Wolfgang-OP:
A Robust Humanoid Robot Platform for Research and Competitions

Marc Bestmann[1] and Jasper Güldenstein[1] and Florian Vahl[1] and Jianwei Zhang[1]

***Abstract*— We present our open humanoid robot platform *Wolfgang*. The described hardware focuses on four aspects. Firstly, the robustness against falls is improved by integrating 3D printed elastic elements. Additionally, a high control loop frequency is achieved by using new custom control electronics. Furthermore, a torsion spring is applied to reduce the torque on the knee joints. Finally, computational power is provided through the combination of different processors. The paper also presents the ROS-based software stack that is used in RoboCup.**

I. INTRODUCTION

In the last years, many new humanoid platforms were developed and released open source. One major factor for this was the RoboCup Humanoid League, where many teams started their participation by using the commercially available, but open, platform Darwin-OP [14]. While this platform gave an easy start point, it showed its limitations when the game complexity increased, i.e. by switching from carpet to artificial turf and by usage of a real soccer ball. Many teams started to further develop the platform, especially in terms of actuator power, height, and computational capacity [10]. From this, new platforms emerged, which again influenced each other. In our opinion, this possibility to start with a complete platform and then improve certain aspects of it, facilitated the research in this interdisciplinary area.

In this paper, we present our open platform (OP) *Wolfgang* which is a descendant of the Nimbro-OP [32]. Our design goals were influenced by the fact that our robots are not only used for research in the lab, but also for participation in RoboCup Humanoid Soccer. Therefore, one main aspect is the robustness to falls, which happen often during the competition. For this, we added compliant elements to the robot's shoulders and head, as well as flexible bumpers to the torso. Another main aspect was the improvement of the electronics. We significantly improved the update rate of the communication with servos and sensors compared to its predecessors. Furthermore, we developed a low-cost parallel elastic actuator (PEA) to reduce the torque on the robot knees. As a fourth main aspect, we improved the computational power, especially in regards to neural networks, by using a combination of different hardware components. We estimate the material cost (excluding manufacturing and assembly) of our platform to be 11,000€.

This research was partially funded by the German Research Foundation (DFG) and the National Science Foundation of China (NSFC) in project Crossmodal Learning, TRR-169.

[1]All the authors are with the Department of Informatics, University of Hamburg, 22527 Hamburg, Germany [bestmann, 5guelden, 7vahl, zhang]@informatik.uni-hamburg.de

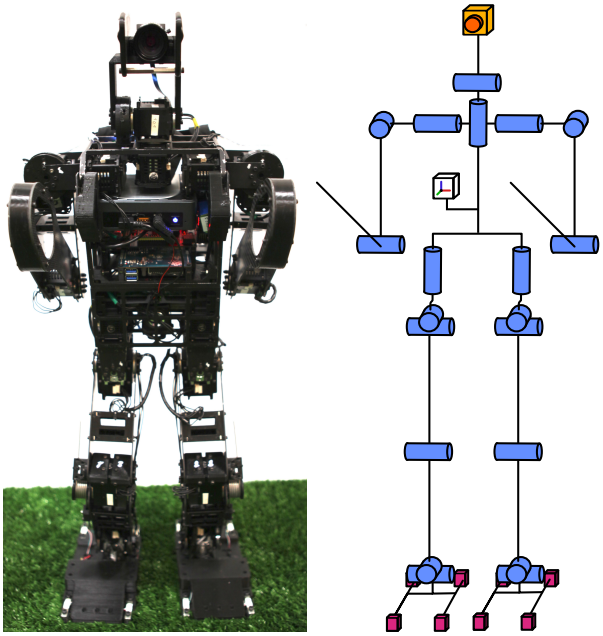

Fig. 1. Picture of the robot (**left**) and illustration of the kinematic chain (**right**). The 20 degrees of freedom are colored **blue**, except the series elastic actuators (**orange**) and the parallel elastic actuators (**yellow**). Positions of the force sensors (**red**), IMUs and camera are included.

Besides those hardware changes, we also briefly describe our RoboCup software architecture and our implementation of different motion skills, as well as our vision and behavior algorithms.

TABLE I
WOLFGANG-OP SPECIFICATIONS

Type	Specification	Value		
General	Height	80cm		
	Weight	7.5kg		
	Battery	5200mAh, 14.8V, 4-cell LIPO		
	Battery Life	25min standing, 10min walking		
	Material	Aluminium, Carbon, PLA, Ninjaflex		
PC	Name	NUC8i5BEK	Odroid-XU4	Intel NCS2
	Processor	i5-8259	Exynos5422	Myriad X
	Memory	8GB DDR4	2GB DDR3	
	Network	GigE	GigE	
	Wireless	2.4/5GHz		
Controller	Microcontroller	Teensy 4.0 + FT4232H		
	CPU	600MHz Cortex-M7		
	Control loop	700Hz - 1000Hz		
Sensors	Camera	Basler acA2040-35gc		
	Camera Lens	Computar Lens M0814-MP2 F1.4 f8mm 2/3"		
	IMU	MPU6500		
	Foot Pressure	Custom strain gauge based		
Actuators	Name	MX-64	MX-106	XH540
	Count	8	10	2
	Stall torque	7.3Nm	10Nm	11.7Nm
	No load speed	78 rev/min	55rev/min	46 rev/min

II. RELATED WORK

Many robot platforms have been developed in recent years, of which several are described in [12]. Some focus on bringing optimal performance at significant costs while others focus on using more low-cost hardware. A survey of current robot platforms has been performed in [31]. Commercially available child-sized robots such as the Robotis OP3[1] (as well as its predecessor the Darwin-OP [14]) and NAO robot [13] enable research groups to focus on developing software rather than hardware. While a considerable amount of research has been done using these platforms, they are limited both in computational power and mechanical ability.

