, such as high dynamics and natural movement, but also with limitations, as the development of new motions is dependent on capturing new data. Due to the differences in kinematics and dynamics between humans and humanoid robots, human motion data cannot be directly applied to humanoid robots [26]. Hence, this approach benefits robots with very human-like kinematics and physiology, such as the PresToe robot [2], used by Marew et al. in their work [11]. The NAO robot, which is used in this thesis, has a very different kinematic structure compared to humans, or more human-like humanoid robots. And while human motion imitation on the NAO has been done and is therefore technically feasible, the extent of motion that was achieved in previous works is quite rudimentary [8] and not comparable to the complex and dynamic motions needed for kicking a ball in a soccer game.

With a few exceptions, such as Müller et al. using hand-crafted piecewise Bézier curves [4, 20], in the field of competitive robot soccer, there are two main approaches, for generating the kick motions: predefined keyframes or spline based trajectories [23,24]. Keyframe based motion, i.e., defining fixed sets of joint angles in time and interpolating between them, is a simple and straightforward approach. Since all relevant motion points are predefined, it is easier to implement and understand, but also excludes any dynamic reaction and adaptation to the environmental conditions and developments, by design [24]. Therefore, the online generation of dynamic kick and step trajectories has been a popular research topic in humanoid robotics for many years. The concept of spline based trajectories in humanoid robotics first emerged in the early 2000s.

In 2004, Zhaoqin Peng et al. [27] presented a method for planning steps by using splines to generate trajectories online, based on parameters such as step length and hip motion. In the RoboCup Standard Platform League (SPL), spline based, online trajectory generation for kicking, was first introduced in 2014, by Wenk and Röfer [24], to overcome the keyframe reliant approach they have originally used within their team. In their work they describe how they automatically generate trajectories based on the ball position, kick velocity, and kick direction [4]. Böckmann and Laue compared the approaches of Müller et al. [20] and Wenk et al., criticizing the complex and time consuming task of handcrafting Bézier curves and the missing possibility to influence the fully automatically generated trajectories by Wenk et al. and therefore presented a way to both easily handcraft and automatically crate kick motions, as a middle ground.

In their bachelor thesis [23], Wege assessed the correlation between a kick's duration and the resulting distance traveled by the ball, finding that the duration can be used to control the kicked distance. In their *Requirements for a Dynamic Kick*, they describe the capability of robots passing the ball to each other, as vital for a soccer game to be successful. Ergo, the robots need to be able to scale how far they are kicking the ball. However, the usability for their approach to control the covered distance with a precision, that enables passing to other robots during a soccer game, is still limited. In their experimental evaluation, they have to account for a relatively high empirical standard deviation for kick distances with a mean distance, higher than 100 cm.

There have been frequent endeavors to let robots learn behavior primitives, such as kicking [9,10]. However, this is usually done in simulated environments and transferring these approaches to real robots is rarely successful. Mostly, in these simulations, factors, such as the uncertainty in perception or the varying friction and unevenness of undergrounds, is not accurately represented, which widens the reality gap [15]. Even though, there have been successful attempts in learning motion and behavior primitives on real robots [6], this was so far only possible in very controlled settings, which limits the representative value for more dynamic scenarios like robot soccer.

In 2010, Mellmann and Xu explored adaptive, static kick motion primitives, based on visual feedback [18, 25]. They achieved real time adaptation of the robots foot to a moving ball, creating a kick motion, that was actively used in robot soccer games from 2009 to 2013 [15]. Their experiments were conducted using an orange ball the robot was able to accurately see. The RoboCup SPL changed to a more realistic soccer ball in 2016, making such adaptive real time motions much more difficult. As mentioned, they focused on static kicking, i.e., kicking out of the stand. In contrast, we will examine kicks executed out of the walk.

Exploring the kinematics of soccer robots, Je Youn Choi et al. found that the foot being parallel to the ground when colliding with the ball, results in the best impulse, compared to other angles of the robot's ankle [5].

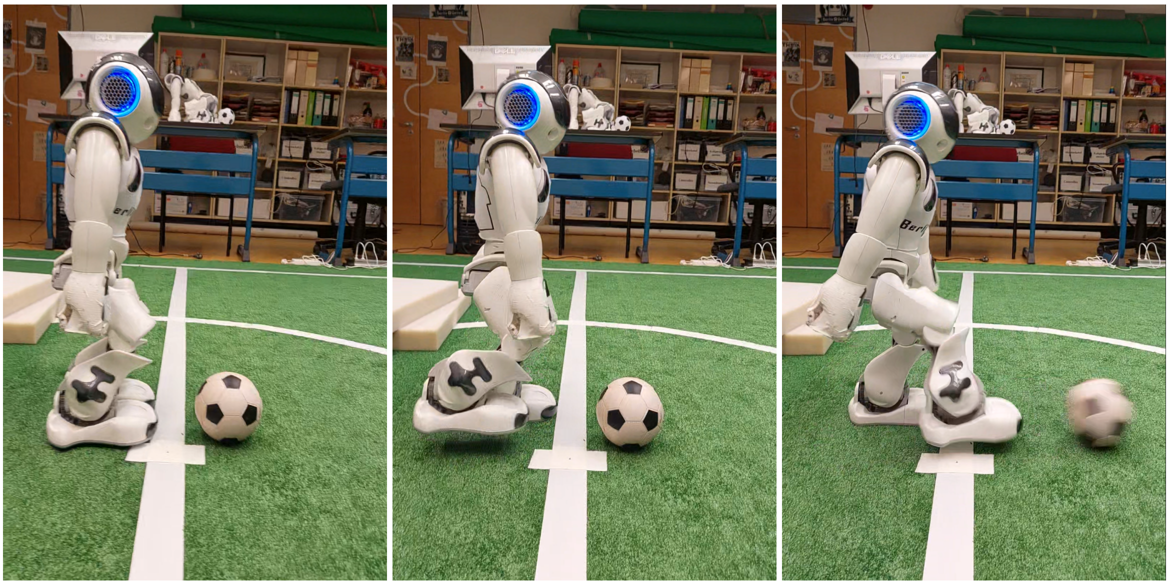


Figure 2: The kick motion primitive in action.

2 Kick Behavior Primitive

In this section, we will assess kick behavior primitives, at the example of the Berlin United kicking process. For this, we will first give an overview about fundamental concepts, to provide the necessary foundation for understanding the following experiments. To describe the kick behavior primitive, we will introduce a model consisting of four consecutive phases, each progressively reducing the robot's influence on the kick.

2.1 Overview

Our robots make use of what is called in-walk kicking. Broken down, this means: Kicks are steps, but different. From a human point of view, this might sound unintuitive, as we might consider steps and kicks as separate actions, but if we look at the very essence, we notice more similarities than differences. Both kicks and steps describe a movement of the robot's foot, which possesses a start and end position, as well as a trajectory to connect them. The main differences are the differently weighted relevances of certain variables that define their respective motion primitive. We can view kicks as a special kind of step. Both consist of the same elements, such as velocity, or how high the foot is lifted. But with kicks, small changes in these parameters can have a big impact on the resulting kick and therefore require finer tuning. The execution of a kick can be seen in Figure 2.

2.1.1 Architectural Foundation

In our team, the codebase is split into two main parts: cognition and motion. The responsibility of each module is easily deductible from their names. Cognition is responsible for the robot's cognitive abilities, such as sensor evaluation and decision making, whereas motion takes in orders from cognition and converts them into real world movements of the robot.

2.1.2 Path Request

Every path related action a robot undertakes, is realized through a path request. Every path request contains a `PathID` (i.e. the behavior primitive), as well as other optional information, relevant for certain path types.

Relevant `PathID`s are:

`PathID::NONE`
Do nothing. If kick is planned, ignore and cancel.

`PathID::AVOID`
Avoid a certain point (e.g., obstacle). The point to avoid is specified in the path request.

`PathID::MOVE_AROUND_BALL`
As name suggests. For this, additional information, such as the direction or the radius for moving around the ball, is specified in the path request.

`PathID::FORWARDKICK`
The kicking behavior primitive, from approaching the ball, to kicking it.

`PathID::SIDESTEP`
Take a step to the side. The direction is specified in the path request.

`PathID::APPROACH_DRIBBLE`
Approach the ball and dribble it to a strategically advantageous position.

The path request starts in the robot's behavior planning. Based on the selected `PathID`, the `PathPlanner` module will call its respective path routine function.

2.1.3 Step Buffer

Just like us humans [13], our robots plan their steps in advance to ensure a smooth and stable walk. For the robots this is done in the step buffer. More precisely, we have two step buffers, one in cognition and one in motion, but we will only focus on the cognition step buffer.

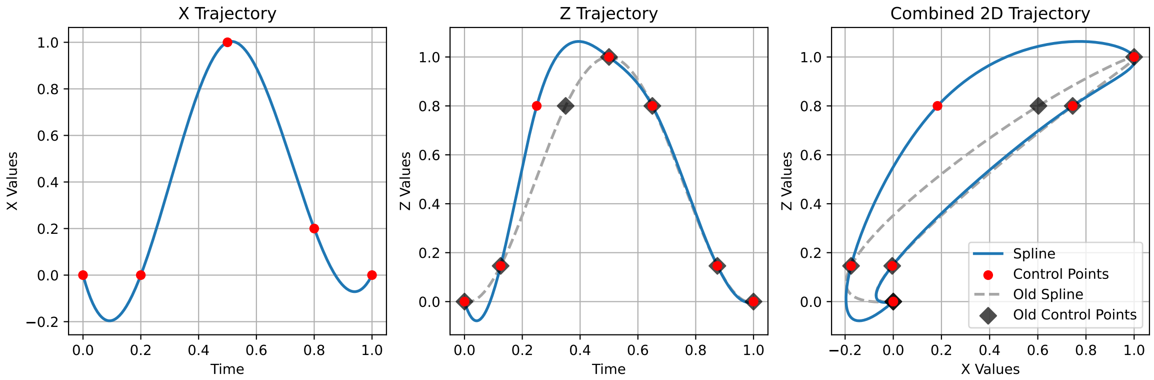


Figure 3: The default forward kick trajectory with references to the deprecated, previously used upwards spline. The left and middle plot show the forward and upwards trajectory of the `KickType` used for the following experiments. The right plot shows the full 2D trajectory. The y-axis motion of the robot's kicking foot is not directly set for this `KickType`, but computed later.

The step buffer is a queue of planned steps. It is used to calculate the robots stability and hip motion in advance. Here, the pre-planned steps are saved as step buffer elements, which contain relevant information about the step or kick. Ideally, we would want it to contain all information that is influential for the kick.

2.1.4 Kick Types

Originally, the spline, used to generate the trajectory, describing the kick's motion primitive, was defined by a set of hard coded points. A change in the trajectory during runtime was therefore not possible. In a study project [7], prior to this thesis, we introduced the `KickType` module, which allows us to define different kick trajectories and switch between them at runtime. Each `KickType` gets initialized with a set of control points and returns splines for up to three dimensions. In said study project, we designed and evaluated a set of new kick types, one of which is now the default forward kick in our RoboCup team and also used in the experiments presented in this thesis. During the RoboCup Eindhoven 2024, the upwards motion trajectory of this kick was further optimized, to achieve a better overall performance (see Figure 3).

2.2 Four Phases of a Kick

The Berlin United kicking process can be logically divided into four consecutive phases, each with shrinking control over the resulting kick.

Alignment Phase
The robot approaches the ball with intent to kick it. Here, the robot decides on

its position, relative to the ball and the foot to kick with. At the start of this phase, the only set decision is the robot's orientation. Apart from this, the robot has full control over the outcome.

Configuration Phase
The robot configures the kick, based on the results of the alignment phase. This includes setting the `KickType` and where to and how fast to kick at.

Execution Phase
Based on the results of the previous two, but especially the configuration phase, the robot executes the kick. This involves computing the trajectory, calculating the kinematic chain and controlling the robot's actuators according to prior configurations.

Ballistic Phase
The kick has been executed and the ball is now in motion. The robot has no further control over the outcome anymore.

2.2.1 Alignment Phase

The alignment phase begins once the robot's decision making calls the `PathPlanner` with a `PathID::FORWARDKICK` path request. Within the `PathPlanner`, there are two relevant functions for executing a kick: `nearApproach_forwardKick()` and `forwardKick()`. The latter is called as soon as the prior returns true, i.e., the robot is in position to kick. This phase describes the lifecycle of `nearApproach_forwardKick()`.

