Climbing Stairs with DARwIn-OP humanoid robots using kinematic tasks

Daniel Ahlers, Pamina M. Berg

University of Hamburg, Department of Informatics
http://www.inf.uni-hamburg.de/en.html

Abstract. The walking of biped humanoid robots has been an active research area, whether it is for educational purpose, for supporting armed forces in difficult terrain or for developing service robots that will act, communicate and walk in the most human manner possible, as our world is created for two-legged locomotion. Another aspect is the development of exoskeletons for the lower limbs, which can benefit from the gait designs for humanoid robots. Especially in the context of developing service robots, the climbing of staircases is an essential part of the ability to move around a real-life environment, such as a two-story apartment or house. This report focuses on the implementation and design of a simple stair climbing gait for the DARwIn-OP, using animations and kinematic tasks.

Key words: Stair Climbing, Humanoid Robot

1 Introduction

Although the human bipedal locomotion is the most elaborate kind of gait, a large number of computer scientists, phisicists, engineers and other analysts of a variety of scientific fields have been experimenting and developing humanoid robots to establish an effigy of a human being, capable to move autonomously on two legs. Despite the large number of robots with four legs or more, or even those on wheels, that can already provide a reliable help as service robots, there are numberous aspects that speak in favor of the development of humanoid robots. First of all, robots have been generated to assist human beings in their everyday life. Our environment, e.g. apartments, cinemas, streets and stores, has been built for beings with an upright posture. Especially people with walking impediments know the kind of challenge that comes with not being able to move around on two legs. One of the most common barriers are staircases. Therefore, we need to build robots that are customized for the environment in which they have to operate. The advancements on this topic are an important part of basic scientific research which can be used for artificial legs that may adapt to the people's walking impediments, thus contributing to their well-being.

There have been several approaches to make a biped humanoid climb stairs. Akhtaruzzaman and Shafie proposed a model of alternating two phases, the

Single Support Phase (SSP), where we have a stance leg and a swing leg, and the Double Support Phase (DSP). The ascending of the stairs has then been provided by running through a series of 15 poses for one single step [1]. Choi et al. used a ZMP motion planning algorithm and described three elementary phases of stepping up one stair tread – the ascending forward phase, the forward motion phase and the upward motion phase [2]. Making use of the advances in stereo vision of robots, Gutmann et al. provided an automatism for stair recognition, where the control system integrated in the humanoid robot decides on the next move to climb a staircase [3]. Control algorithms have been the basis of the approaches made by Kim et al. [5] and Sheng et al. [9] which focus on the gait optimization and also work on the balance and stability, as well as minimizing the energy required to achieve these goals. One of the latest advancements comes from *Boston Dynamics*, who presented a modified version of its humanoid robot *PETMAN* that established a stable walking up stairs.

Regarding a variety of possibilities to achieve a stable walking gait, the biped humanoid is often considered to be an open kinematic chain [1]. In our case this means, that we have rigid bodyparts, that are connected using motors as joints. For example, the leg of a robot consists of the rigid upper leg, the rigid lower leg and the foot, connected by motors which we will refer to as the knee and the ankle. Every joint in the robot has at least one of the three angles of rotation. These are called *roll, pitch* and *yaw*, and as it can be seen in Fig. 1, each one has a specific movement in the three-dimensional space. For the DARwIn-OP of the Hamburg BitBots, we will have a special orientation on the axes x, y and z, as the x-axis will always direct to the next joint in the kinematic chain. E.g. the x-axis of the hip joint directs vertically down to the knee, while the x-axis of the ankle joint directs horizontally toward the front endpoint of the foot.

As we will refer to the concept of [1], now a short introduction into their work. Referring to the above, Akhtaruzzaman and Shafie assembled a gait for ascending stairs using a 'combination of various postures and poses' [1, p.9]. These are the two *Action Poses* at the beginning and the end, where the robot will stand upright, supporting its weigth with both feet, followed by the *Tilt Pose* and the *DS-SS Pose*, which means, that the robot shifts its weigth support from two legs (Double Support) to a single one (Single Support), thus we get a stance and a swing leg. After this, the robot proceeds to the *Foot Lifting Pose*, the *Foot Forward Pose* and the *Foot Adjust Pose* to lift the swing leg upwards an onto the tread, hence, concluding the first swing phase. In the following *SS-DS Pose*, *SS-DS Complete Pose* and *Tilt Pose* the foot of the swing leg will be placed onto the tread, so that the robot will now stand on both of its feet again. Now, 'before the beginning of the second swing phase with the rear foot', in the *CoM Lifting Pose*, 'the anthropoid moves its Center of Mass (CoM) gradually upwards to fix its stability by front foot support' [1, p.9-10]. In the second swing phase, the robot runs through the *DS-SS Pose*, the *Foot Lifting Pose*, the *Foot Forward*