Multiple open source child-sized humanoid robots were developed in recent years. These include the Sigmaban+ robot by the four times RoboCup champion team Rhoban [11] (with a buying price of 30,000€), the predecessor of the platform presented in this paper, the Nimbro-OP [32], and its mostly 3D printed successor, the igus platform [2]. While these platforms are relatively low cost, the iCub [24] is a robot equipped with many more sensors and degrees of freedom but at a much higher price of approximately 250,000€.

Furthermore, different adult-sized humanoid robots are commercially available such as Robotis' THORMANG3[2] or PAL Robotics' TALOS [34]. The adult-sized open-source robots Nimbro-OP2X [8] is reproducible with at a relatively low cost of 20,000€ to 40,000€.

Multiple robots were developed at companies or research institutions such as Boston Dynamics' Atlas [23], Honda's Asimo [33], and Toyota's T-HR3[3], but these are neither open-source nor commercially available.

III. HARDWARE

The general kinematic structure of the Wolfgang platform is similar to the original Nimbro-OP [32]. We made some changes to the mechanics and completely changed the electronics of the robot. These are described in the following.

A. Elastic Elements

The RoboCup teams WF Wolves and Taura Bots developed series elastic actuators (SEAs) using compliant polyurethane elements [22]. They also added the compliant elements in the robot's shoulder roll motor. These prevent damage to the gears when the robot falls on its side. We adopted these elements for the shoulder roll motors but later improved them to be 3D printable to ease manufacturing while maintaining a similar geometry. The used material is NinjaFlex[4], a thermoplastic polyurethane with a shore hardness of 85A. To reduce cost and complexity, we also did not include a position sensor to measure the deformation

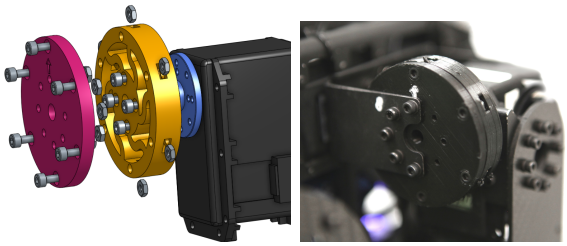

Fig. 2. Exploded view of the compliant element (**left**) and photo of assembly (**right**). introduced by the elastic parts The inner part (**yellow**) is elastic and made of NinjaFlex. The outer part (**red**) is stiff and made of PLA. It is necessary to provide the same mounting as the motor horn (**blue**). Standard hex nuts are inserted in both parts to provide threads for mounting.

of the elastic element since the exact position of the shoulder roll motor is not crucial in our use-case. Instead of using an aluminum part to connect it to the robot, we use a printed part made of PLA (see Figure 2). Since these can both be produced on a low-cost fused deposition modeling (FDM) printer and the material is widely available, this facilitates production. Furthermore, due to the rapid prototyping nature of 3D printing, multiple versions with different degrees of compliance can be created in a short time.

The same element was also installed in the robot's neck to further reduce the risk of damage to the neck when falling caused by the rapid deceleration of the considerable mass of the camera and lens. Since this introduces an uncertainty in the measurement of the kinematic chain, we installed an additional IMU in the robot's head.

Multiple approaches have been proposed to prevent damage to the robot due to falls. Firstly, a rigid structure to withstand the impact may be used [17]. Secondly, active or passive compliance mechanisms such as airbags [16] or elastic elements such as piano wire [11] can be employed. Furthermore, the robot can assume a pose in which damage is prevented as described for our robot in [5]. We decided for passive compliance since no maintenance has to be done between falls as presented in [16] and only minimal weight is added. 3D printed elastic elements at the front and back of the robot's torso, also made of NinjaFlex, were designed and added. In the case of a sagittal fall, these elements will first come in contact with the ground, thus reducing the impact force applied on the robot. The elasticity can easily be controlled by using different amounts of infill material.

Mechanical development is done using the online platform Onshape. This allows anybody to view the model, without installing additional software. Furthermore, it is possible to automatically generate the corresponding URDF by using a tool[5]. This prevents differences between the CAD and the URDF model. It also enables a faster hardware development cycle, as no manual changes to the URDF are necessary.

B. Parallel Elastic Actuator

The robot's kinematic structure forces it to bend its knees during walking and many motion skills. Thereby, the knee

[1] https://emanual.robotis.com/docs/en/platform/op3/introduction/

[2] https://emanual.robotis.com/docs/en/platform/thormang3/introduction/

[3] https://global.toyota/en/newsroom/corporate/30609642.html

[4] https://ninjatek.com/ninjaflex/

[5] https://github.com/Rhoban/onshape-to-robot

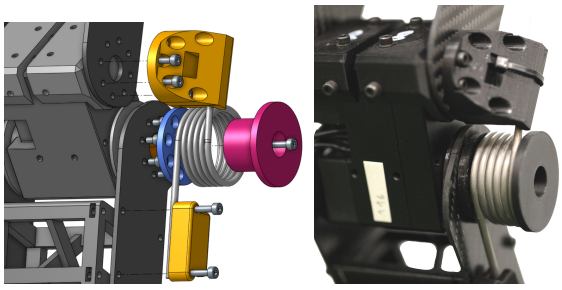

Fig. 3. Exploded view of the torsion spring assembly on the knee (**left**) and photo of assembly (**right**). The spring (**grey**) is fixed centered on the rotation axis by a 3d printed part (**red**). A second part (**blue**) countersinks the screw heads. The arms of the spring are fixed to below and above the joint (**yellow**).