First, we decide on which foot to kick with. This is done by evaluating the ball's position relative to the robot's center. The kicking foot is chosen based on what side the ball is on. For the further approach, we define a target position $\vec{z}$ which is computed from the ball's position $\vec{b}$, a set offset $\vec{o}$ and the ball's radius r, such that:

$$z_1 = b_1 - o_1 - r \text{ and } z_2 = b_2 - o_2 \tag{1}$$

Based on a fixed, rule based decision, every frame we evaluate whether the robot is close enough to the target position $\vec{z}$ to execute the kick. This *close enough* is defined by an allowed deviation $\vec{d}$. The function `nearApproach_forwardKick()` will return `true` and with this initialize the execution of the kick as soon as the foot to kick with is movable and:

$$z_1 < d_1 \text{ and } |z_2| < d_2 \tag{2}$$

Elsewise, the robot will keep approaching the ball. For the further approach, we need to plan the robot's path towards the target position $\vec{z}$. Since $\vec{z}$ is oftentimes not directly reachable with just one step, we need to find an intermediate position $\vec{i}$, such that

$$i_1 = \min(l, z_1 - |z_2|) \text{ and } i_2 = \min(l, |z_2|) \cdot \operatorname{sgn}(z_2), \tag{3}$$

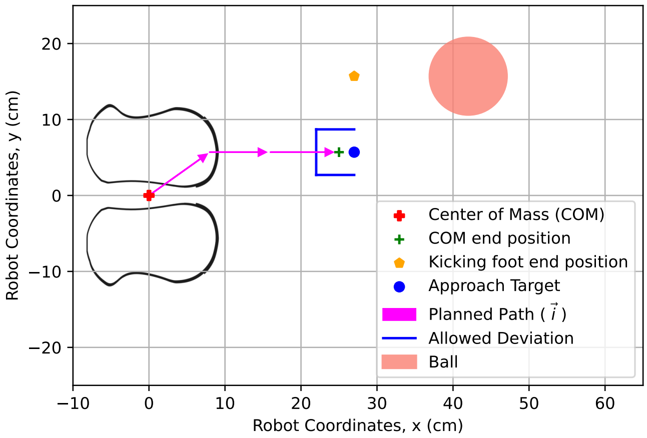


Figure 4: Visualization of `nearApproach_forwardKick()`, showing the path planning for the robot's final steps before kicking the ball. We can see the desired approach target and the actual end position of the robot's center, within the allowed deviation.

with l being a selected step length.

This function gets called every frame, until the robot is close enough to the target position $\vec{z}$ and ready to kick. In this case, the function returns `true` and the robot will proceed to the configuration phase.

2.2.2 Configuration Phase

In the configuration phase, we still have a fair amount of control over the kick, yet we can not longer influence the robot's position relative to the ball. As mentioned before, the successful termination of `nearApproach_forwardKick()` leads to the call of `forwardKick()`. This function describes the configuration phase.

We start this phase by confirming the previously estimated ball position relative to the robot's center and finally deciding on the kicking and supporting foot. To further prepare the kick, we select a kick type [7]. Next, the robot chooses the kick and step target. Kick target being the desired position of robot's foot at the kick's climax (e.g., the ball) and step target being the position of the robot's foot at the end of the kick. Since the kick target is a point in mid air, we need a 3D vector, whereas the step target represents a point on the ground, so two dimensions are sufficient. This separation allows for both, kicks with the same start and end position, as well as kicks that seamlessly transition into the next step. In the context of this work, we will only use the first option. Originally, the kick target would always be set to a point, straight in front of the robot. Before the work presented in this thesis, adaptive kicks were implemented, with the kick target set to the ball position and the kick time dynamically computed based on the now varying kick and a newly introduced velocity. The kick time t_{kick} for a kick target $\vec{z}$ and velocity v is

$$t_{kick} = \frac{|\vec{z}|}{v}. \tag{4}$$

So rather than having a fixed kick time, we can just focus on the desired velocity. All of this is then formed into a step buffer element and added to the cognition step buffer. Following this, the step buffer gets executed, which leads to the creation of a motion request. This marks the end of both the configuration phase and the cognition part of the kicking process.

2.2.3 Execution Phase

Note: *This phase will not be covered as detailed as the previous two since it is not the main focus of this thesis. For a more in depth explanation, please refer to the 2019 Berlin United Team Report [17], especially Chapter 6: Motion Control.*

The execution phase starts with the reception of a motion request. The `footStepPlanner` module will then examine whether the requested step is a `walkStep`, `zeroStep` or a `kickStep`. Walk steps are just what the name suggests, zero steps, steps without any movement, to keep the robot stable and kick steps are the ones we are interested in. Further, in the same module, a walk request is created [1, 17]. In the walk request, we can choose from two interfaces: standard and step control. Step control is an extension to standard that allows us to further specify a step by selecting a foot to use and influencing the step's trajectory by e.g., setting the executions time [17]. The walk request is then added to the motion step buffer. From here on, the trajectory is computed, the hip motion planned and the zero moment point calculated [17]. Finally, the inverse kinematics are computed and the actuators controlled accordingly, which leads to the robot executing the kick.

2.2.4 Ballistic Phase

The ballistic phase starts once the kick is executed and the ball has left the robot's foot. The robot has no further control over the ball's trajectory and everything that happens now fully depends on the three previous phases. In this phase, the robot should keep track of the ball's position and move to a sensible, strategic position, to either intercept the ball or prepare for the next kick.

3 Parameterless Kicking

Our kicks rely on a set of externally defined parameters. This results in very limited flexibility and adaptivity, e.g., not allowing us to easily switch the `KickType`. This section's goal is to first evaluate how different parameters modify the resulting kick and then to either dynamically chose them, pass them through in the step buffer or to bundle all relevant parameter sets as kick actions, allowing for an easy switch at runtime. A good example for this is the kick height. It is first introduced in the trajectory computing part of the kicking process, i.e., execution phase. Since variations in kick height are interesting for the kick planning, this should be set in the `PathPlanner` module, such as the other variables of the configuration phase. As part of a step buffer element, this could be specified for different situations, much more conveniently and then simply being passed through to motion.

To explore the impact of different parameters on the robot's kicking performance, we conduct a number of testing series, with the goal to determine what parameter sets yield the best results in terms of kick distance and accuracy. The baseline for this work is the kick configuration used for the velocity study (Section 3.3), with a velocity of $0.45\,\mathrm{m\,s^{-1}}$.

3.1 Dynamic ZMP-Offset

One of the many parameters we manually have to set, when making changes on the kick is the Zero Moment Point (ZMP) kick offset. The ZMP kick offset indicates, how the robot's ZMP changes on the y-axis, while executing the kick and therefore compensates for the foot being lifted during the kick motion, to ensure the robot's balance. A pilot study revealed, that the ZMP kick offset highly relies on the kick time, while the influence of kick height or length are negligible. Bearing in mind that we use dynamically calculated kick times, it is easy to see why a fixed ZMP kick offset would be suboptimal. In another pilot study, we evaluated the best performing ZMP kick offset, for different, fixed kick times, ranging from 300 to 900 ms, in increments of 50 ms. For this we used ZMP kick offsets between -5 and $-20\,\mathrm{mm}$, in steps of 5 mm.

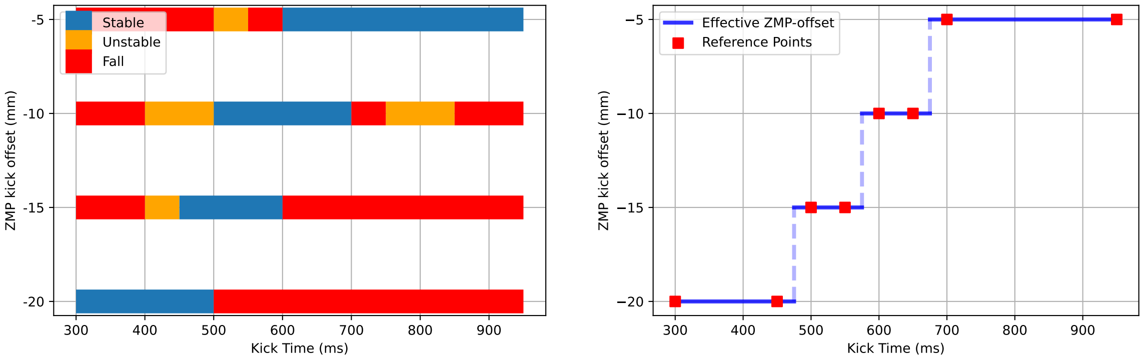
To qualify the kick's stability, we use the following classifiers:

Stable The robot remains stable throughout the entire kick motion.

Unstable The robot does not fall, but struggles to keep its balance.

Fall The robot falls while executing the kick motion.

For each combination of kick time and ZMP kick offset, we test as follows: If the robot falls, it is immediately classified as *Fall.* If the first kick is stable or unstable, we conduct two more kicks to evaluate the consistency. Based on these results, shown in Figure 5a, we can now derive a dynamic ZMP kick offset. For this, we expand the



(a) Kick stability for different kick times and ZMP kick offsets. The robot remained stable during the blue intervals, while orange indicates the robot struggling to keep its balance and red stands for a fall.

(b) Implementation of the dynamic ZMP kick offset, based on experimental data. The red reference points are defined by us and the blue line shows, what ZMP kick offset is assigned for a given kick time.

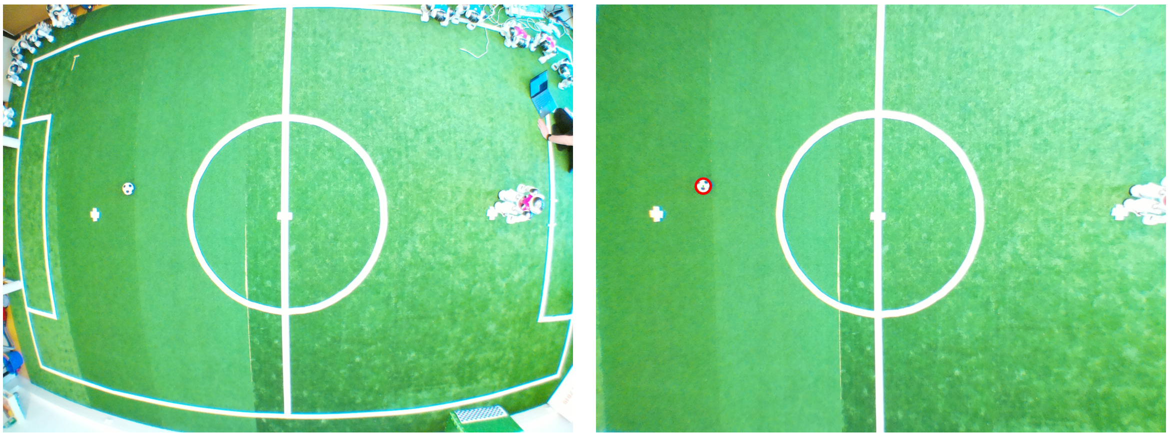
Figure 5: Results of the ZMP kick offset pilot study showcasing the robot's stability for kicks with different kick times. From this, we can derive the reference points and implement the dynamic ZMP kick offset.

existing `KickType` module with a set of reference points, i.e., pairs of kick time with the corresponding, best performing ZMP offset. The dynamic ZMP kick offset with reference points and the effectively assigned offset value for a given kick time, based on our experimental data, can be seen in Figure 5b. Furthermore, we implement a function that chooses the reference point with the smallest difference in kick time and returns the corresponding ZMP offset. This dynamic ZMP kick offset will be used in all further experiments.

In the experiments, described in this work, the experimentally determined dynamic ZMP kick offset, has worked consistently on all used robots, however, this should be validated in future work. Further research needs to also show how suitable these values are for other kick types and varying undergrounds.

3.2 Methodology

This section outlines the methodology employed to investigate the robot's kicking behavior, detailing the experimental setup and the procedures used for evaluation. Furthermore, we will introduce important terminology and discuss the limitations of our work.



(a) Raw view of the top down camera after a kick.

(b) Cropped, undistorted and automatically annotated image.

Figure 6: Process of capturing the kick's result using a top down camera. The ball end position gets automatically annotated and converted into real world coordinates, to calculate the kick distance and angle.

3.2.1 Experiment Setup

The experiments were carried out in our RoboCup lab, on a 4.35 m × 6.80 m soccer field. At the start of each test, the robot was placed at the center of one sides penalty area, facing the opposing goal. The ball was placed on the penalty mark, requiring the robot to take a couple of steps before kicking it. The robot would then walk straight towards the ball, and execute a kick, aiming for the opposing goal. To validate the results, a camera was installed on the ceiling, to work as an optical measuring system. Furthermore, a program was developed, which would capture a picture of the entire field, once the ball came to a halt. Using the circle Hough Transform, the program would automatically determine the ball's position, convert it to real world coordinates and then use these to calculate the distance and angle of the kick (as seen in Figure 6).

Most other papers, describing similar experiments, usually aim to eliminate as much uncertainty as possible, by letting the robot kick out of the stand, with the ball being placed at a fixed position, directly in front of it [18, 23, 25]. The choice of including a short approach to the ball, might therefore seem unorthodox and counterintuitive. This is why we, when first evaluating the kick performance in the preceded study project [7], went with the more conventional approach of letting the robot kick the ball out of the stand. However, we found that the results were not very representative of a real game situation, where the robot would almost always approach the ball before kicking it. The uncertainty of the robot's exact kicking position, as well as the momentum the robot builds up while approaching the ball, both have a significant impact on the resulting kick. Since our aim is to optimize the kick for real, dynamic and also uncertain game situations, not an artificial lab setting, we will include this uncertainty in our experiments.

3.2.2 Terminology

In order to better understand experiment's results and execution, we will introduce some important terminology. For this, we will differentiate between input and output variables.

Input Variables

Kick Target A point in the robot's local coordinate system, for which the kick is aimed at.

Ball Distance The distance between the robot and where the robot perceives the center of the ball, in mm.

Kick Time The time it takes to execute the kick, in ms. Either a fixed time, or dynamically calculated based on the kick target and velocity.

Kick Velocity The velocity at which the ball is kicked, in $\mathrm{m\,s^{-1}}$. This is used to dynamically calculate the kick time.

Kick Length The one-way travel distance of the robot's kicking foot, in mm.

Kick Height The maximum height of the kick, in mm.

Adaptive Kick For an adaptive kick, the ball position is set as kick target. Otherwise, the kick target is set to a fixed point in the robot's local coordinate system.

Output Variables

Kick Distance The travel distance of the ball, in cm.