Pose and the *SS-DS Pose* again, terminating the process with the *Action Pose* [1]. (PMB)

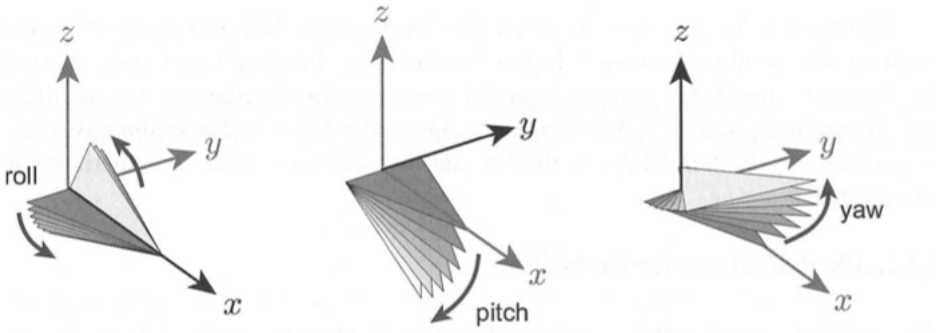

Fig. 1. The three angles of rotation [4, p.21]

2 Implementation

2.1 Preliminaries

The first step we needed to make was to build a model. We build the staircase made of wood on a scale of 1 : 4 using the DIN18065 for staircases and it therefore has a 45mm height of treads and a depth that is slightly bigger than the foot of the DARwIn-OP. So the first tread has a width of 124mm as we tested with the first one and did not want the robot to fall because of a lack of space an the tread, followed by three treads with a width of 104mm and a nosing of 8mm each.

The stair climbing module has been developed for and tested with the DARwIn-OP of the Hamburg BitBots. They have some modifications compared to the original version from Robotis, as the head has been replaced by one that will not be as much affected by an impact on the ground as the former construction, and some alterations considering the armor of the legs, as well as the feet. Hence, the Center of Mass is in a different position. This has an effect on the stabilization of the robot during bipedal walking and – in our case – climbing up stairs, but fortunately our implementation does not rely on calculations based on the CoM. (DA)

Fig. 2. Staircase model

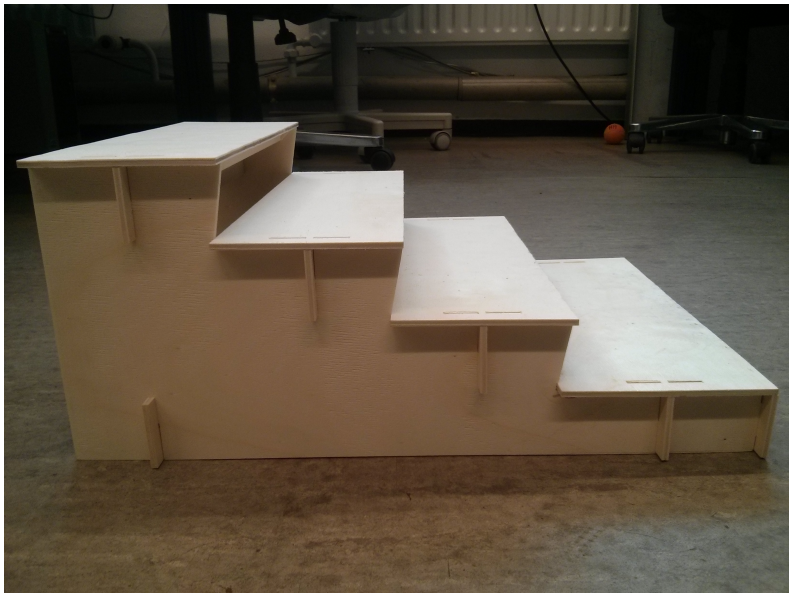

Fig. 3. Staircase model side view

2.2 General Aspects Of The Robotic Bipedal Walking

To understand the difficulty of stabilizing a humanoid robot, it is useful to get an overview of the different factors that need to be considered when planning motions. One of the essential aspects is the *stability region*, which is the convex envelope of the feet [4, p.54]. For a robot standing on one leg, the stability region would be the outer edges of the foot.
The next basic term is the so called *Zero Moment Point (ZMP)*. Sardain and