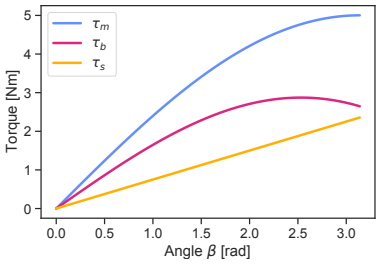

Fig. 4. Plot of the theoretical relationship between knee angle β and torque for the presented robot platform with $x = 0.13$ while standing with both legs on the ground. All torques are displayed without sign for better comparison. The torque without spring τ_m (**blue**) is higher than using a spring τ_b (**red**). When the leg is free, an additional torque τ_s (**yellow**) is created when using a spring. Still, this leads to a minimization of maximum torque.

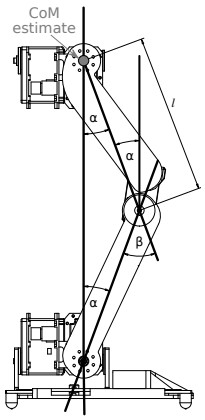

Fig. 5. Visual explanation of the symbols used in the formulas.

is mostly bent, e.g. by around $60°$ when standing in a pose from which it can directly start walking. This leads to high torques that are acting on the motor, which is undesirable due to high energy consumption and eventual overload issues. Furthermore, the knee joints experience the highest peak loads in most commonly performed motions, thus limiting the performance of the robot. To tackle these issues, we added a torsion spring on the side of each knee, which produces a counter torque (see Figure 3), turning these motors into parallel elastic actuators (PEAs). Similar approaches to reduce energy consumption by applying different springs have already been investigated in [28], [19], and [15]. The use of torsion springs was also investigated for exoskeletons, e.g. to reduce the energy needed during cycling [6]. Different other spring based exoskeletons were developed for industrial applications [18]. While clutching mechanisms for PEAs exist [25] we decided against it, since we estimated the disadvantages of increased weight and complexity to outweigh the benefits of the reduced required torque during specific motion (e. g. the swing phase of the walking).

In a static case, we can assume that the knee torque resulting from the mass of the robot (τ_m) follows the equation for an inverted pendulum.

$$\tau_m = \frac{1}{2} mgl \cdot \sin \alpha \tag{1}$$

Where m is the mass of the robot above the knees, which is divided by two since it is distributed over two legs. Furthermore, g is the gravitation constant, l the distance to the mass point and α is the angle of the pendulum. The torque resulting from the torsion spring (τ_s) can be computed using Hooke's law.

$$\tau_s = -k\beta \tag{2}$$

Where k is the spring constant and β the bending angle of the spring, which is in our case the bending angle of the knee. It would also be possible to mount the spring already bent at $\beta = 0$, resulting in an additional offset in equation 2. However, this reduces the bending range of the spring and we need almost $180°$ movement.

We can now exploit the fact that the robot's center of mass is typically above its feet when it is upright. Otherwise, it would fall due to the ZMP leaving the support polygon. Furthermore, our robot, as most humanoids, has the same link length for upper and lower legs. For simplicity, we assume that the CoM is at the hip joint. Typically, it is a few centimeters higher than that and changes based on arm and head movement. Still, this allows us to do a rough estimation of t_m which is generally lower than the actual torque. See Figure 5 for a visual explanation.

Based on this, we can assume:

$$-\beta \approx 2\alpha \tag{3}$$

We can choose k freely as long as we find a spring with this constant that fits on the robot. We choose as following.

$$k = x \cdot \frac{1}{2} mgl \tag{4}$$

Where x is a scaling factor that we will use later. If we now combine formulas 1 and 2 we can get the resulting torque on the knee when the robot is standing on both legs τ_b.

$$\tau_b = \tau_m + \tau_s = \frac{1}{2} mgl \cdot \sin \alpha - k\beta \tag{5}$$

Inserting 3 and 4 into 5 leads to the following equation.

$$\tau_b \approx \frac{1}{2} mgl \cdot \left(\sin \left(\frac{1}{2} \beta \right) - x \cdot \beta \right) \tag{6}$$

Similarly, we can compute the torque τ_o when this leg is supporting the complete weight of the robot, as well as τ_a when the leg has no ground contact. We neglected the torque resulting from the mass of the foot for τ_a.

$$\tau_o \approx 2\tau_b \qquad\qquad \tau_a \approx \tau_s \tag{7}$$

We can use the factor x to scale the spring and thereby change the trade-off between applied torque in different knee angles for the cases of τ_b, τ_o, and τ_a. There are different possibilities to choose this factor based how much time the knee spends in certain angles and in which of these cases, as well as if the focus is on reducing mean or maximal torque.

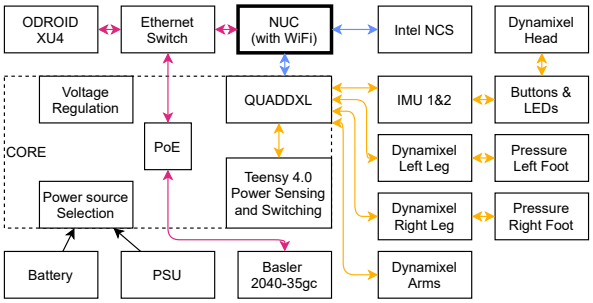

Fig. 6. Electronic systems in the Wolfgang-OP. The Intel NUC connects to the QUADDXL on the Wolfgang CORE and the Intel NCS2 using USB (**blue**). The Basler camera, ODROID-XU4, and NUC are connected via Ethernet (**red**). The Basler camera is powered using Power over Ethernet induced on the Wolfgang CORE. Sensors (IMUs, foot pressure, and voltage and current sensing) and actuators are connected to the QUADDXL on four separate RS485 buses (**yellow**) and communicate through it to the NUC. The Wolfgang CORE also handles the selection of either a battery or power supply unit. It furthermore handles the voltage regulation for the ODROID-XU4 and Ethernet switch and the control of the power supply to the servos and sensors.