Kick Angle The angle of the ball's travel direction.

3.2.3 Evaluation of the Results

As part of an experiment's evaluation and analysis, we will frequently encounter tables, such as Table 3. These are to be understood as follows:

The tables aim to rate kick distance, angle and consistency in range. They feature a number of statistical measures, such as mean, standard deviation, minimum and maximum value of the absolute kick distance, as well as the mean absolute angle. To compare the different test sets, we compute a score. For this, we perform a min-max normalization on every column, assigning the best sample in this column a score of 1 and the worst a score of 0. Depending on the feature, a higher or lower value is better, e.g. a higher mean kick distance is better, while a lower standard deviation is more

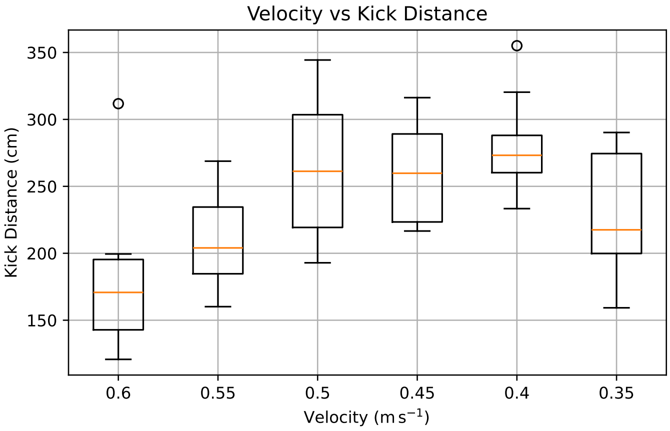


Figure 7: The difference in kick distance for different velocities, used to dynamically compute the kick time.

desirable. The samples in between are assigned a score in the range of 0 to 1, based on their distance to the best and worst value. Finally, we compute the average score for each sample, to get an overall rating. This is then multiplied by 100, for more intuitive understanding.

3.2.4 Limitations and Scientific Value

It was taken care to ensure the highest possible objectivity and comparability in the experiments. However, there are some factors one cannot control. While moving, the robot's joints will warm up. For kick motions this is especially important for the robot's knees. The joints temperature was closely monitored and cool down breaks included in the experiments, to ensure the robot's joints were always within a reasonable temperature range. Despite this, small variations in temperature could still have an effect on the kick's performance we were not able to control or measure. Since the experiments for this thesis were conducted during the summer months, the room temperature was also higher than usual, which might impact the robot's motion performance. Moreover, due to robots being absent for competitions or simply broken, unfortunately it was not possible to conduct all experiments with the same robot.

3.3 Velocity

The first series of tests aimed to determine the best performing kick velocity, as well as to evaluate how the kick distance and angle are affected by different velocities. We conduct six sets of tests, starting with a kick velocity of $0.6\,\mathrm{m\,s^{-1}}$, going down to $0.35\,\mathrm{m\,s^{-1}}$, with a step size of $0.05\,\mathrm{m\,s^{-1}}$. For each set, we collect 10 samples.

Velocity	Absolute Distance (cm)				Angle (°)	Score
	Mean	σ	Min	Max	Mean	
$0.6\,\mathrm{m\,s^{-1}}$	179.06	51.0	120.76	311.75	7.32	12.86
$0.55\,\mathrm{m\,s^{-1}}$	208.5	33.55	160.13	268.81	6.0	32.46
$0.5\,\mathrm{m\,s^{-1}}$	262.76	50.18	192.89	344.37	15.02	67.5
$0.45\,\mathrm{m\,s^{-1}}$	260.02	36.61	216.63	316.25	13.06	75.66
$0.4\,\mathrm{m\,s^{-1}}$	281.04	33.28	233.36	355.06	15.1	100.0
$0.35\,\mathrm{m\,s^{-1}}$	229.58	43.29	159.27	290.23	14.19	48.42

Table 1: Kick performance comparison by velocity.

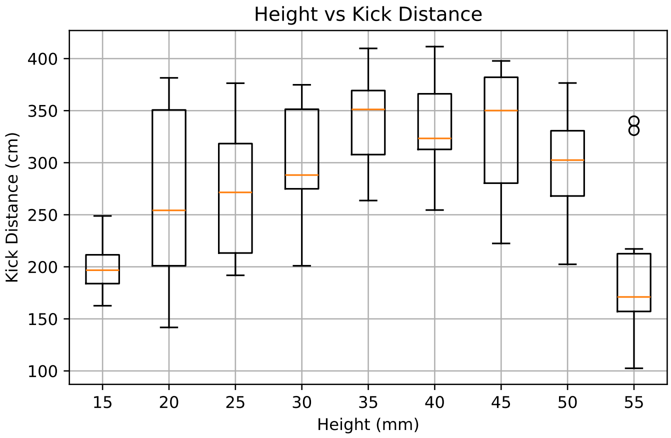


Figure 8: The impact of how high the robot lifts its foot while kicking, on the kick distance.

In Figure 7 and Table 1, we can see a clear performance peak between $0.5\,\mathrm{m\,s^{-1}}$ and $0.4\,\mathrm{m\,s^{-1}}$. For its highest mean and maximum kick distance, as well as the low variance in range, we select $0.4\,\mathrm{m\,s^{-1}}$ as kick velocity for the further experiments.

3.4 Kick Height

Since a good velocity was found, we continue the experiments by examining the kick height. We conduct nine sets of tests, starting with a kick height of 15 mm, going all the way up to 55 mm, in increments of 5 mm. For each set, we collect 10 samples.

In Figure 8, we can observe two interesting results. First, we can see a performance peak between 35 and 45 mm. The collected kick distances with a kick height of 35 and 40 mm have a very similar distribution but quite different medians. Since the distances achieved with a kick height of 35 mm have the highest mean, we select it as the kick height for the further experiments. But now for the second interesting result: The kick distance, achieved with a kick height of just 15 mm has are relatively low

Height	Absolute Distance (cm) Mean	σ	Min	Max	Angle (°) Mean	Score
15 mm	198.24	24.35	162.61	248.86	12.49	42.23
20 mm	265.13	82.97	141.79	381.48	14.53	50.48
25 mm	270.16	60.61	191.84	376.36	13.45	61.55
30 mm	299.91	52.29	201.0	374.81	11.03	63.06
35 mm	343.2	43.47	263.66	409.78	13.8	91.35
40 mm	332.94	42.62	254.52	411.51	13.97	89.76
45 mm	328.71	61.89	222.46	397.74	6.91	58.4
50 mm	294.96	50.26	202.41	376.55	9.98	60.7
55 mm	197.28	75.86	102.53	339.97	7.96	16.38

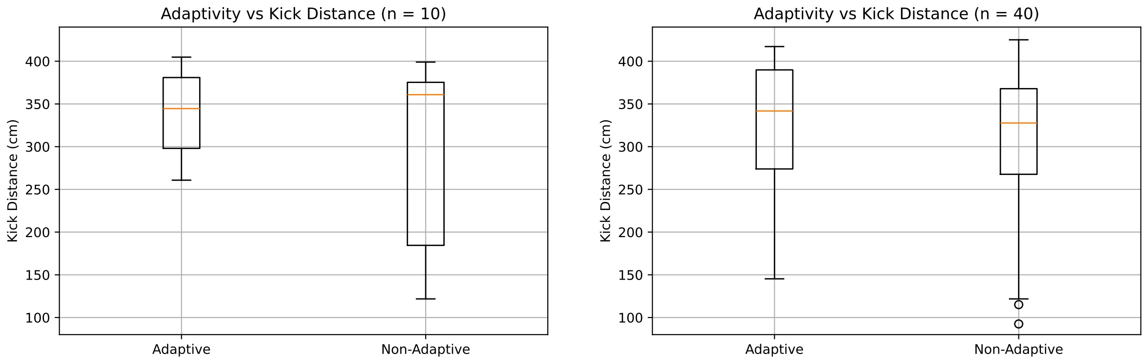
Table 2: Kick performance comparison by used kick height.

distance, with a mean of just under two meters, but also a remarkably low deviation, compared to the other tests. This makes the kick height of 15 mm a good candidate for a short, but very stable kick.

3.5 Kick Length and Adaptivity

Having found a good kick velocity and height, these parameters were evaluated in a test game. In this less controlled environment, we noticed that sometimes the robot would kick with a larger distance to the ball than set by the approach offset. This is, amongst other things, caused by the dynamic and uncertain nature of the soccer game. A ball that is still slightly moving while approaching, or after having reached the target, can result in these longer than planned kicks. During these situations, the robot would oftentimes struggle to keep its balance and sometimes fall, even with the dynamic ZMP kick offset. Moreover, these long and slow kicks, did not result in a good kick distance if the robot managed to reach and move the ball at all. To eliminate this issue, we make some adjustments to the kick length. As mentioned in Section 2, our robots usually kick adaptively, which means that the kick target is set to the ball's position. This allows the robot to adapt to the ball on both the x and y-axis, rather than just kicking straight forward, or to the ever-same point. To combat the balance issue, we try a mixed approach. The robot adaptively kicks towards the ball, but with a normalized kick vector scaled to a fixed length of 200 mm. This way, while kicking, the robot could keep some of its adaptability without engaging in too long kicks. Also, we assessed non-adaptive kicks to see whether adaptive kicking is really beneficial. So, we conduct two sets of pilot tests, one with adaptive kicks and one with non-adaptive kicks, both using a set kick length of 200 mm. For each set, we collect 10 samples.

The results of these pilot tests in Figure 9a, show that adaptive kicks in our sample set have a slightly lower median but also a much lower variance in kick distance, which



(a) Results for a sample set of $n = 10$ each.   (b) Results for a sample set of $n = 40$ each.

Figure 9: Kick Distance for adaptive and non-adaptive kicks with a fixed length. Adaptive kicking refers to kicking towards the ball, while non adaptive kicks, kick straight forward. In both cases a fixed kick length of 200 mm is used.

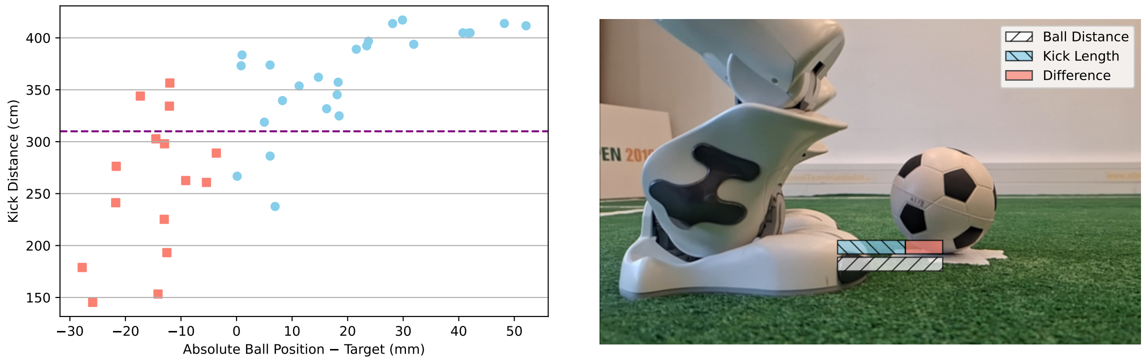
seems to make adaptive kicking the much better choice. Evaluating adaptive kicks with a fixed kick length in a test game, we observed that both falling and low performing kicks, caused by a too high ball distance were no longer an issue. To verify the results of our pilot test, we increase the sample size by 30.

Reviewing this larger data set, shown in Figure 9b, we can see a now similar performance in terms of kick distance for both adaptive and non-adaptive kicks. However, the adaptive kicks have a slightly higher median (341.78 cm, compared to 327.69 cm) and a lower, empirical standard deviation ($\sigma = 75.12\,\text{cm}$, compared to $\sigma = 87.93\,\text{cm}$). Therefore, we will not continue with the non-adaptive kick for further experiments.

Since all future experiments will use adaptive kicks, we will not mention the kick's adaptivity anymore. All further kicks not explicitly labeled as non-adaptive, can be assumed to be adaptive.

3.6 Kick Target Selection

Deeper analysis of the collected data revealed something interesting. Kicks, for which the ball position exceeded the 200 mm of the normalized kick length, had a significantly higher kick distance than kicks with the ball within the 200 mm. By just focusing on the difference d between the ball position and the kick length ($d = |\vec{b}| - 200\,(\text{mm})$) and comparing this to the kick position, we can see a strong trend: In Figure 10a, the red squares represent kicks, for which the ball was within the 200 mm of the kick length (i.e., with a difference of $d \leq 0\,\text{mm}$), while the blue dots represent kicks, for which the ball position exceeded the kick length (i.e., with a difference of $d > 0\,\text{mm}$). Drawing a line at a kick distance of 310 cm emphasizes the difference. This line separates the samples into two groups, with 88% of the blue samples above and 80% of the red



(a) Kick distance vs. difference between the kick target and ball distance. Red squares showing kicks with a negative, blue circles, kicks with a positive difference.

(b) Visualization of a positive difference between kick target and ball position.