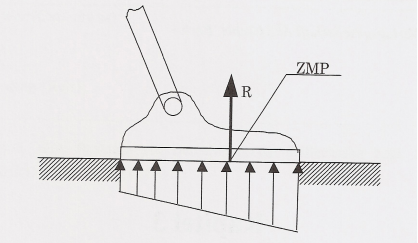

Fig. 4. Visualization of the Zero Moment Point, [4, p.54]

Bessonnet define the ZMP as 'the point on the ground, where the tipping moment acting on the biped, due to gravity and inertia forces, equals zero' [8, p.2]. In other words, the ZMP is the point of contact on the ground in the stability region, on which the robot bases its posture on. The ZMP is also an indicator for the stability of the robot: if the ZMP is in the center of the stability region, the stability of the robot during its walking is at its maximum. But there is also a limit of using solely the ZMP for designing a walking gait for a humanoid robot, as this concept can not be adapted to walk on uneven surfaces [4, p.81].
The *Center of Pressure (CoP)* is the point where the single force, which is equivalent to the 'field of pressure forces (normal to the sole)' [8, p.1] exerts when the resultant moment is zero. Ideally, the ZMP and the CoP are in the same spot within the stability region.

Human locomotion has been translated into bipedal walking of robots in different ways. There are the approaches of the *Passive-dynamic Walking*, used by the *Cornell Biped*, and the *Static Walking*, where the perpendicular point of the CoM on the ground never leaves the stability region, thus leading to enormous feet of the robot. Nowadays, the most common walking concept is the *Dynamic Walking*. Using the pattern of a three-dimensional inverted pendulum, an imitation of the human locomotion can be implemented. For this, a walking pattern can be described, as well as the floor trajectory of the feet (Fig. 5) [4].

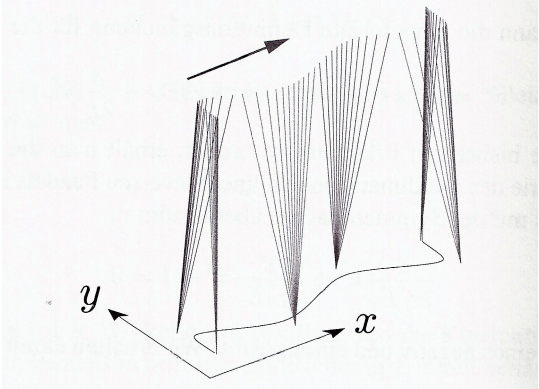

Fig. 5. Pattern for a straight forward walk with three steps

A relatively new approach in the dynmamic locomotion has been made using Linear Quadratic Optimal Control. Based on the prior experiences using optimization-based techniques, made by e.g. KAJITA, a new development has been made by KUINDERSMA, where a fully actuated robot body system is assumed to be a set of linear dynamics, thus, an optimal walking gait can be obtained by using linear optimal control [6]. (PMB)

2.3 Motion Design

The general idea consists of running through several poses and stabilizing the robot at certain points using animations.

Following the concept of [1] as presented in the preceded section, we started by planning the motion process of one step up stairs. In our project, it consists of five different phases. After the robot comes to a stable pose on both feet, the next phase would be that the robot shifts the support of its weight to a single leg, e.g. the right one, while keeping up its balance. This first phase ends with the robot standing on the right (left) leg with the other leg slightly above the ground as shown in Fig.6.

In the second phase the robot moves its swing leg onto the tread. For this, the foot of the swing leg will be raised, then moved forward to hover over the tread and subsequently set down on the tread (Fig. 7). The robot will then go on to the next phase where the support will be shifted from the foot on the ground to the foot on the tread (Fig. 8). The difficulty of this phase lies in the height difference, as the robot has to overcome it while shifting its weight and Centre of Mass onto the tread without losing its balance. Before actually standing upright on the stairs, the robot needs to proceed through the fourth phase, thus lifting

Fig. 6. Phase 1

the former stance leg from the ground and onto the tread, similar to phase two. Now the robot can bring itself to an upright position in phase five. Because of the limited power of the motors in the knees, the robot can only stand up from a kneeling position (e.g. in phase four (Fig. 9)) using both of its legs.