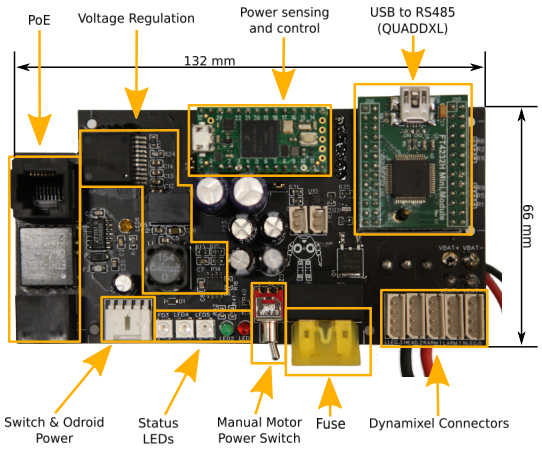

Fig. 7. Overview of the components and functionalities on the Wolfgang CORE board, the main electronics board of the Wolfgang platform.

For this robot, the goal was to reduce mean torque to improve energy consumption but also to reduce $\max \tau_b$ since the knee joints are close to their limit during stand-up motions (see also Figure 12), leading to issues with older servos or when the battery is running low. At the same time, we need to ensure $\max \tau_a \leq \max \tau_b$, since otherwise we would create to high torque when pulling the legs towards the body while lying on the back during stand-up. When we assume that the knee angle is always between 0 and π, we result with $x \approx 0.15$. From this we can compute the corresponding spring parameter for our robot.

$$k = 0.15 \cdot \frac{1}{2} \cdot 6.0\text{kg} \cdot 9.81 \frac{\text{m}}{\text{s}^2} \cdot 0.17\text{m} = 0,75\text{N m} \qquad (8)$$

Due to size and availability we chose a spring with $k = 0.76$. This leads to a theoretical reduction of 37% for the maximal τ_b torque. A plot of the theoretical torques is shown in Figure 4 and the results are presented in Section V.

C. Electronics

To efficiently use a robot, a good interface to its hardware is needed. For this, we developed the CORE board (see Figure 6 and 7), which builds upon our previous work on high-frequency hardware control [4]. It provides four RS-485 buses with up to 12 MBaud. Since the used Dynamixel Motors[6] are only able to communicate at 4 MBaud [4] we use this frequency on all buses. Additionally, the CORE board handles the power management of the robot. A battery and an external power supply unit (PSU) can be connected at the same time which allows battery swapping without loss of power. Uncontrolled charging of the battery by the PSU or reverse current into the PSU is prevented by a double anode single cathode diode. The input voltage is regulated to 5V and 9V for the ODROID-XU4 and network switch respectively. To power the camera, a Power over Ethernet (PoE) power sourcing equipment is implemented on the board.

Both, a manual switch and the installed Teensy 4.0 microcontroller can be used to enable and disable power to the devices on the Dynamixel bus, i.e. actuators and sensors. The microcontroller is also used to measure the input voltages of the power supplies and the cell voltages of the battery as well as the current consumed by the whole robot using a hall effect based sensor. It is connected to one of the RS-485 buses like the other sensors and actuators in the robot.

To be able to place the IMUs at arbitrary positions in the robot (i.e. the torso and the head), we developed an IMU module that connects to the Dynamixel bus. This reduces the need for running additional cables through the robot. By using an IMU in the robot's head we hope to improve our results from reprojecting the camera image to real-world coordinates by improving the accuracy of pitch and roll angle. The used sensor is an MPU6500. As a microcontroller, we chose the ESP32 since it features two cores. One of these cores is dedicated to communication on the DXL bus, while the other reads the sensor and runs a complementary filter based on [35]. Additionally, we added buttons and RGB LEDs to the IMU module in the torso for interaction and status indication.

In [4] we introduced our foot pressure sensors. While keeping the analog electronics the same, we have changed the microcontroller to an ESP32 to share code with the IMU module and avoid errors due to conflicts between the reading of the sensors and bus communication.

When the robot is turned off suddenly while standing, its joints will rotate while falling. This can induce voltages in the motor controller chips which are higher than the ones they are rated for, thus leading to damage [1]. To prevent this, we added a transient-voltage-suppression (TVS) diode to all MX servos.

A single Basler acA2040-35gc camera is used on the Wolfgang-OP platform. It features a global shutter and, compared to the commonly used webcams, a higher dynamic

[6]https://emanual.robotis.com/docs/en/dxl/

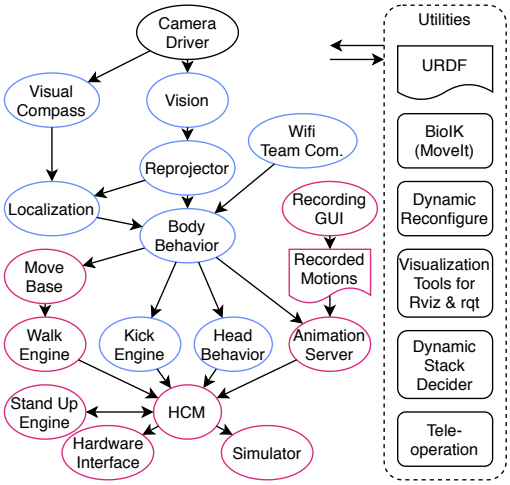

Fig. 8. Simplified overview of the ROS based software architecture. The vision and high-level behavior (**blue**) is specific for RoboCup competitions. Still, the motion part (**red**), including falling handling and walking, can be easily used for other purposes due to the decoupling by the Humanoid Control Module (HCM).

range and lower noise due to a larger photosensor. The global shutter ensures that all pixel measurements are done at the same time reducing errors due to wrong timestamps for parts of the image, especially if the robot moves during the exposure. The camera uses Gigabit Ethernet for communication and power (PoE). Data is transferred via the GigE Vision interface standard. Images are captured at 2048 x 1536 px but they are binned 4 times to reduce the noise and dimensionality for our vision system. While the camera supports up to 36 FPS at full resolution, we limit it to the framerate of the vision pipeline of currently 16 FPS.