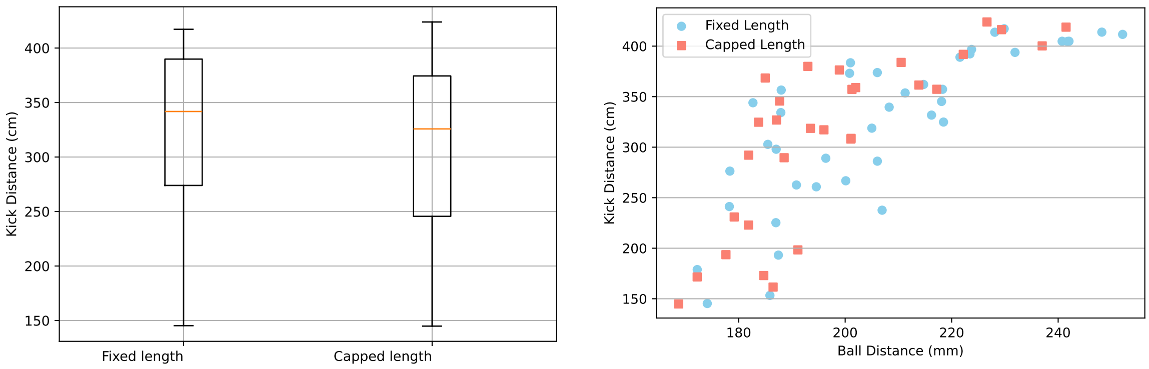
Figure 10: Visualization of the difference between the kick target and ball distance and its effect on the kick distance. A negative difference implies that the robot's foot surpassed the ball center during the kick. Analogous, a positive difference, as visualized in Figure 10b, indicates the foot not reaching the ball center.

samples below it. This leads to a key takeaway: The kick target should not exceed the ball's position. A fixed kick length is beneficial for the robot's stability, but not in terms of kick distance.

To overcome this issue, we try a hybrid approach. We keep the adaptive kick, but cap the kick length to a maximum of 200 mm. This means, if the ball is closer than 200 mm in the ball's direction.

Examining the results, especially the lower performing kicks for which the ball was within 200 mm, we can see that going from exceeding, to matching the ball position, was not enough to produce desirable results. Figure 11a even suggests a slightly worse performance for the capped length kicks although this could be due to the relatively small sample size. Figure 11b underlines the similarities. This calls for a different approach and a further going analysis of our so far collected data. Revisiting our fixed length experiments in Figure 10a, we can identify three interesting points on the x-axis, at 10, 20 and 30 mm, visualized in Figure 12. At these points, the kick distance seems to change its behavior:

10mm At a ball distance, exceeding the kick target by 10 mm, a noticeable shift in kick distance occurs, accompanied by a substantial reduction in variance. Samples between 10 mm and 20 mm behind the kick target have a standard deviation of just 13.51 cm with a mean of 345.8 cm, whereas samples between 0 and 10 mm have a standard deviation of 51.06 cm with a mean of 322.42 cm.



(a) (b)

Figure 11: Comparison of fixed (blue circles) and capped (red squares) kick lengths. Both kicks have a maximum length of 200 mm, with the capped kicks, reaching to the ball center for smaller distances.

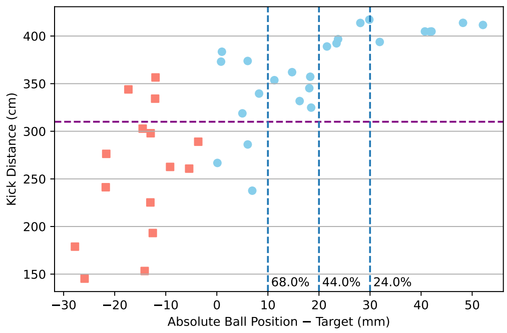


Figure 12: Interesting points in kick distance. The vertical lines indicate trend changes in absolute kick distance, with percentages showing what portion of good performing kicks is on the line's right hand side.

20mm At a ball distance, exceeding the kick target by 20 mm, we can see a clear trend towards the 400 cm mark. Also, right around this point of 20 mm, we can see a shift in the kick distance. The standard deviation in the section between 20 and 30 mm is similarly low, with 11.47 cm with a mean of 401.8 cm, showing a clear leap in kick distance.

30mm At a ball distance, exceeding the kick target by 30 mm, we can see a stagnation in the kick distance. Kicks with a difference of 30 mm or more have a standard deviation of just 6.4 cm with a mean of 405.59 cm.

In Figure 12, these three points are marked and feature a percentage value, indicating how many of the good performing kicks (i.e., kicks with a kick distance of more than 310 cm) are on the right hand side of the respective point. In other words, how many of the good performing kicks have a ball distance exceeding the kick target by at least n mm, with n being 10, 20 or 30 mm. This leads to the assumption that kicks where the kick target is offset in such a way, that the ball distance exceeds the kick target by 10 to 30 mm, could solve our problem. In Figure 12, kicks executed with a ball distance, exceeding the kick target by at least 10 mm had the highest share of good performing kicks. Therefore, it is likely to find a descending trend in range, with an increasing offset. To test our assumption, we implement a Kick Target Offset (KTO), that allows us to offset the kick target by a set value, while still keeping the original direction towards the ball. We retain the cap on the kick length at 200 mm, as this has been shown to improve both the robot's stability and the achieved kick distance. Therefore, the kick target is computed as follows:

1. The kick target $\vec{k}$ for a relative ball position $\vec{b}$ is computed as

$$\vec{k} = \frac{\vec{b}}{|\vec{b}|} \cdot (|\vec{b}| + n), \tag{5}$$

   with n being the selected KTO.

2. If $|\vec{k}| > 200\,\text{mm}$, we set

$$\vec{k} = \frac{\vec{b}}{|\vec{b}|} \cdot 200\,(\text{mm}), \tag{6}$$

   to cap the kick length at 200 mm.

We conduct a series of tests with KTO values of −10 mm, −20 mm, −30 mm and −50 mm.

Length Type	Absolute Distance (cm)				Angle (°)	Score
	Mean	σ	Min	Max	Mean	
Fixed	323.95	75.13	145.29	417.19	14.51	61.34
Capped	310.76	83.34	144.86	423.91	11.6	50.3
KTO, -50 mm	257.3	92.27	94.88	362.29	6.82	2.3
KTO, -30 mm	300.28	99.4	112.89	435.59	10.88	40.98
KTO, -20 mm	339.95	69.39	175.5	412.13	11.05	59.17
KTO, -10 mm	358.13	37.33	284.49	403.14	7.91	73.98

Table 3: Kick performance comparison by length type.

In Figure 13 and Table 3, we can see the results of these tests, compared to the previous fixed and capped length (i.e., 0 mm KTO) kicks. It shows that our assumption was correct. Kicks with a KTO of −10 mm perform best, while the performance decreases with a larger offset.

Comparing the best performing −10 mm KTO kicks to our previously examined fixed length kicks in Figure 14, reveals what we have been aiming for. The −10 mm KTO kicks outperform the fixed length kicks in the low ball distance range. To put a number

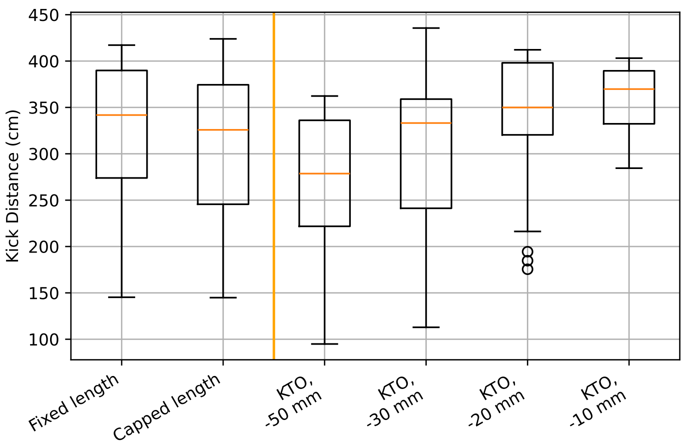


Figure 13: Comparison of different kick length types. With the fixed and capped length kicks as reference. The capped length configuration is equivalent to kicks with a KTO of 0 mm.

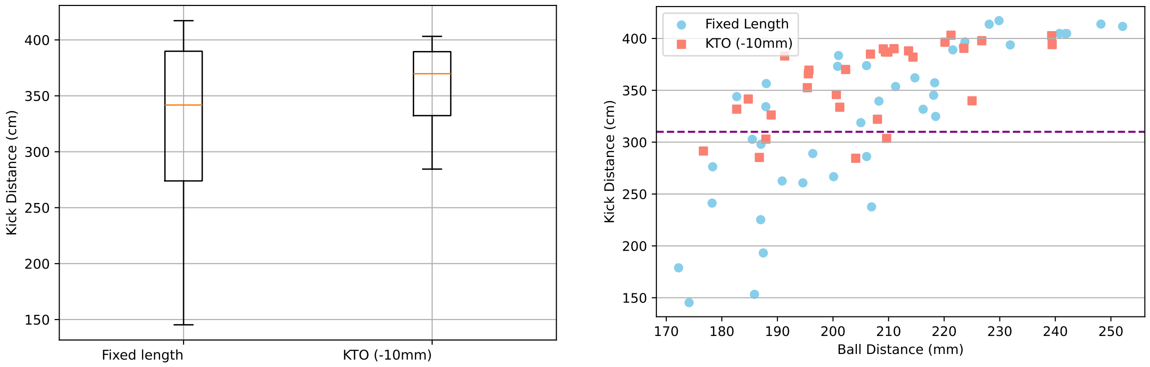


Figure 14: Comparison of fixed length and -10 mm KTO. We can see that the -10 mm KTO configuration yielded better results for kicks with smaller ball distance while keeping the good performance for balls further away.

on it, 83.33 % of the -10 mm KTO kicks ended up above the previously used threshold of 310 cm, while only 62.5 % of the fixed length kicks did so.

3.7 Ball Distance

In the previous experiments, we have focused on the configuration phase. In this phase, we have to deal with whatever the alignment phase has set us up with. We cannot impact the robot's position, relative to the ball, or its orientation in the field anymore. This calls for an adaptive kicking ability, inclusive to a range of ball distances, as we have discussed. While exploring this, we have seen that the kick distance is directly related to the ball distance, but we are yet to evaluate the impact of different ball

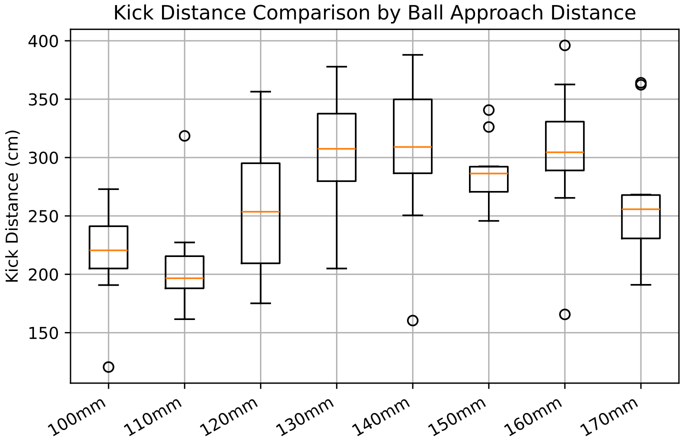


Figure 15: The impact of the robot's approach distance (see Equation (1)) on the resulting kick.

distances on the overall performance of the kick. To dive deeper into this, we first need to understand how the target distance is determined, while approaching the ball.

As defined before (see Section 2), we have a 2D vector $\vec{b}$ representing the ball's position in the robot's local coordinate system, the approach offset $\vec{o}$, the allowed deviation $\vec{d}$ and the target position $\vec{z}$. Our default values, which were used in the experiments so far, are $\vec{o} = (120, 0)$, $\vec{d} = (50, 30)$ and $z_1 = b_1 - r - o_1$, for a ball with radius r and $z_2 = b_2 - o_2$. As soon as $z_1 < d_1$ and $|z_2| < d_2$, the robot will execute the kick. In simpler terms, offset $\vec{o}$ is how we influence the robot's distance to the ball, while the deviation $\vec{d}$ is the precision with which the robot has to approach a target position $\vec{z}$.

To gain a better understanding of how different ball distances affect the kick performance we conduct a series of tests, with $o_1 \in \{100, 110, ..., 170\}$, and an allowed deviation of $\vec{d} = (10, 30)$. This results in an allowed ball distance of 160 to 230 mm on the x-axis. The results of this test series are portrayed in Figure 15 and Table 4.

Approach Distance	Absolute Distance (cm)				Angle (°)	Score
	Mean	σ	Min	Max	Mean	
100 mm	217.41	39.84	120.57	272.93	6.51	14.51
110 mm	208.1	40.88	161.57	318.54	7.31	27.3
120 mm	257.6	57.38	175.15	356.42	6.74	35.28
130 mm	303.8	48.99	204.96	377.81	9.6	62.8
140 mm	304.87	61.46	160.38	387.94	13.1	56.22
150 mm	287.41	27.1	245.74	340.66	12.7	77.89
160 mm	302.94	58.41	165.65	396.06	12.28	58.37
170 mm	263.15	56.65	190.99	364.03	18.3	60.23

Table 4: Kick performance comparison by approach distance.

3.8 Kick Actions

As a great finale to this and prelude to the next chapter, we will now bring all of our previous insights together and implement them as kick actions. This time, we will focus purely on the configuration and execution phase, but we will revisit and amend information, necessary for the alignment phase and the predictive action model, i.e., the whole influenceable kicking process. We first need to decide, what to feature in our preliminary kick action and what not. Since it did not prove to be beneficial, we will not include an option for non-adaptive kicks. Experimental options that we did not find to be an improvement, such as fixed kick lengths, will not be included either.