Note, that since balancing the robot, especially during a SSP, is a challenging task without having a fully developed balance system, we used not only the foot, leg and hip motors, but created a full body movement. (DA)

2.4 Implementation and Methods

As there is a kinematic for the DARwIn-OP already provided by the framework of the Hamburg BitBots, our first attempt to transfer our ideas into some sort of code was to use the given kinematic.
Kinematic in general is the field of research describing the relationship between the angles and the alignment of joints. It represents the basis of autonomous robotic locomotion [4, p.15]. There are different approaches for using kinematic to achieve a stable walking gait for humanoid robots. One is the calculation of

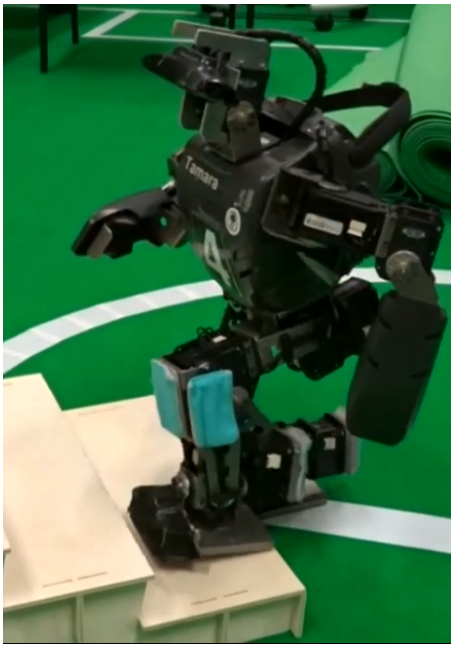

Fig. 7. Phase 2

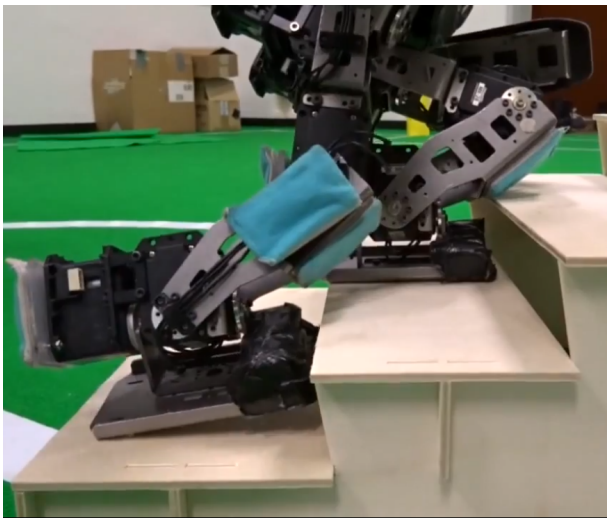

Fig. 8. Phase 3

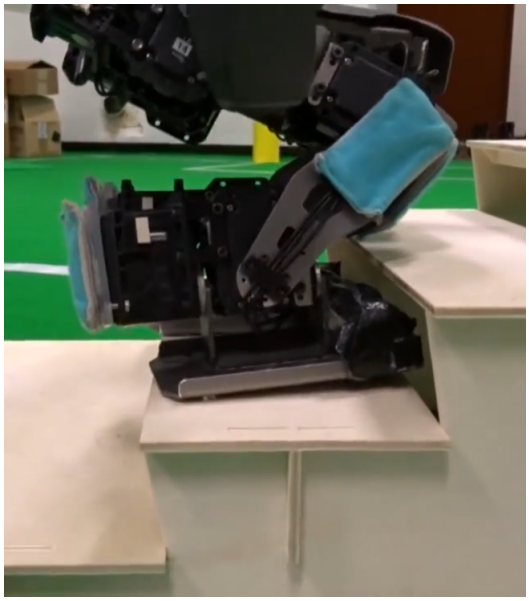

Fig. 9. Phase 4

joint positions given the joint angles, called forward kinematics, the other one is the inverse kinematic, where the joint angles will be reckoned from the position and the pose of the leg and torso [4]. Kajita describes the inverse kinematic as a necessary element for positioning the leg of a humanoid robot in the right height for stepping up a tread [4, p.40].