D. Computer

The robot's software mainly runs on an Intel NUC 8i5BEK. It is used to execute real-time critical and resource-intensive software, e.g. the robot's hardware control, motion, and vision. An Odroid-XU4 single board computer is used for additional less real-time critical tasks, for example, the localization or the high-level behavior. The communication between these computers is done via Ethernet. Other devices, e.g. the camera or external computers for maintenance and configuration, are also connected to this network.

For neural network inference, an Intel Neural Compute Stick 2 is used. It is based on the Intel Movidius Myriad X Vision Processing Unit. This hardware accelerator is used to reduce the CPU load and power consumption while maintaining an equivalent processing speed compared to the CPU inference. The device is connected via USB 3.0 to the NUC and uses about 2 W [20]. With 35 g it is also a significant weight reduction considering the 102 g of the Jetson TX2 which we used previously for this task.

IV. SOFTWARE

Our software is based on ROS [27]. The main advantage is the large library of existing software packages and tools. To take full advantage of this, we use the standard messages and packages as far as possible. This also facilitates the use of parts of our software by others. An overview of the architecture can be seen in Figure 8. For development and testing, the Wolfgang-OP is available for three different simulators (see Figure 9).

Fig. 9. Screenshots from different available simulation environments. From left to right: Webots, Pybullet, Gazebo.

One distinctive feature of our architecture is the Humanoid Control Module (HCM) [5] that adds an additional abstraction layer between the hardware and high-level behavior. It allows the usage of software originally meant for wheeled robots, i.e. *move_base* [21], and creates a loose coupling of the different motion skills. Furthermore, it takes control of the robot in case of falls and makes sure that it lands on one of its elastic elements to prevent damage. Afterward, the HCM will automatically perform a stand-up motion to bring the robot into a standing pose. This encapsulation of low-level, reflex-like behavior allows the high-level behavior to concentrate on high-level goals. The fall detection uses a threshold-based classifier that uses the torso's angular velocity and orientation in fused angles [3] based on IMU data. It predicts the direction of the fall to allow the robot to go in the correct safe pose. Fall measurements are only invoked if the classifier predicts the same results over 10 ms to prevent false positives from sensor noise.

The high-level behavior is split into two parts. The *head behavior* controls the facing direction of the camera to gather information about the environment. It can get different goals, e.g. find the ball or gather localization information, and uses the current world model to perform corresponding head movements. The *body behavior* is the highest instance of decision making. It decides where the robot should go, if it should perform a motion skill, and tells the head behavior what it should look for. This modularity leads to less complex code and enables the reuse of parts, e.g. the robot can be teleoperated by directly controlling the HCM.

All behavior modules are implemented using the Dynamic Stack Decider (DSD), a lightweight open-source control architecture [26]. It allows to define complex behaviors and was also used for different service robot scenarios outside of RoboCup.

A. Sensing

Most of our vision pipeline is already described in a previous work [9]. It can detect balls, lines, goalposts, obstacles,

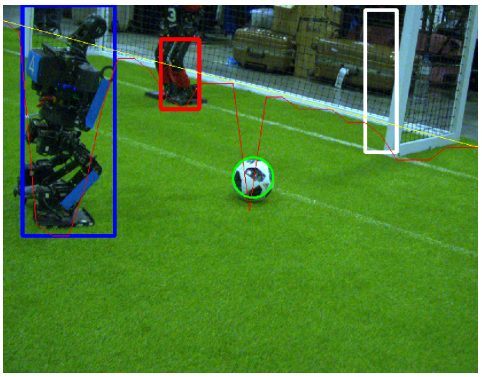

Fig. 10. Vision debug output showing the detected robots divided in their team colors (**red/blue box**), balls (**green circle**), goalposts (**white box**), and the convex field-boundary (**yellow line**).

and the field boundary as seen in Figure 10. We recently changed our neural network object detection to a YOLO V3 tiny architecture which we use to detect balls and goalposts. YOLO V3 tiny is a simplified version of the established YOLO V3 [29] architecture. The neural network inference is done on the Intel Neural Compute Stick 2 which we described in section III-D. Based on the camera's orientation, the object detections and segmentations are reprojected from the image space onto the field plane for further processing in our behavior, path-planning, and localization. Our localization uses a particle filter to estimate the robot's position using the line, goalpost, and odometry inputs. Additionally, we can use our visual compass for an estimation of the robot's orientation relative to the background which is necessary to solve the symmetry problem of a soccer field.

B. Motion

The three main motion skills that the robot has to perform (walk, kick, stand up) are all implemented by parameterized Cartesian quintic splines with additional stabilization controllers. Our previous approaches were mostly based on interpolation of keyframes in joint space that were recorded on the robot. The new approach has multiple advantages. The manual fine-tuning is more straight forward, as parameters relate to a Cartesian unit, rather than a joint angle. Furthermore, the number of parameters is lower, making optimization easier. It is also possible to add a goal, e.g. the position of the ball, to adapt a motion to the current situation. For the stand-up motion, interpolation in Cartesian space is necessary to ensure a straight vertical trajectory of the upper body.