Preliminary `KickAction` module:

Mandatory:

- Kick Type
- Kick Velocity
- Kick Height

Optional: (if not specified, the default values will be used)

- Max Kick Length
- Kick Target Offset
- Ideal Ball Distance

3.9 Summary

With the improvement oriented studies presented in this section, we were able to increase the robot's kicking performance by 37.98 %, or 98.76 cm in absolute kick distance, compared to the baseline configuration. The best results were achieved with a KTO of −10 mm. Despite the advance in performance, the standard deviation in distance remained almost the same ($\sigma_{old} = 36.61\,\text{cm}$, $\sigma_{new} = 37.33\,\text{cm}$). The difference between these two kick configurations, as well as their distribution and mean value on the field, can be seen in Figure 16.

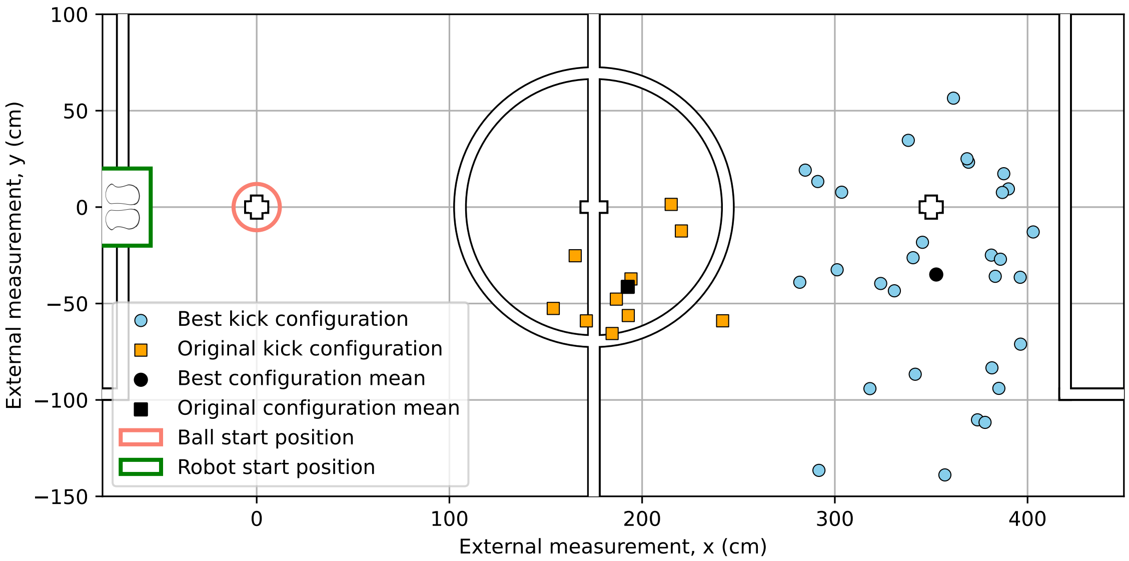


Figure 16: Comparison of the best performing kick configuration in this thesis (blue circles), with the one used before (orange squares). The mean kick is indicated by a black marker of the respective shape.

4 Target Prediction and Simulation Correction

The first thing we do when we are at a competition, is to calibrate the robots camera, so it can see well in the new lighting condition and orient itself on the field. Our forward simulation based kick action selection, however, relies on values that were once determined experimentally, not taking into account changes in the kick motion or variations in the field's carpet. This can lead to the robot having no idea how far the ball will travel, causing too short or too long kicks, sending the ball out of bounds or missing the goal. This section is built around three main questions:

1. How can the robot autonomously adjust its action prediction model (APM)?
2. How can the APM be improved in accuracy?
3. How can we decide when to end the approach and execute a kick?

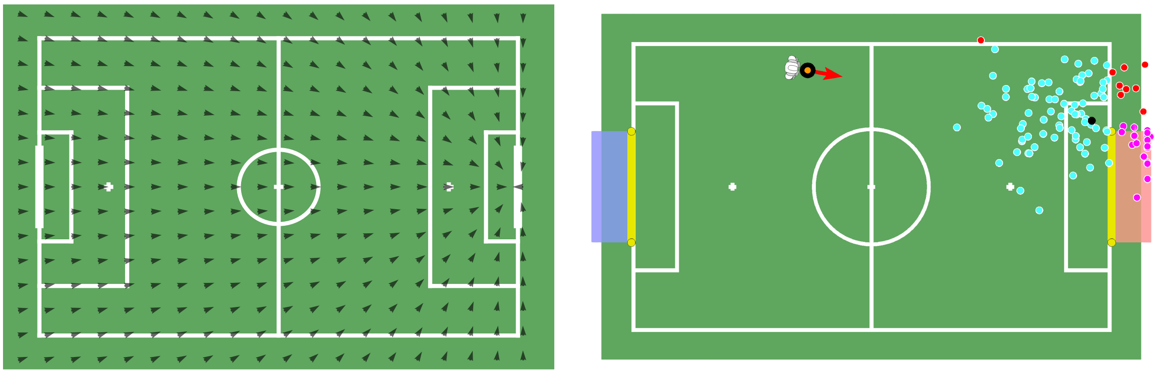
To commence with the experiments in this section, we first need to collect a data set to work with. For this, we can reuse our existing test setup, as described in Section 3.2.1, with the addition of the robot measuring the kick distance itself. In order to get a good coverage of the approach positions, we will randomly select the approach offset using Mersenne-Twister [12]. At the start of each test, the robot therefore randomly select an approach offset to the ball in the range of [100 mm, 180 mm] on the x and [−10 mm, 10 mm] on the y-axis. In addition to the generally collected data, such as kick time and target, we will also log the robot's perceived ball position, as well as the robot's position and orientation on the field. With this setup, we will conduct a total of 100 kicks to collect data for our further experiments.

For the robot to autonomously adjust its APM, we will examine the reliability of the robots ball model to be able to verify whether an executed kick belongs to the APM's distribution. As a way to improve the APM's accuracy, we will investigate the influence of different ball positions on the resulting kick, to evaluate whether this should be included in the APM. In order to decide when to kick, we will introduce a time-based and a probabilistic approach.

4.1 Integration of Kicking in Decision Making

In the first sections of this thesis, we have taken the robot's decision to kick the ball for granted. To change this and prepare for this section, we will now assess, how the kick behavior primitive is integrated in the robot's decision making process.

If a robot perceives itself as striker, it will walk towards the ball, oriented according to the attack direction [16, 21]. The attack direction works as a guideline and provides a strategically valuable kick direction for every possible ball position on the field. This is implemented as a vector field, visualized in Figure 17a. While approaching the ball, the robot constantly runs a forward simulation, predicting a set of 30 possible ball



(a) The attack direction vector field. (b) Action simulation with the best action.

Figure 17: Process of finding the best action. The vector field in Figure 17a, provides a strategically valuable orientation fo every possible ball position on the field. The forward simulation in Figure 17b, predicts 30 different outcomes for a kick action (shown as blue, red and pink dots), evaluating the possible yield of the current orientation.

positions after being kicked from this direction for all, available kick actions, as seen in Figure 17b. The simulation relies on a friction model and predicts the ball's end position, based on the experimentally determined average ball speed and angle, and their respective standard deviation, for each kick action. For a set of experimentally determined kick distances $d = \{d_1, ..., d_n\}$ (in mm), friction coefficients c and gravity g (in $\mathrm{mm\,s^{-2}}$), we can calculate the average ball speed v_{ball} (in $\mathrm{mm\,s^{-1}}$) after being kicked as follows:

$$v_{ball} = \sqrt{2 \cdot c \cdot g \cdot \bar{d}}, \text{ where } \bar{d} = \frac{1}{n} \cdot \sum_{k=1}^{n} d_k \text{ is the sample mean.} \tag{7}$$

Prior to this thesis, kick actions were rather a concept than an actual object; a set of four different kicks with semi hardcoded simulation parameters for each action. Out of the four available, we only use `forwardKick.short`. This is i.a., due to the missing implementation of side kicks. Therefore, currently our decision making will only decide whether to kick or not, rather than which kick action to use.

Having this particle cloud of possible ball positions, the robot can now evaluate the situation and make a decision when to start the approach. In Figure 17b, we can see in the visualization that the simulated balls can have one of three different characteristics, displayed as colors: red being an illegal or bad ball position such as a ball leaving the field or an own goal, blue being a legal infield position and pink being in the opponent's goal. Based on this information, the robot can now decide whether the current orientation is good and what kick action to use. To achieve this, the Berlin United codebase [1] contains a function called `decide_smart()`, that works as follows:

1. If one or more simulated balls are in the own goal, this action will be ignored.

2. The robot will then calculate a probability score s to evaluate the situation:

$$s = \mathsf{P}(\text{ball infield}) + \mathsf{P}(\text{ball in opp. goal}) \tag{8}$$

3. If $s \geq 0.85$, this action is considered acceptable and will be pushed back for further evaluation.

4. If only one acceptable action is found, the robot will return this as the best action. If none is found, the robot will continue turning around the ball.

5. If multiple acceptable actions are found, we will look for goal actions. A goal action is every action with $\mathsf{P}(\text{ball in opp. goal}) \geq 0.3$, i.e., more than 30% of the simulated balls in the opponent's goal.

6. If a goal action with a higher probability to score is found, the previously selected action will be overwritten. Hence, at every point we will only have one goal action at most.

7. If only one goal action is found, this will be returned as the best action. If none is found, we will select one of the acceptable actions, based on their strategic value.

4.2 Autonomous Simulation Adjustment

Adjusting the APM autonomously requires a few preparations. First, the robot needs to be able to qualify its own kicks. In Section 3, we have established distance, angle and deviation as the main quality metrics of a kick. The deviation is highly dependent on the sample size, but measuring the distance and angle of just a few kicks can already give a good indication about the accuracy of the APM's distribution. To realize this, we need an online kick distance measurement, that allows the robot to measure the distance and angle of its own kicks without the need for external hardware. The results of this measurement can then be used verify or disprove and in that case adjust the distribution, the APM is based on.

4.2.1 Online Kick Distance Measurement

Luckily for us, the robot already possesses a ball model that estimates the ball's position relative to itself based on the perceived size of the ball [14]. To utilize this model for measuring the distance of a kick, we need to determine a kick's start and end. Deciding on the ball's start position is trivial since we already do this for every kick when setting the kick target (see Section 3.5). The easiest way for getting the ball's end position and therefore deciding when it has come to a rest, would be to manually send a signal to the robot, once we notice that the ball has stopped moving. Having the robot do

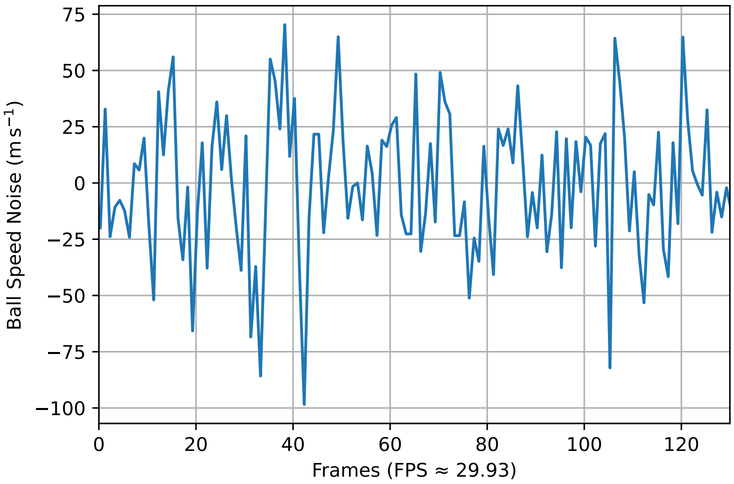


Figure 18: Noise in the ball model's velocity estimation on the robot's x-axis, at a distance of 1 m, with the ball at rest.

this autonomously, however, would be more elegant and less prone to human error. Letting the robot wait for a fixed amount of time after the kick, before assessing the ball's position, would be a simple solution, but also not be very robust, since this could either cause premature measurements, for longer or unnecessary waiting time for shorter kicks. Hence, we will combine this with a velocity check. Thankfully, the ball model already estimates the ball's speed as well. Since most kicks will mainly travel on the robot's x-axis, we only check the ball's velocity in this direction. The perceived ball model has a certain amount of noise, which causes a standard deviation of $31.51\,\mathrm{mm\,s^{-1}}$ in the velocity estimation on the x-axis, when the ball is at rest (see Figure 18). Therefore, we will use a threshold of $100\mathrm{mm\,s^{-1}}$ to determine whether the ball is still moving or has come to a halt. To avoid premature measurements and allow for uncertainty, caused to the robot's movement during the kick, we will only start checking the ball's velocity after 100 frames (≈3.3 s), following the kick's execution.

Having collected the start and end position of the ball, we can easily compute the kicked distance and angle. This way, we can take three different types of kick distance measurements:

Local Kick Distance: As described above, we only use the ball position in the robot's local coordinate system, prior and after the kick, to determine the kick distance. This does not take the robot's movement during the kick into account.

Global Kick Distance: To get this, we also have to log the robot's perceived position on the field, prior and after the kick. Having this, we can transform the local ball positions into global coordinates and then determine the kick distance. This way, we can take both movement and orientation changes of the robot into account. However, flaws in the self-localization can lead to errors in measurement.