Due to a lack of functionality of one essential method (`keep_robot_stable`) in the kinematic of the Hamburg BitBots framework, we began writing our own animations for the stabilization of the robot. These animations are, in general, fixed poses which the robot is told to take in a certain amount of time, using its motors. In each pose, every joint of the robot has a specific angle. To make the robot move, we can change these angles within a given amount of time using animations. The code for one of our animations is shown in Fig. 10. For keeping our stair climbing robot stable, we developed four different animations, two for balancing on a single leg (`balance_left` and `balance_right`) and two for changing the support leg (`balance_left_to_right` and `balance_right_to_left`). These animations are used in Phase 1, where the robot will be told whether to start with the right or left leg. If we want the robot to climb the tread with the left leg first, it will then use the animation `balance_right` to shift its weight to the right leg. Since there is a similarity between the motion at the beginning of the stair-climb and the motion before performing a kick (e.g. on the soccer field with the

ball in front of the robot), we used parts of the kick module given in the framework of the Hamburg BitBots for the stabilization in Phase 1. The robot then

```
{
    "name": "balance_right",
    "default_interpolator": "LinearInterpolator",
    "keyframes": [
        {
            "duration": 1,
            "goals": {
                "LAnklePitch": -28,
                "LAnkleRoll": -13,
                "LElbow": 0.0,
                "LHipPitch": 34,
                "LHipRoll": 0.0,
                "LHipYaw": 0.0,
                "LKnee": -52,
                "LShoulderPitch": 61.7,
                "LShoulderRoll": 0.0,
                "RAnklePitch": 29.5,
                "RAnkleRoll": -13,
                "RElbow": 49.4,
                "RHipPitch": -38,
                "RHipRoll": 2.8,
                "RHipYaw": 0.0,
                "RKnee": 52,
                "RShoulderPitch": -15.6,
                "RShoulderRoll": 0.0
            },
            "pause": 0.5
        }
    ]
}
```

Fig. 10. Code for the animation `balance_right`

lifts its leg onto the tread, proceeding to Phase 2. We implemented this by using kinematic tasks, which we defined as `foot_up`, `foot_forward` and `foot_down`. In each of these tasks, we needed to define the angles for an array of joint positions through the motor ID of each joint. To achieve the optimal joint angles, we experimented with several different options. Although we found good working values, we made some additional pose updates in the tasks `foot_forward` and `foot_down` to readjust the balance in the ankle pitches, ankle rolls and the knees. After the robot set down its foot on the tread, it will proceed to Phase 3, thus shifting its weight from the right leg on the ground to the left leg on the tread. For this, we used the animations `balance_left_to_right` and `balance_right_to_left`, depending on which foot is positioned on the tread. So for the described scenario, we would use the animation `balance_right_to_left`, where we defined the goals for the joint angles of both ankle pitches, ankle rolls,

elbows, hip pitches, hip rolls, hip yaws, shoulder pitches and shoulder rolls, just like we did for the animation `balance_right` as it can be seen in (Fig. 10).
In the subsequent Phase 4, where the right leg will be lifted and set down on the tread, we used our kinematic tasks `foot_up2`, as well as an additional pose update in the ankle goal, `foot_forward2` with an adjustment in both hip pitches and `foot_down2`, correcting the angle values of again both hip pitches as well as the ankle pitches, due to stability reasons. We obtained these values by trial and error like we did for the animations in Phase 2.
For the last phase, Phase 5, we modified the ankle roll joint values using kinematic tasks. At last, the robot brings itself into an upright pose using the animation `walkready` provided by the framework of the Hamburg BitBots, thus concluding the step up the first tread. The whole gait can be illustrated by a diagram, where the robot would start at the position *Walkready* (Fig. 11). (DA)