The stabilization of the motion skills is improved by adding a PID controller based on the fused angles [3] of the torso which is measured by the IMU sensor. The control signal is also applied in Cartesian space. Finally, inverse kinematics (IK) is needed to compute the goal angles of the joints. We use the MoveIt [7] IK interface, which allows us to choose between different IK implementations. We currently apply BioIK [30] to all tasks since it is generic, fast, and allows us to solve for different types of goals, e.g. the *look-at* goal in our head behavior.

V. EVALUATION

Previous versions of this platform were successfully used in multiple RoboCup tournaments. One technical challenge in RoboCup was the push recovery, where the robot has to withstand an impact while walking. In this challenge, our platform scored first place in the teen-size category in 2019, proving its stability.

A. Robustness

The usage of good fall detection together with the bumpers and compliant elements significantly reduced the number of gears that needed replacement. Even after numerous hours of use, the 3D printed elements showed no damage while some of the previous polyurethane-based ones broke.

Without the compliant element in the neck joint, we frequently encountered failure in the 3D printed neck connector. We tried using a metal connector, but this led to damages to the plastic servo casing. After integrating the compliant element in the neck joint no further failures occurred.

To test our falling detection, the IMU data was recorded while the robot was falling, standing, and walking. Manual perturbations were performed to produce edge cases where the robot is almost falling. The training set consists of 13,944 frames where the robot is not falling and 7,961 where it is falling in one of the four different directions. For evaluation, a set of 9,436 frames where the robot is not falling is used to compute the number of false positives. Furthermore, the impact times for five falls while standing and walking for each direction (together 40) were labeled to compute the lead time. The classifier predicts a fall with a minimal lead time of 295 ms and a mean lead time of 596 ms without false positives. It is possible to achieve higher minimal lead times if false positives are accepted. Since the maximal observed time between detection of a fall and the robot reaching a safe pose is 231 ms, the resulting minimal lead time is sufficient.

B. Control Loop

Our custom electronics can read and write all servos and sensors with 750 Hz if only the positions of the servos are read. When reading positions, velocities, and efforts of the servos, the rate is 715 Hz. This rate is limited by the fact that the four buses can not be evenly distributed on the devices due to the kinematic structure of the robot. The slowest buses are the ones on the legs which need to control six servos and a foot sensor each. If no foot sensors are used, a control frequency of 1,000 Hz is possible. For more details on this and comparison with other approaches see our previous paper [4].

C. Torque Reduction

To evaluate the PEA performances, the joint feedback was recorded during the performance of different typical motions with and without springs. The servos are only able to estimate the torque based on the sensed current. Therefore, the recorded torques are not precise. Still, since the experiments were performed on the same robot, we assume the values are comparable to each other. Naturally,

TABLE II

COMPARISON OF KNEE TORQUES

Motion		No PEA [Nm]	PEA [Nm]	Relation PEA / no PEA
Standing in a walk ready pose	mean	0.48	0.31	0.66
	max	0.81	0.80	0.99
Walking	mean	2.47	1.91	0.78
	max	9.4	7.29	0.78
Stand up from the front	mean	1.85	1.61	0.87
	max	7.58	6.60	0.87
Stand up from the back	mean	1.90	1.49	0.78
	max	7.03	6.80	0.97
Standing to squatting	mean	1.29	0.48	0.37
	max	2.85	1.17	0.41
Squatting to standing	mean	2.75	1.74	0.63
	max	7.08	3.75	0.53

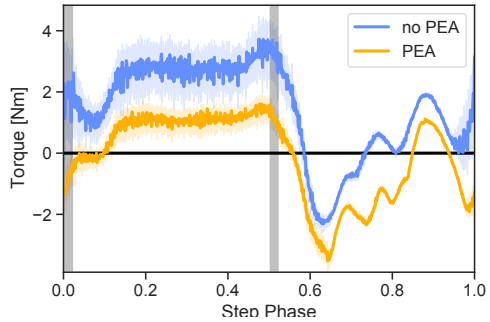

Fig. 11. Exemplary plot of a knee servo during a walking, with (**yellow**) and without (**blue**) torsion spring. The double support phases are marked in **grey**. The PEA reduces the load in the first half where the servo is supporting the robot's weight. In the second half, additional torque is generated since the leg is not under load.

the torques are also dependent on the specific parameters, e.g. how high the foot is lifted from the ground during a step. All experiments were performed with the same parameters that are used in our competitions. The results are displayed in Table II. The mean torque is evaluated to show how the energy consumption is improved and the maximal torque to show how much the servos are stressed. It is visible that using the PEA improves the mean torque in all motions. Even during walking, where the PEA on the swing knee creates additional torque, a significant overall reduction can be measured (see Figure 11). A similar pattern can be seen when standing up from the back (see Figure 12). Here, both feet need to be pulled towards the torso without ground contact, therefore additional torque is created by the PEAs. Still, the overall mean torque remains lower as the energy saved during later phases of the motion compensates for this. The maximal torque values show almost no change in some motions. This value is influenced by the fact that the torque is often not equally distributed between both legs when the center of mass is shifted slightly to one side. Furthermore, this value is stronger influenced by the noisiness of measurements.