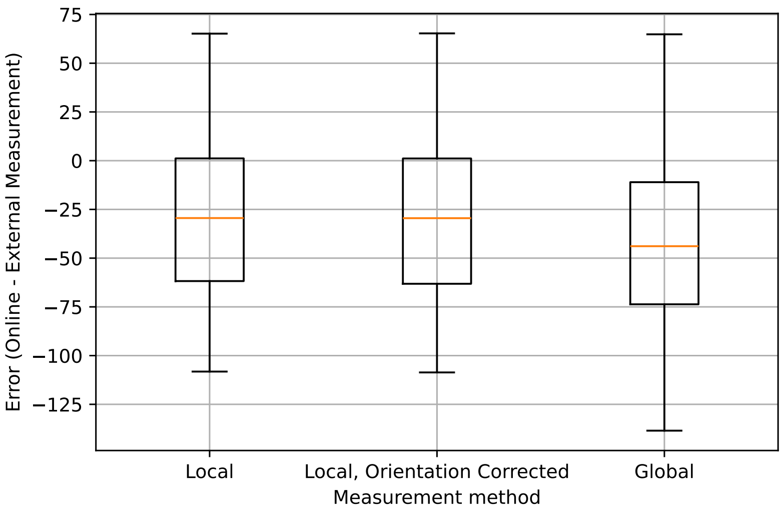


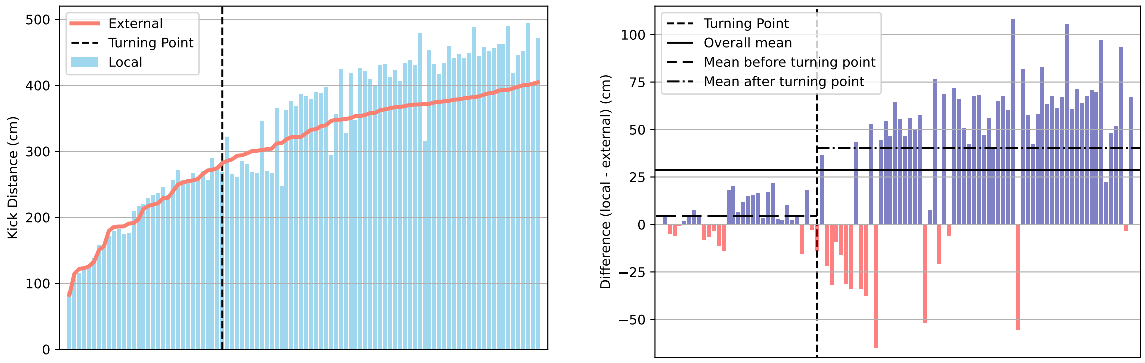
Figure 19: Error comparison of the different online measurement methods, compared to the external camera measurement.

Orientation-Corrected Kick Distance: This is a compromise between the two previous methods. Here, we only take the robot's orientation change into account, but not its movement. This way, we can at least correct for the robot turning during the kick, which could otherwise lead to a wrongly perceived kick angle. While kicking, the robot's orientation changes more than its position, so we can account for the biggest error source, without being affected by self-localization errors.

4.2.2 Reliability of the Online Measurement

To evaluate the reliability of the online kick distance measurement, we can compare it our external camera measurement, which we consider to be the ground truth. Comparing all three methods to the external measurement (see Figure 19), we find that the local measurement has the lowest error, with an mean absolute error (MAE) of 39.15 mm (compared to 47.22 mm for global and 39.26 mm for local, orientation corrected measurement).

Up until now, the robot's ball model was only used for navigating towards a ball further away, or approaching closer ones with the intent to kick them. Both of these use cases did not need a precise distance estimation for balls outside the robot's near space. This was also evident, in our experiments. When matching the online measured values to their respective external counterparts, we observe a bias towards overestimating the kick distance for balls being further away (see Figure 20). The mean online measured distance exceeds the actual mean distance by 28.59 cm or 9.41 %. Taking a closer look at Figure 20a, we can see that up until a distance of just below 300 cm (marked by



(a) Local measurement, compared to the external camera measurement.

(b) Difference between local and external measurement.

Figure 20: Local vs. external kick distance measurement. Sorted ascending by external measurement. As we can see, the robot's ball model provides accurate measurements, for ball distances up to ~2.7 m, but tends to overestimate the distance of balls further away.

a black, vertical line), the value set on the line's left hand side has an MAE of just 9.11 cm, whereas the ones on the right has an MAE of 53.48 cm.

4.3 Refined Action Prediction

Currently, the APM predicts the outcome of a kick action, only from the mean ball speed and angle, and their respective standard deviations (see Section 4.1). To decide whether this is sufficient to reliably predict kick actions, we will assess the impact of how the ball is positioned, relative to the kicking foot, on the performed kick. In Section 3.7, we have already shown that anterior distance to the ball matters. Now, for determining the effects on both axes, we will analyze the results of our most recent sample set, both at its own and supplemented with the results from Section 3.7. Analyzing Figure 21, we can see that the x-axis offset has the bigger impact on the kick distance, at least for the recorded y-axis range. We also notice that, as expected, the ball travels further, the greater the initial distance was, up until the robot can hardly or not at all reach the ball.

The impact of the ball's position on the robot's y-axis, was assessed by Wege [23], finding it to influence the mean kick distance substantially. However, the evaluated ball positions, shown in their work (60-200 mm, on the y-axis, relative to the supporting foot), are much higher than what we have recorded during our experiments, with a maximum distance of 32.09 mm for regular approach settings and 41.00 mm while using randomized approach targets. This limits the meaningfulness of Wege's finding for our usual kick behavior primitive.

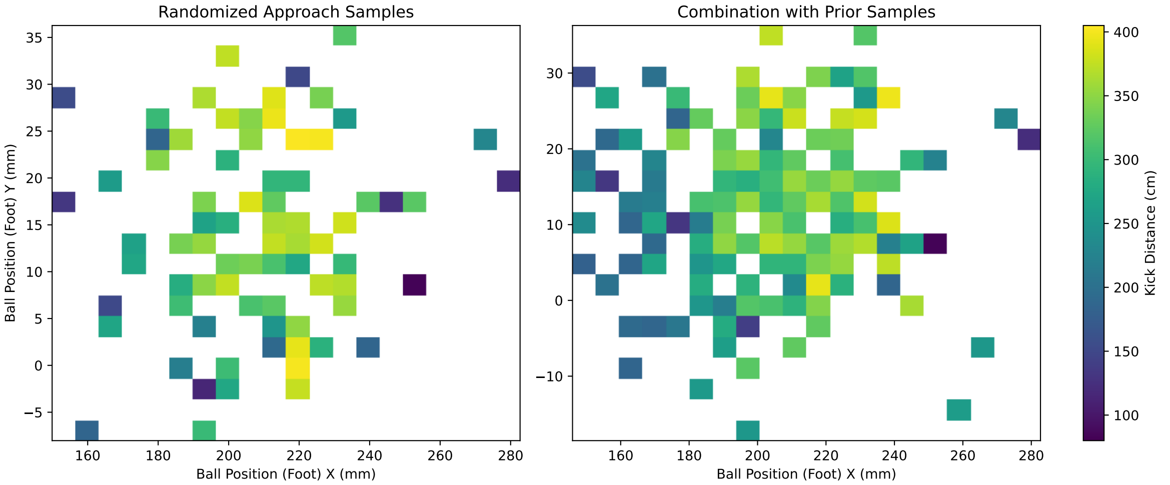


Figure 21: Heatmaps showing the kick distances for different ball positions, right before kicking the ball.

Nonetheless, we come to the conclusion that taking the ball position into consideration for predicting kick actions, could increase the APM's accuracy.

4.4 Prediction Based Kick Decision

As probabilistic way to decide on the best moment to kick, we will develop a new predictive model, to constantly assess the current approach position. For this, we will evaluate, how precise we can predict a kick from what the robot is aware of while approaching the ball, using different machine learning methods. We aim to end the approach and execute the kick, once the model has predicted the kick to be of strategic value. This could be both, a threshold of minimum distance being surpassed or, the ball being predicted to end up in a beneficial area, such as the opponent's goal.

4.4.1 Kick Prediction

Features and Labels:
To predict the kick, we can make use of the following features:

- Ball Position
- Kick Target
- Kick Time

For the ball start position, we can use either local or global coordinates, both in the coordination system of the robot's hip (i.e., center) or its non kicking foot. All coordinate features can be represented as vector $\vec{v}$ or scalar values v_1, v_2 or $|\vec{v}|$.

As labels, we will use the kick distance and angle as measured by the external camera. Due to its previously discussed bias and general lack of reliability, online measurement has shown to underperform the external system for predicting the kick's outcome, as assessed in a pilot study. Since we cannot utilize the robot's ball model to verify the prediction in the highly uncertain environment of a soccer game, using the external data is sufficient. To predict kicks, we will probe both, single features, as well as feature sets. A pilot study, aiming to identify the most promising feature sets, revealed, that pairing the positional features with the kick time, resulted in the highest prediction accuracy.

Model Training and Evaluation:
To train and evaluate our prediction models, we will use a 4-fold cross validation, with random shuffling of the previously mentioned 100 samples. Given our limited computational resources, the models need to be kept as lightweight as possible. Therefore, we employ the following supervised learning approaches:

- Linear Regression
- k-nearest neighbor (k-NN)
- Multilayer perceptron (MLP)

We will evaluate the models with their MAE.

For k-NN we set $k = 7$ and use uniform weights, as this has worked best, in initial tests. For the MLP, we use two layers with 16 and 5 neurons and ReLU as activation function.

Preprocessing:
Due to their nature, some of the models are more sensitive to feature scaling than others. Therefore, in addition to the raw features, we will also evaluate the models on standardized and normalized features.

For this we will use the following methods:

Standard Scaling: For each feature x_j, with values x_{ij}, mean μ_j and standard deviation σ_j, we compute the standardized value z_{ij} as:

$$z_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j} \tag{9}$$

With standard scaling applied, all features will have a mean of 0 and a variance of 1.

Min-Max Normalization: For each feature x_j, with values x_{ij}, minimum $\min_j$ and maximum $\max_j$, we compute the normalized value n_{ij} as:

$$n_{ij} = \frac{x_{ij} - \min_j}{\max_j - \min_j} \tag{10}$$

With min-max normalization applied, all features will be in the range of [0, 1].

Max-Abs Normalization: For each feature x_j, with values x_{ij} and maximum absolute value $\max_j^{\text{abs}} = \max(|\max_j|, |\min_j|)$, we compute the normalized value a_{ij} as:

$$a_{ij} = \frac{x_{ij}}{\max_j^{\text{abs}}} \tag{11}$$

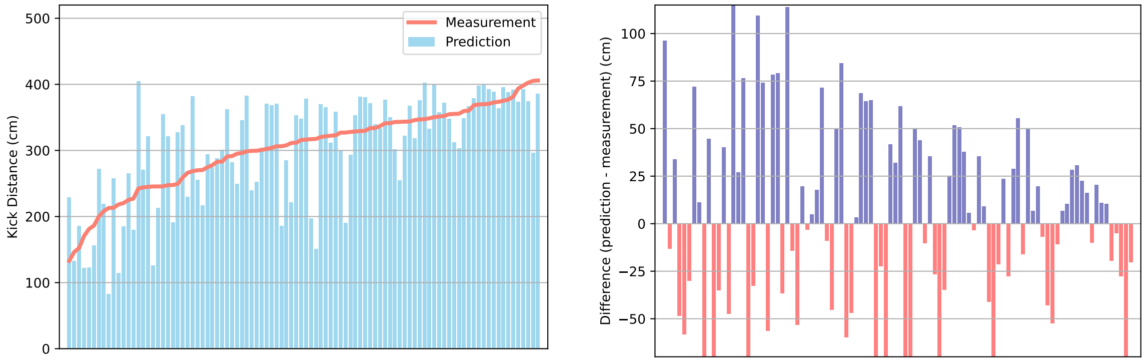
With max-abs normalization applied, all features will be in the range of [-1, 1].

4.4.2 Evaluation

Best single features for predicting distance.			
ML-approach	Feature	Normalization	MAE
Linear Regression	Kick Target	Irrelevant	60.56 cm
k-NN	Ball Position (Foot)	None	57.56 cm
MLP	Ball Position	Max-Abs	56.28 cm
Best feature sets for predicting distance.			
ML-approach	Feature Set	Normalization	MAE
Linear Regression	(Ball Position, Kick Time)	Irrelevant	46.46 cm
k-NN	(Ball Position, Kick Time)	None	52.99 cm
MLP	(Ball Position, Kick Time)	Min-Max	47.17 cm
Best single features for predicting angle.			
ML-approach	Feature	Normalization	MAE
Linear Regression	Kick Target	Irrelevant	0.1388 (7.95°)
k-NN	Ball Position, y-axis	Irrelevant	0.1412 (8.09°)
MLP	Ball Position, y-axis	None	0.1396 (8.00°)
Best feature sets for predicting angle.			
ML-approach	Feature Set	Normalization	MAE
Linear Regression	(Ball Position, Kick Time)	Irrelevant	0.1385 (7.94°)
k-NN	(Ball Position (Foot), Kick Time)	Max	0.1443 (8.27°)
MLP	(Kick Target, Kick Time)	Max	0.1421 (8.14°)

Table 5: Overview of the best features and feature sets with respective learning methods, labels and applied normalization.