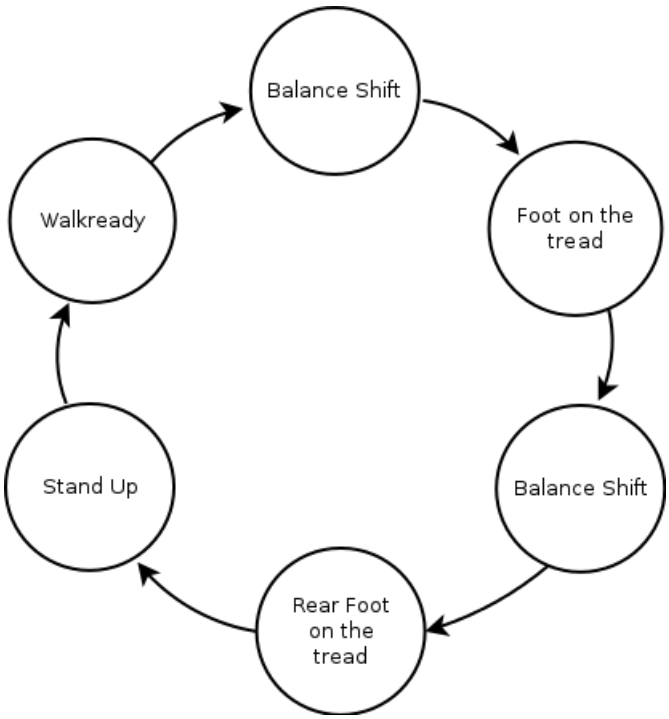

Fig. 11. The different phases of the motion sequence

3 Results

We established a stable up-stair motion process for the DARwIn-OP, using animations and kinematic tasks, thus enabling a robot to climb a tread of 45mm height. The code given in the appendix shows that we provided an implementation where the user can dynamically change the height of the treads, hence making this code adaptable to other staircases. Note that we only tested the code on our model staircase with the measurements as described in Section 2.1.
During the development process for making the robot climb up a staircase, we discovered some limits in the hardware as well as in the software.
Unfortunately, the motors in the knees do not have enough power to raise the robot using only one knee. Hence, we had to bring the robot back to a double support stance after Phase 4 and the robot will get into a stable upright position on both feet before proceeding to take the next tread.
Another point was that we used the simulation software of the Hamburg BitBots for testing the angles that had to be defined manually. The disadvantage of this was that the DARwIn-OP sometimes had a slight difference in the values, thus making it hard to decide on an optimal value for the angles. With regard to this problem, the robot sometimes rotated on one foot during the phases three to five, so that it was then impossible for the robot to subsequently proceed to climb the next tread. Making use of the vision to let the robot recognize the tread and its position in relation to it and positioning itself correctly before proceeding to step up the next tread as a result could be a further enhancement to the software.
The simulation software also does not provide a model staircase, thus we could only simulate e.g. the motion of the swing leg itself without knowing if it would fit our staircase properly.
While testing the code on the robot we also noticed a lack of stability of our staircase model. Since the upper treads are very slim, they would bend down some millimeters, such that as a consequence, the stability and balance of the robot will be influenced once the robot reaches the second tread. Because our animations are based on fixed joint angle values, the robot is not capable of adapting to this circumstance.
Unfortunately, we were not able to develop a concept to calculate the required motion processes dynamically, which would give a solution to the preceded problem.
Concluding the results, the developed motion process allows the robot to climb a staircase without falling off, but can be optimized with regard to the speed.
The project results do not provide a direct enhancement for the soccer playing DARwIn-OP, but helps to understand the process of movement for humanoid robots, as it uses very basic elements of the motion (e.g. changing values of the joint angles to move the foot or leg). This collaborates with the original idea behind the RoboCup Rescue, which is one of the major competition domains in the RoboCup tournament in general and has been created as a consequence of an enormous earthquake in Kobe City in 1995 [7]. With this division that challenges teams all over the world to develop rescue robots that can act and

move autonomously, the advancements made in the basic scientific research of human walking patterns and their adaption due to this challenge did not only enable the research on robotics to benefit from this but also medical science, e.g. in the development of exoskeletons. (PMB)

4 Conclusion

A simple walking up stairs gait for the DARwIn-OP has been presented, using kinematic tasks and animations. The stabilization of the robot is made by manually correcting the joint angles after several poses.
As there is a concept by the Hamburg BitBots for building a new humanoid robot using a 3D printer for the rigid bodyparts, as well as more powerful motors as joints, the preceded gait for ascending stairs can be enhanced with regard to the motion speed. For this, a shorter period of time can be chosen to perform the angle goals in the animations.
Another possibilty for further enhancement is to use the vision of the robot to recognize not only the staircase, but the height of the treads, following the example set by [3]. It could then be considered to make use of the concept provided by [6] and therefore developing a feedback control loop, in which the sensor data of the robot will be evaluated and used in the process of calculating the next step. As this is not only a time-consuming process, but has also great computational costs, the next generation robot of the Hamburg BitBots will probably not be able to achieve this kind of dynamic walking gait.
For future academic projects considering the climbing up stairs, there can be made some changes and improvements in the kinematic of the Hamburg BitBots framework, e.g. in the method `keep_robot_stable`. Using this method, we could have simulated the human locomotion more efficiently because the robot would have then used the inverse kinematic to perform a task and keep itself stable during the process. This could lead not only to smoother movements of the robot, but to a walking gait that is much more gentle to the joints of the robot, and as a result the motors could be prevented from excessive wear. (PMB)