D. Computational Performance

The current accuracy of the vision pipeline is shown in table III. The metrics and evaluation data set are identical

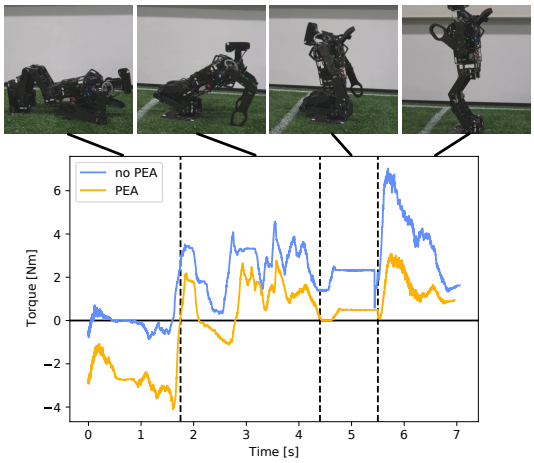

Fig. 12. Exemplary plot of a knee servo during a stand up motion from lying on the back, with (**yellow**) and without (**blue**) torsion spring. In the first phase, the feet are pulled towards the torso, requiring a higher torque when using the PEAs. Afterward, the robot tilts onto its foot soles. In the third phase, the robot briefly stays in a squatting pose to stabilize. Finally, the robot gets up to a standing pose. In the last three phases, the robot's weight rests on the feet where the PEA solution requires lower motor torque.

TABLE III

EVALUATION OF THE VISION PIPELINE (MEAN JACCARD-INDEX)

Version	Ball	Field boundary	Line	Goalpost
Current (1.3.3)	**0.794**	0.922	**0.058**	**0.658**
Previous (1.1.0) [9]	0.677	**0.925**	0.021	0.183

to the ones proposed in [9]. The current ball and goalpost detection performed significantly better than the approach described in [9] due to the new YOLO v3 tiny. The field boundary score decreased slightly overall due to minor modifications which were necessary to prevent issues when a second soccer field is visible in the image. A case that often happens during RoboCup tournaments. These changes also improved the execution time slightly. The vision runs at 20.7 FPS when using all cores of the Intel NUC. When using a single core for the vision to reserve performance for other tasks, the frame rate drops to 11.2 FPS. Running the neural networks on the NCS2 Vision Processing Unit increases the FPS to 15.67 while still using one CPU core. The remaining cores of the NUC are sufficient to run the motion part of the software stack at 700Hz.

VI. CONCLUSIONS

The Wolfgang-OP was designed to increase robustness as well as performance while still keeping the robot low-cost. An early and reliable falling detection together with the 3D printed compliant elements ensure high robustness against falls. Reacting fast to falls requires a high control loop rate which is achieved by our custom electronics. Furthermore, the load on the robot's knees is reduced by integrating a torsion spring. Our paper provides the formula to choose the correct stiffness for humanoid robot knee joints. Additionally, we investigated how computational performance is improved by integrating a tensor processing unit (TPU).

In the future, the robot could be improved by adding a second rotary encoder after the compliant elements. This would enable us to read the correct joint position and estimate the torque by measuring the rotation of the compliant part. Software-wise, switching to ROS 2 would reduce the overhead of message passing.

The robot's hardware, software, and simulation environments are open source and available here: **https://github.com/bit-bots/wolfgang_robot**

ACKNOWLEDGMENT

Thanks to the members of the Hamburg Bit-Bots for helping to develop and test this platform.

REFERENCES

[1] NUbots Hardware Overview and Specifications. `https://nubook.nubots.net/system/hardware/overview`. (visited on Jan. 21, 2021).

[2] P. Allgeuer, H. Farazi, M. Schreiber, and S. Behnke. Child-sized 3D Printed Igus Humanoid Open Platform. In *International Conference on Humanoid Robots (Humanoids)*. IEEE, 2015.

[3] Philipp Allgeuer and Sven Behnke. Fused Angles: A Representation of Body Orientation for Balance. In *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015.

[4] Marc Bestmann, Jasper Güldenstein, and Jianwei Zhang. High-Frequency Multi Bus Servo and Sensor Communication Using the Dynamixel Protocol. In *Robot World Cup*. Springer, 2019.

[5] Marc Bestmann and Jianwei Zhang. Humanoid Control Module: An Abstraction Layer for Humanoid Robots. In *International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 2020.

[6] Ronnapee Chaichaowarat, Diego Felipe Paez Granados, Jun Kinugawa, and Kazuhiro Kosuge. Passive Knee Exoskeleton Using Torsion Spring for Cycling Assistance. In *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017.

[7] Sachin Chitta, Ioan Sucan, and Steve Cousins. Moveit! *IEEE Robotics & Automation Magazine*, 19(1), 2012.

[8] G. Ficht, H. Farazi, A. Brandenburger, D. Rodriguez, D. Pavlichenko, P. Allgeuer, M. Hosseini, and S. Behnke. NimbRo-OP2X: Adult-Sized Open-Source 3D Printed Humanoid Robot. In *International Conference on Humanoid Robots (Humanoids)*, 2018.

[9] Niklas Fiedler, Hendrik Brandt, Jan Gutsche, Florian Vahl, Jonas Hagge, and Marc Bestmann. An Open Source Vision Pipeline Approach for Robocup Humanoid Soccer. In *Robot World Cup*. Springer, 2019.

[10] Reinhard Gerndt, Daniel Seifert, Jacky Hansjoerg Baltes, Soroush Sadeghnejad, and Sven Behnke. Humanoid Robots in Soccer: Robots Versus Humans in RoboCup 2050. *IEEE Robotics & Automation Magazine*, 22(3), 2015.

[11] Loïc Gondry, Ludovic Hofer, Patxi Laborde-Zubieta, Olivier Ly, Lucie Mathé, Grégoire Passault, Antoine Pirrone, and Antun Skuric. Rhoban Football Club: RoboCup Humanoid KidSize 2019 Champion Team Paper. In *Robot World Cup*. Springer, 2019.

[12] Ambarish Goswami and Prahlad Vadakkepat. *Humanoid Robotics: A Reference*. Springer, 2019.