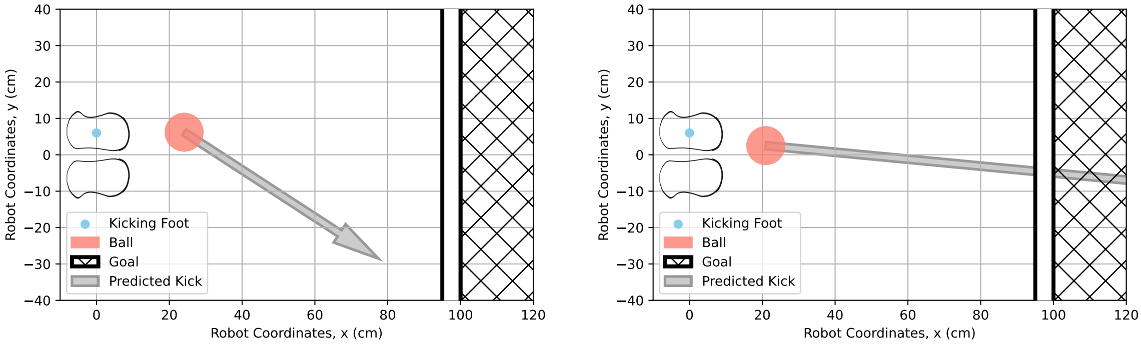
In Table 5, we can see that from the three methods we tested, linear regression has the lowest MAE, in predicting both distance and angle. Comparing the predicted kick distances to the actual measurements and examining the error of the single predictions, as visualized in Figure 22, we can see frequent, substantial deviation from the actual



(a) Predicted kick distances, compared to the actual measurements.

(b) Difference between prediction and actual measurement.

Figure 22: Measured kick distances vs. predictions using linear regression with the best performing feature set from Table 5. Sorted ascending by measurement.



(a) The predicted kick is very short, with a high angle and will likely miss the goal. The robot will continue with the approach and not yet execute a kick.

(b) The predicted kick is long, with a low angle and will likely score a goal. The robot will end the approach and execute a kick.

Figure 23: Predicted kicks, based on different ball positions and their potential of scoring a goal.

measurements, with 21.88 % of the prediction errors being greater than the MAE. Future experimentation, aimed to increase the sample size needs to reveal whether a more accurate predictions can be achieved, or if the randomness and uncertainty produced by the friction and unevenness of the underground has a too high impact on the kick's outcome.

Even though, our predicted values are not as precise, as we would wish them to be, we can still use them to simulate the functionality of the prediction based kick decision and perspectively implement it on the robot. In Figure 23, we can see two examples, In which the current distance to the ball is used to predict how far the it would travel, if the approach was ended and the kick executed from this very position.

4.5 Time-Based Kick Decision

As time-based option to decide, when to execute a kick, we will introduce a novel approach, which we will refer to as Progressive Deviation Control (PDC).

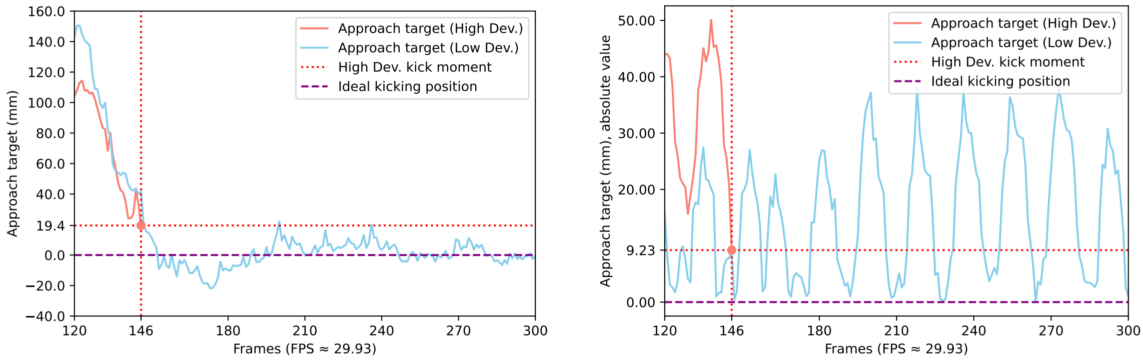
4.5.1 Motivation

As mentioned before, for every kick approach, we set an allowed deviation $\vec{d}$ from the target position $\vec{z}$. To demonstrate this, we let the robot approach and kick the ball with the current default deviation of $\vec{d} = (50, 30)$ and then with a lower deviation of just $\vec{d} = (10, 10)$. A lower deviation in the approach results in a lower deviation in kick distance. Therefore, it seems reasonable to decrease the allowed deviation, to improve the robot's kicking. Here, we encounter a new metric to evaluate our kick performance: time.

Figure 24, displays the results of these two approaches, both for the x and y axis.

The reader might notice that there is only a moment of kick for the high deviation approach. This is neither neglect, nor a sign of no kick being executed, but rather indicates the ridiculously long time it took to complete the kicking process. While the low deviation approach took just 147 frames, or ∼4.9 seconds, the high deviation approach took a staggering 1668 frames, or ∼55.8 seconds to complete. Yet, we can observe something interesting with the low deviation approach: starting shortly after frame 150 in Figure 24a, the robot has already reached the target position and spends the remaining ∼50 seconds just treading in place, waiting for both dimensions to be within the allowed deviation and also to be physically able to kick the ball.

So, due to the lower deviation, the robot has reached a much better position in just around 5 more frames and therefore could have kicked the ball with a much higher precision at almost the same time, but instead waited for its position to get even better. This shows the limitations of the static threshold, the current approach is based on.



(a) Comparison of the approach deviation on the x-axis.

(b) Comparison of the approach deviation on the y-axis.

Figure 24: Comparison of two approaches with a low and high allowed deviation.

To overcome this, we introduce PDC. The core idea of PDC is to start the kicking approach with a very low allowed deviation, such as 10 mm, and then to progressively increase the allowed deviation, as the robot approaches the ball. This way, the robot aims for a low deviation to the target position and while doing so, the allowed deviation is increased, so that the robot can kick the ball at a much earlier point in time, while still being able to kick it with a high enough precision. In other words, we can decide on how long the robot should take to approach the ball and with PDC it will achieve the highest precision possible within this time frame.

For the lion's share of a soccer game, time is one of the most crucial resources. Being quick with kicking the ball is sometimes even more important than achieving great distances or accuracy, because a too slow kicking process can result in not being able to kick the ball at all. However, there are situations, where time is not an issue, such as the kick off, free kicks or penalty kicks [3]. In these situations, the ball cannot be kicked by the opponent, within a certain time frame at least, so the robot could take its time to approach the ball and kick it with higher precision.

4.5.2 Functionality

For a PDC approach, we need a start and an end deviation $\vec{\mathcal{D}}^{start}$ and $\vec{\mathcal{D}}^{end}$, as well as a approach duration, specified by $\vec{\mathcal{T}}^{start}$ and $\vec{\mathcal{T}}^{end}$. Furthermore, we need a function $f_k(t)$ that interpolates the allowed deviation over time between our start and end values. For the interpolation, we borrow the concept of easing functions [19] from computer graphics. In computer graphics, easing functions are used to describe the character of how an animation progresses over time. They are functions, defined on the interval $[0, 1]$, which take a value $t \in [0, 1]$ and return a value $f_k(t)$ on the same interval. This characteristic makes easing functions very suitable for our use case. We can utilize

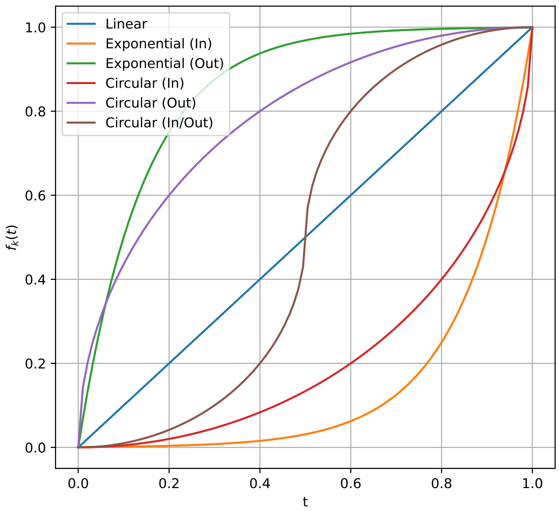


Figure 25: Overview of the used easing functions.

them to interpolate between $\vec{\mathcal{D}}^{start}$ and $\vec{\mathcal{D}}^{end}$ over the time span $[\vec{\mathcal{T}}^{start}, \vec{\mathcal{T}}^{end}]$. In the following, we will take a closer look at six different easing functions.

Ease in Linear:

$$f(t)_1 = t \tag{12}$$

Ease In Exponential:

$$f(t)_2 = \begin{cases} 0 & t = 0 \\ 2^{10(t-1)} & 0 < t \leq 1 \end{cases} \tag{13}$$

Ease Out Exponential:

$$f(t)_3 = \begin{cases} 0 & t = 0 \\ 2^{-10(t-1)} & 0 < t \leq 1 \end{cases} \tag{14}$$

Ease In Circular:

$$f(t)_4 = 1 - \sqrt{1 - t^2} \tag{15}$$

Ease Out Circular:

$$f(t)_5 = \sqrt{1 - (t-1)^2} \tag{16}$$

Ease In/Out Circular:

$$f(t)_6 = \begin{cases} \frac{1-\sqrt{1-t^2}}{2} & 0 \leq t < 0.5 \\ \frac{\sqrt{1-(-2x+2)^2}+1}{2} & 0.5 \leq t \leq 1 \end{cases} \tag{17}$$

4.5.3 Implementation

To be able to use PDC in our kicking process, we first implement an `approachCounter` τ that counts the frames, while the current path routine is `FORWARDKICK`. However, a small movement of the ball, posterior to the approach start, an insufficiently planned approach path or simply vision imparity can lead to the robot not ending up in the desired position, relative to the ball, after the approach. In these cases, a new approach would be initiated, which would reset τ and therefore restart the PDC process, causing an unnecessarily long approach duration. To overcome this, we implement an allowed time gap between two approaches, that does not cause τ to be reset but rather keeps on counting during these gaps. For this, a small number of frames, such as 10, is sufficient. In every frame of the kicking process, the current frame number is therefore remembered, to identify a possible time gap to the next approach.

Next, we implement a function, to control the easing. This function takes in τ, the selected easing style k, as well as $\mathcal{D}_i^{start}, \mathcal{D}_i^{end}$ and $\mathcal{T}_i^{start}, \mathcal{T}_i^{end}$.

If τ is less than $\mathcal{T}_i^{start}$, or greater than $\mathcal{T}_i^{end}$, the function returns the start or end deviation, respectively. Otherwise, it calculates t, by normalizing τ to the interval $[0, 1]$ with respect to $\mathcal{T}_i^{start}$ and $\mathcal{T}_i^{end}$:

$$t = \frac{\tau - \mathcal{T}_i^{start}}{\mathcal{T}_i^{end} - \mathcal{T}_i^{start}} \tag{18}$$

Our newly computed t is then passed to the selected easing function $f_k(t)$ which returns a value in the interval $[0, 1]$. This is then back scaled to $[\vec{\mathcal{D}}_i^{start}, \vec{\mathcal{D}}_i^{end}]$ by:

$$f_k(t) \cdot (\vec{\mathcal{D}}_i^{end} - \vec{\mathcal{D}}_i^{start}) + \vec{\mathcal{D}}_i^{start} \tag{19}$$

PDC can be used either on both the x and y-axis, or just on one of them (most likely the x-axis). To achieve this or even fully disable PDC, we can set the start and end deviations to the same value, for one or both axes. Single axis PDC is a good compromise between precision and speed, since the highest deviation from the target position is usually on the x-axis.

4.5.4 Evaluation

We conduct a small pilot study to evaluate the performance of PDC with different easing functions on the x-axis. Due to the nature of our `StepBuffer` (see Section 2.1.3), the already buffered steps can cause a too long approach, even though the requirements to kick, as defined in Section 2.2.1, are fulfilled. Therefore, we will consider the earliest possible kick moment i.e., the moment, when all requirements are met, as qualifying metric for comparing the different PDC easing functions. We can observe, that functions with a slower growth seem to cause higher precision in approach than faster growing

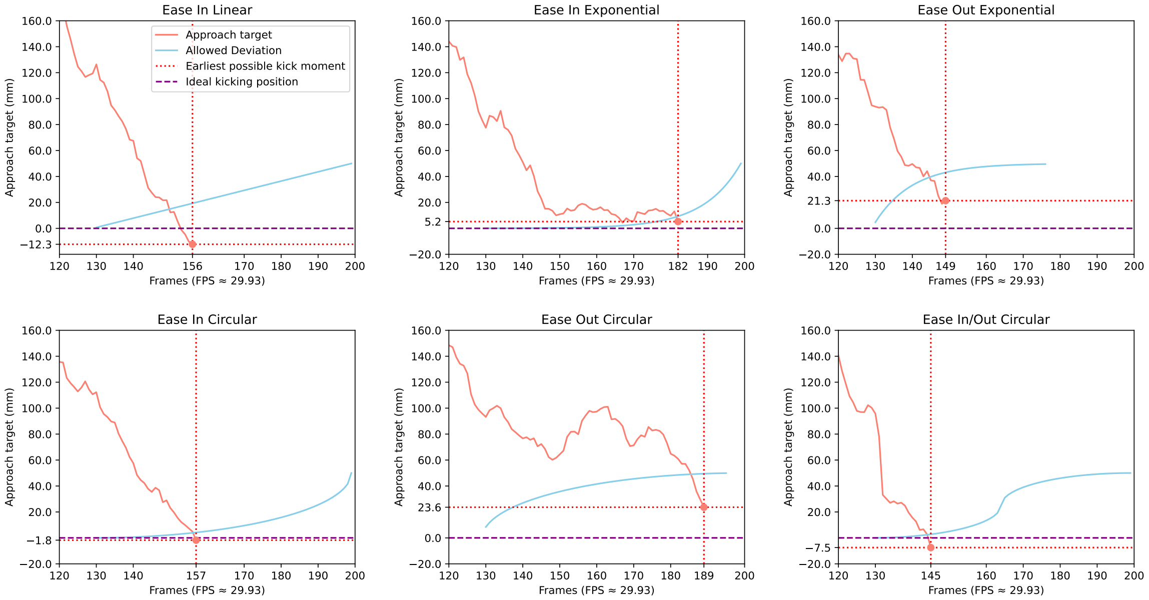


Figure 26: Comparison of PDC with different easing functions on the robot's x-axis.

ones, such as exponential and circular ease out functions. The function, with the highest precision and lowest needed time is ease in/out circular. Further research needs to be conducted to verify the consistency and determine the best working values for the different PDC easing functions.