References

1. Akhtaruzzaman, Md., Shafie, A.: Novel Gait for an anthropoid and Its Joint Demeanors while Stepping Up and Down Stairs. In: Journal of Mechanical Engineering and Automation, 1(1):8–16 (2011)
2. Choi, J., Choi, Y., Yi, B.: A Up-Stair Motion Planning Algorithm for a Biped Robot. In: 5th International Conference on Ubiquitous Robots and Ambient Intelligence, pp. 681–686. (2008)
3. Gutmann, J.-S., Fukuchi, M., Fujita, M.: Stair Climbing for Humanoid Robots Using Stereo Vision. In: Proceedings of the 2004 IEEE/RSJ International Vonference on Intelligent Robots and Systems, pp. 1407–1413. Sendai, Japan (2004)

4. Kajita, S., Hirukawa, H., Yokoi, K., Harada, K.: Humanoide Roboter - Theorie und Technik des Künstlichen Menschen. Ed. Shuji Kajita, Akademische Verlagsgesellschaft, Berlin (2007)
5. Kim, J.-Y., Park, I.-W., Oh, J.-H.: Realization of Dynamic Stair Climbing for Biped Humanoid Robot Using Force/Torque Sensors. In: Journal of Intelligent and Robotic Systems, Vol. 56, Iss. 4, pp. 389–423. Springer Netherlands (2009)
6. Kuindersma, S., Permenter, F., Tedrake, R.: An Efficiently Solvable Quadratic Program for Stabilizing Dynamic Locomotion. In: Proceedings of the International Conference on Robotics and Automation (ICRA). Hong Kong, China (2014)
7. RoboCup Rescue Wiki, http://www.robocuprescue.org/wiki/index.php?title=Main_Page
8. Sardain, P., Bessonnet, G.: Forces Acting on a Biped Robot. Center of Pressure - Zero Moment Point. In: IEEE Trans. Syst. Man Cybern. A., Syst. Humans, Vol. 34, No. 5, September (2004)
9. Sheng, B. et al.: Multi-Objective Optimiation for a Humanoid Robot Climbing Stairs based on a Genetiv Algorithm. In: Proceedings of the 2009 IEEE International Conference on Information and Automation, pp. 66–71. Zhuhai/Macau, China (2009)