[13] D. Gouaillier, V. Hugel, P. Blazevic, C. Kilner, J. Monceaux, P. Lafourcade, B. Marnier, J. Serre, and B. Maisonnier. Mechatronic Design of NAO Humanoid. In *International Conference on Robotics and Automation*. IEEE, 2009.

[14] Inyong Ha, Yusuke Tamura, and Hajime Asama. Development of Open Platform Humanoid Robot DARwIn-OP. *Advanced Robotics*, 2013.

[15] Clinton G Hobart, Anirban Mazumdar, Steven J Spencer, Morgan Quigley, Jesper P Smith, Sylvain Bertrand, Jerry Pratt, Michael Kuehl, and Stephen P Buerger. Achieving versatile energy efficiency with the wanderer biped robot. *IEEE Transactions on Robotics*, 36(3), 2020.

[16] S. Kajita, R. Cisneros, M. Benallegue, T. Sakaguchi, S. Nakaoka, M. Morisawa, K. Kaneko, and F. Kanehiro. Impact Acceleration of Falling Humanoid Robot with an Airbag. In *International Conference on Humanoid Robots (Humanoids)*. IEEE, 2016.

[17] Y. Kakiuchi, M. Kamon, N. Shimomura, S. Yukizaki, N. Takasugi, S. Nozawa, K. Okada, and M. Inaba. Development of Life-Sized Humanoid Robot Platform with Robustness for Falling down, Long Time Working and Error Occurrence. In *International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[18] Hyung Joo Kim, Jaeho Noh, and Woosung Yang. Knee-Assistive Robotic Exoskeleton (KARE-1) Using a Conditionally Singular Mechanism for Industrial Field Applications. *Applied Sciences*, 10(15), 2020.

[19] Matthieu Lapeyre, Pierre Rouanet, and Pierre-Yves Oudeyer. The Poppy Humanoid Robot: Leg Design for Biped Locomotion. In *International Conference on Intelligent Robots and Systems*. IEEE, 2013.

[20] Leandro Ariel Libutti, Francisco D Igual, Luis Pinuel, Laura De Giusti, and Marcelo Naiouf. Benchmarking Performance and Power of USB Accelerators for Inference with MLPerf. In *2nd Workshop on Accelerated Machine Learning (AccML), Valencia, Spain*, 2020.

[21] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige. The Office Marathon: Robust navigation in an indoor office environment. In *2010 IEEE International Conference on Robotics and Automation*, pages 300–307, May 2010.

[22] Leandro Tomé Martins, Christopher A Arend Tatsch, Eduardo Henrique Maciel, Reinhard Gerndt, and Rodrigo da Silva Guerra. A Polyurethane-Based Compliant Element for Upgrading Conventional Servos into Series Elastic Actuators. *IFAC-PapersOnLine*, 48(19), 2015.

[23] Gabe Nelson, Aaron Saunders, and Robert Playter. The PETMAN and Atlas Robots at Boston Dynamics. In *Humanoid Robotics: A Reference*. Springer, 2019.

[24] Alberto Parmiggiani, Marco Maggiali, Lorenzo Natale, Francesco Nori, Alexander Schmitz, Nikos Tsagarakis, José Santos Victor, Francesco Becchi, Giulio Sandini, and Giorgio Metta. The Design of the iCub Humanoid Robot. *International Journal of Humanoid Robotics*, 09(04), 2012.

[25] M. Plooij, M. van Nunspeet, M. Wisse, and H. Vallery. Design and evaluation of the Bi-directional Clutched Parallel Elastic Actuator (BIC-PEA). In *International Conference on Robotics and Automation (ICRA)*. IEEE, 2015.

[26] Martin Poppinga and Marc Bestmann. DSD - Dynamic Stack Decider: A Lightweight Decision Making Framework for Robots and Software Agents. *in peer-review process*, 2021. doi:`10.13140/RG.2.2.14585.01129`.

[27] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. ROS: an Open-Source Robot Operating System. In *ICRA workshop on open source software*, volume 3. Kobe, Japan, 2009.

[28] Katayon Radkhah, Christophe Maufroy, Moritz Maus, Dorian Scholz, Andre Seyfarth, and Oskar Von Stryk. Concept and Design of the Biobiped1 Robot for Human-Like Walking and Running. *International Journal of Humanoid Robotics*, 8(03), 2011.

[29] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[30] Philipp Ruppel, Norman Hendrich, Sebastian Starke, and Jianwei Zhang. Cost Functions to Specify Full-Body Motion and Multi-Goal Manipulation Tasks. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 2018.

[31] Saeed Saeedvand, Masoumeh Jafari, Hadi S. Aghdasi, and Jacky Baltes. A Comprehensive Survey on Humanoid Robot Development. *The Knowledge Engineering Review*, 34, 2019/ed.

[32] Max Schwarz, Michael Schreiber, Sebastian Schueller, Marcell Missura, and Sven Behnke. NimbRo-OP Humanoid Teensize Open Platform. In *In Proceedings of 7th Workshop on Humanoid Soccer Robots, International Conference on Humanoid Robots, Osaka*, 2012.

[33] Satoshi Shigemi. ASIMO and Humanoid Robot Research at Honda. In *Humanoid Robotics: A Reference*. Springer, 2019.

[34] O. Stasse, T. Flayols, R. Budhiraja, K. Giraud-Esclasse, J. Carpentier, J. Mirabel, A. Del Prete, P. Souères, N. Mansard, F. Lamiraux, J. Laumond, L. Marchionni, H. Tome, and F. Ferro. TALOS: A new humanoid research platform targeted for industrial applications. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, 2017.

[35] Roberto G. Valenti, Ivan Dryanovski, and Jizhong Xiao. Keeping a Good Attitude: A Quaternion-Based Orientation Filter for IMUs and MARGs. *Sensors*, 15(8), 2015.