4.6 Kick Action Extension

Based on the preparatory work in Section 3.8, we are now ready to bring together the full kick action. For the integration in the action selection, we have two options: we can keep the best action selection algorithm as is, or add the mean time it takes to execute the respective kick action as final decider. This means, if more than one possible goal action (see Section 4.1) exist, the robot would choose and execute the fastest one.

Preliminary `KickAction` module, as introduced in Section 3.8:

Mandatory:

- Kick Type
- Kick Velocity
- Kick Height

Optional: (if not specified, the default values will be used)

- Max Kick Length
- Kick Target Offset
- Ideal Ball Distance

Additions to the `KickAction` module, resulting from this section:

Mandatory:

- Mean ball speed
- Standard deviation in speed
- Mean ball angle
- Standard deviation in angle

Optional: (if PDC should be used)

- PDC function
- Start and end deviation
- Start and end time

Optional: (if time is considered)

- Mean kick execution time

5 Discussion and Outlook

The main focus of this thesis lied on studying robotic kicking, both as motion and behavior primitive. To achieve this, we have given our selves two main objectives: First, evaluating the different motion variables and their influence on the resulting kick, optimizing the kick distance and preparing for hot-swappable kick actions. Second, to explore possibilities to increase the accuracy of the predictive action model by evaluating the robot's ability to autonomously verify and adjust it, assessing the inclusion of the spacial context and investigating ways to decide, when to end the approach and execute a kick.

With the experiment-driven optimizations, described in Section 3 and introduced techniques, such as the kick target offset, we were able to increase the mean absolute kicking distance, by 37.98 %, or 98.76 cm, compared to the baseline, i.e., the configuration used prior to this work, while keeping the standard deviation nearly the same ($\sigma_{old} = 36.61\,\text{cm}$, $\sigma_{new} = 37.33\,\text{cm}$). A comparison of the baseline and the best performing configuration, with their respective means can be seen in Figure 16.

We also found that, while a kick height of 35 mm produced the highest absolute kick distance, balls kicked at a hight of just 15 mm, had a short travel distance, with a mean of just under two meters and a remarkably low standard deviation ($\sigma = 24.35\,\text{cm}$), making this a good choice for short and precise kicks.

When evaluating our kicking performance against the results, featured in Wege's bachelor thesis [23], we find that their highest performing kick configuration, yielded a similar mean distance to our baseline configuration, but had a higher standard deviation ($\sigma \approx 46\,\text{cm}$). Comparing our short and precise configuration with the most similar one of theirs, in terms of kick distance, we can see a clear difference in the standard deviation. While our kick comes with a standard deviation of $\sigma = 24.35\,\text{cm}$, theirs is noticeable higher with $\sigma \approx 42\,\text{cm}$.

In order to find possible ways to increase the accuracy of the predictive action model, we started by evaluating the robot's ability to autonomously verify and adjust it. For this, we used the already existing ball model to implement the ability of recognizing a completed kick, i.e., noticing that the ball has stopped rolling. This was done by analyzing the noise in the perceived ball velocity and defining a threshold. We then compared the robot's measurements to a reliable external measuring system, assessing the error in the robot's perception. In our experiments, we found the ball model to be accurate for distances up to ~2.7 m, but tended to overestimate the distance of balls further away.

Evaluating the ball's position in the robot's coordinate system before a kick, we found the x distance of the ball to have a high impact on the kick. The range y distances resulting from the robot's approach, showed to be too small to have a substantial impact. Therefore, we concluded, that including the ball position into the predictive action model, will most likely be beneficial for the model's accuracy.

To decide, when to end the approach and execute a kick, we proposed a probabilistic and a time-based method. For the probabilistic method, we developed a new model to predict a kick's distance and angle from the robot's current approach position. We compared three supervised learning techniques, finding linear regression with a feature set, consisting of the ball position and the dynamically computed kick time, to work best for predicting both distance and angle. With our data set, we were able to predict the kick distance with a mean absolute error of 46.46 cm. However, when assessing the single predictions, we noticed errors beyond acceptable limits, with 21.88 % of the prediction errors being greater than the MAE.

As time-based method, we introduce a novel technique, that uses easing functions to progressively increase the initially low, allowed deviation from the approach target, with the goal to realize fast and precise approaches. For this, we compared six different easing functions in a pilot study to asses their usability for this cause.

5.1 Future Work

The first step for improving the prediction based kick decision, would be increasing the sample size and experimenting with deeper neural networks. Doing so, enables for evaluation whether more accurate predictions are possible, or if the uncertainty, caused by both the robot and the environment is too high. If a precise prediction of kick distance and direction is not possible, it might be a good compromise to predict, if a kick will surpass a chosen distance n. In a small pilot study, this classification approach was attempted on our dataset, but did not yield any reliable results. If however, the increase of the sample size leads to better predictions, it would be very interesting to explore whether this can be reversed, to accurately predict the robots approach target based on a desired end position of the ball. Achieving this, would enable the implementation and utilization of a backwards simulation, allowing for very precise kicks on the field. This was also attempted in a small pilot study, but again did not produce good results. Another, very interesting, future task would be to refine the algorithm, described by Equation (3), by using the kick prediction to choose the next step target, based on its potential.

Also, the impact of the robot's initial momentum, gained through the approach, on the resulting kick needs to studied. Possible test scenarios for this could include, the robot kicking the ball out of the stand, after approaching from different distances or directions, as well as including short pauses or a turning motion, right before the kick. While experimenting on the robot's kick, the joint temperatures, especially those of the knees were found to impact the kick performance, when getting too high. This needs to be studied in depth in the future. Both, the joint temperatures and the robot's acceleration before kicking, could potentially be included as features for the kick prediction. However, this would likely require, collecting a relatively large sample set.

References

[1] BerlinUnited/NaoTHSoccer repository. `https://github.com/BerlinUnited/NaoTH/tree/develop/NaoTHSoccer/`. Accessed: 20.07.2025.

[2] PresToe: A Humanoid Robot with an Actuated Toe : Robotics @ UMass Amherst. `https://www.umass.edu/robotics/projects/prestoe-humanoid-robot-actuated-toe/`. Accessed: 09.09.2025.

[3] RoboCup Standard Platform League (NAO) Rule Book. `https://spl.robocup.org/wp-content/uploads/SPL-Rules-2025.pdf/`, 2025. Accessed: 28.07.2025.

[4] A. Böckmann and T. Laue. Kick Motions for the NAO Robot Using Dynamic Movement Primitives. In *Lecture Notes in Computer Science*, pages 33–44. Springer International Publishing, 2017.

[5] J. Y. Choi, B. R. So, B.-J. Yi, W. Kim, and I. H. Suh. Impact Based Trajectory Planning of a Soccer Ball in a Kicking Robot. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 2834–2840. IEEE, 2005.

[6] T. Haarnoja, B. Moran, G. Lever, S. H. Huang, D. Tirumala, J. Humplik, M. Wulfmeier, S. Tunyasuvunakool, N. Y. Siegel, R. Hafner, M. Bloesch, K. Hartikainen, A. Byravan, L. Hasenclever, Y. Tassa, F. Sadeghi, N. Batchelor, F. Casarini, S. Saliceti, C. Game, N. Sreendra, K. Patel, M. Gwira, A. Huber, N. Hurley, F. Nori, R. Hadsell, and N. Heess. Learning agile soccer skills for a bipedal robot with deep reinforcement learning. *Science Robotics*, 9(89), 2024.

[7] J. Hartung and L. Craatz. Advanced Kicking for NAO Robots. Humboldt-Universität zu Berlin, 2024. Study Project.

[8] J. Koenemann and M. Bennewitz. Whole-body imitation of human motions with a nao humanoid. In *Proceedings of the Seventh Annual ACM/IEEE International Conference on Human-Robot Interaction*, HRI '12, pages 425–426. Association for Computing Machinery, 2012.

[9] D. L. Leottau, J. Ruiz-Del-Solar, P. Macalpine, and P. 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.

[10] S. Liu, G. Lever, Z. Wang, J. Merel, S. M. A. Eslami, D. Hennes, W. M. Czarnecki, Y. Tassa, S. Omidshafiei, A. Abdolmaleki, N. Y. Siegel, L. Hasenclever, L. Marris, S. Tunyasuvunakool, H. F. Song, M. Wulfmeier, P. Muller, T. Haarnoja, B. Tracey, K. Tuyls, T. Graepel, and N. Heess. From motor control to team play in simulated humanoid football. *Science Robotics*, 7(69), 2022.

[11] D. Marew, N. Perera, S. Yu, S. Roelker, and D. Kim. A Biomechanics-Inspired Approach to Soccer Kicking for Humanoid Robots. In *2024 IEEE-RAS 23rd International Conference on Humanoid Robots (Humanoids)*, pages 722–729. IEEE, 2024.

[12] M. Matsumoto and T. Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30, 1998.

[13] J. S. Matthis, J. L. Yates, and M. M. Hayhoe. Gaze and the Control of Foot Placement When Walking in Natural Terrain. *Current Biology*, 28(8):1224–1233.e5, 2018.

[14] H. Mellmann. Ein anderes Modell der Welt - Alternative Methoden zur Lokalisierung Mobiler Roboter. Humboldt-Universität zu Berlin, 2010. Diploma thesis.

[15] H. Mellmann. Anticipation – an Approach to Complex Decision-Making under Uncertainty in Artificial Physical Agents. Humboldt-Universität zu Berlin, 2025. Dissertation.

[16] H. Mellmann and S. Schlotter. Advances on Simulation Based Selection of Actions for a Humanoid Soccer-Robot. In *Proceedings of the 12th Workshop on Humanoid Soccer Robots, 17th IEEE-RAS International Conference on Humanoid Robots (Humanoids), Madrid, Spain.* IEEE, 2017.

[17] H. Mellmann, S. Schlotter, S. Kaden, P. Strobel, T. Krause, E. Couque-Castelnovo, C.-N. Ritter, and R. Martin. Berlin United - NaoTH Team Report 2019.

[18] H. Mellmann and Y. Xu. Adaptive motion control with visual feedback for a humanoid robot. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3169–3174, 2010.

[19] N. Mulvaney. An easing equation browser. `https://nicmulvaney.com/easing/`. Accessed: 01.08.2025.

[20] J. Müller, T. Laue, and T. Röfer. Kicking a Ball – Modeling Complex Dynamic Motions for Humanoid Robots. In J. Ruiz-del Solar, E. Chown, and P. G. Plöger, editors, *RoboCup 2010: Robot Soccer World Cup XIV*, volume 6556, pages 109–120. Springer Berlin Heidelberg, 2011.

[21] S. Schlotter. Selection of Actions based on Forward Simulations. Humboldt-Universität zu Berlin, 2017. Bachelor thesis.

[22] A. Ude, C. Atkeson, and M. Riley. Planning of joint trajectories for humanoid robots using B-spline wavelets. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 3, pages 2223–2228. IEEE, 2000.

[23] F. Wege. Development and Implementation of a Dynamic Kick for the NAO Robotic System. Technische Universität Hamburg, 2017. Bachelor thesis.

[24] F. Wenk and T. Röfer. Online Generated Kick Motions for the NAO Balanced Using Inverse Dynamics. In *Lecture Notes in Computer Science*, pages 25–36. Springer Berlin Heidelberg, 2014.

[25] Y. Xu and H. Mellmann. Adaptive motion control: Dynamic kick for a humanoid robot. In R. Dillmann, J. Beyerer, U. D. Hanebeck, and T. Schultz, editors, *KI 2010: Advances in Artificial Intelligence*, pages 392–399. Springer Berlin Heidelberg, 2010.

[26] K. Yamane, S. O. Anderson, and J. K. Hodgins. Controlling humanoid robots with human motion data: Experimental validation. In *2010 10th IEEE-RAS International Conference on Humanoid Robots*, pages 504–510. IEEE, 2010.

[27] Zhaoqin Peng, Qiang Huang, Xiaojun Zhao, Tao Xiao, and Kejie Li. Online Trajectory Generation Based on Off-line Trajectory for Biped Humanoid. In *2004 IEEE International Conference on Robotics and Biomimetics*, pages 752–756. IEEE, 2004.

Appendix

The NAO Robot

Dimensions	
Hight × Depth × Width	573mm × 311mm × 275mm
Weight	5.4 kg
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)
Actuators	
Degrees of Freedom (Joints)	25
LEDs	
Sound	2 Speaker (Stereo), located on the sides of the head.
Sensors	
Joint Position Sensors	Hall effect sensors in each joint (25)
Cameras	2, 640 × 480, vertical arrangement with minimal overlap
Microphones	4 microphones in trapezoid arrangement
Ultrasound	
Inertia Sensors	3D Accelerometer, 3D Gyrometer