Appendix

```
#-*- coding:utf-8 -*-

"""
Treppensteigen

Entwickelt im Rahmen des Robocup Projekts im WiSe 14/15

Es soll dem DARwIn-OP ermoeglichen kleine Stufen zu erklimmen.
Die Stufenhoehe kann variiert werden,
es wurde aber nur mit einer Stufenhoehe von 45mm Stufen getestet.
Die Stufentiefe sollte in etwa die Tiefe des Fusses des DARwIn-OP haben.

Die Stabilisierung erfolgt ueber Animationen,
Die Bewegung der Beine ist mit Kinematischen Tasks umgesetzt.

"""

from bitbots.ipc.ipc import *
from bitbots.robot.pypose import *
from bitbots.util.kinematicutil import *
from bitbots.robot.kinematics import *
from bitbots.util.animation import *

import numpy as np
import time

class StairClimb:
    #Initialisieren des Roboters
    def __init__(self):
        self.ipc = SharedMemoryIPC()
        self.pose = PyPose()
        self.robot = Robot()
        self.task = KinematicTask(self.robot)

    #Funktion zum Warten auf eine laufende Animation plus x Sekunden
    def wait_for_end(self, sleeptime=2):
        time.sleep(0.1)
        #warte auf Animation
        while (not self.ipc.controlable) or
        (self.ipc.get_state() == STATE_ANIMATION_RUNNING:)
            time.sleep(0.05)
        #warte weitere x Sekunden
        time.sleep(sleeptime)

    #schreibt die Pose auf den IPC und
    # aktualisiert im Anschluss Pose und robot
```

```
48      def update_pose(self, sleeptime=2):
49          self.ipc.update(self.pose)
50          self.wait_for_end(sleeptime)

52          self.pose=self.ipc.get_pose()
53          self.robot.update(self.pose)

55      #spielt eine Animation ab und aktualisiert
56      #im Anschluss Pose und robot
57      def animation_play(self, animation, sleeptime=0.5):
58          play_animation(animation, self.ipc)
59          self.wait_for_end(sleeptime)
60          self.pose=self.ipc.get_pose()
61          self.robot.update(self.pose)

63      #bringt den Roboter in Ausgangsstellung
64      def walk_init(self):
65          self.wait_for_end(0)
66          self.animation_play("walkready",0)

68      #fuehrt einen Kinematischen Task aus
69      def perform_task(self, array, i, legnr, speed=1):
70          if legnr==1:
71              #erstes Bein auf der Treppe
72              self.task.perform(0, 34+i, [(1, 0, 0), array],
73                          (1e-2, 1), (0, 3), 100, [15+i], [7+i,17+i])
74          else:
75              #oder zweites
76              self.task.perform(0, 35-i, [(1, 0, 0), array],
77                          (1e-2, 1), (0, 3), 100, [16-i], [8-i,18-i])
78          #Winkel auf Roboter schreiben
79          self.robot.set_angles_to_pose(self.pose,-1,speed)

81      #steigt eine Stufe hoch
82      def walk_step(self, height, i):
83          i = i%2
84          if i==1:
85              #linkes Bein zuerst
86              balance="balance_right"
87              foot_up=np.array((-20, 47, -265+height))
88              foot_forward=np.array((120, 47, -265+height))
89              foot_down=np.array((120, 47, -275+height))
90              balance_shift="balance_right_to_left"
91              foot_up2=np.array((-50, -47, -270+height))
92              foot_forward2=np.array((35, -47, -270+height))
93              foot_down2=np.array((60, -47, -275+height))
94          else:
95              #rechtes Bein zuerst
96              balance="balance_left"
97              foot_up=np.array((-20, -47, -285+height))
```

```
98              foot_forward=np.array((120, -47, -285+height))
99              foot_down=np.array((120, -47, -315+height))
100             balance_shift="balance_left_to_right"
101             foot_up2=np.array((-50, 47, -270+height))
102             foot_forward2=np.array((35, 47, -270+height))
103             foot_down2=np.array((60, 47, -275+height))

105         #balanciert auf einem Bein
106         self.animation_play(balance)

108         #anderen Fuss heben
109         self.perform_task(foot_up,i,1)
110         self.update_pose(1)

112         #Fuss vor bewegen
113         self.perform_task(foot_forward,i,1)
114         #Gleichgewichtskorrekturen fuer rechts oder links
115         if i==1:
116             self.pose.l_ankle_pitch.goal = -20
117             self.pose.r_ankle_roll.goal = -12
118             self.pose.r_knee.goal = 49
119         else:
120             self.pose.r_ankle_pitch.goal = 20
121             self.pose.l_ankle_roll.goal = 12
122             self.pose.l_knee.goal = -47
123         self.update_pose(1)

125         #Fuss absetzten
126         self.perform_task(foot_down,i,1,0.5)
127         #Gleichgewichtskorrekturen fuer rechts oder links
128         if i==1:
129             self.pose.l_ankle_pitch.goal = -5
130             self.pose.l_ankle_roll.goal = -15
131         else:
132             self.pose.r_ankle_pitch.goal = -24
133             self.pose.r_ankle_roll.goal = 15
134         self.update_pose(0.5)

136         #Wechsel des Schwerpunktes vom einen auf den anderen Fuss
137         self.animation_play(balance_shift)

139         #hinteren Fuss heben
140         self.perform_task(foot_up2,i,2)
141         ##Gleichgewichtskorrekturen fuer rechts oder links
142         if i==1:
143             self.pose.l_ankle_roll.goal = 20
144         else:
145             self.pose.r_ankle_roll.goal = -20
146         self.update_pose(1)
```

```
148         #hinteren Fuss vor bewegen
149         self.perform_task(foot_forward2,i,2)
150         #Gleichgewichtskorrekturen
151         self.pose.r_hip_pitch.goal = -100
152         self.pose.l_hip_pitch.goal = 100
153         self.update_pose(1)

155         #hinteren Fuss absetzten
156         self.perform_task(foot_down2,i,2,0.5)
157         #Gleichgewichtskorrekturen fuer rechts oder links
158         self.pose.r_hip_pitch.goal = -110
159         self.pose.l_hip_pitch.goal = 110
160         if i==1:
161             self.pose.r_ankle_pitch.goal = 52
162             self.pose.r_ankle_pitch.speed = 20
163         else:
164             self.pose.l_ankle_pitch.goal = -52
165             self.pose.l_ankle_pitch.speed = 20
166         self.update_pose(0.5)

168         #Roboter wieder gerade stellen
169         self.pose.l_ankle_roll.goal = 0
170         self.pose.r_ankle_roll.goal = 0
171         self.pose.l_ankle_roll.speed = 10
172         self.pose.r_ankle_roll.speed = 10
173         self.update_pose(1)

175         #Roboter wieder in Ausgangsstellung bringen
176         self.animation_play("walkready")

178     #Roboter steigt mehrere Stufen hoch, beginnend mit dem linken Fuss
179     def walk_stairs(self, steps=1, height=45):
180         self.walk_init()
181         for i in range(1,steps+1):
182             self.walk_step(height,i)
```