Modeling and Implementations of a Quadruped Walking Dog Robot

Abizer Patanwala
Robotics Engineering
Worceter Polytecnic Institute
Worcester, MA
apatanwala@wpi.edu

Danniel Echeverria
Robotics Engineering
Worceter Polytecnic Institute
Worcester, MA
decheverria@wpi.edu

Pranav Moorthy
Robotics Engineering
Worceter Polytecnic Institute
Worcester, MA
pmoorthy@wpi.edu

***Abstract*—In this study we demonstrate the use of a commercially available quadruped robot dog (Petoi), performing various gaits. The gaits are generated using MATLAB and Gazebo and then imported into the robot using the Arduino IDE interface. Calibration on the robot is performed using mechanical jigs as well as offset parameters in the joint angles for each linkage. The results are forward walking gait a factor of 0.75 as well as a running gait**

Keywords—component, formatting, style, styling, insert (*key words****)***

I. Introduction

The field of walking robots has been one that has been of great importance in recent years. Boston Dynamics, being one of the leaders in this field, has demonstrated the robustness that can be achieved in the controls of walking robots like Spot (Fig 1), a quadruped walking robot that was the source of inspiration for this project.

Figure 1: Boston Dynamics Spot

A commercially available quadruped robot from Petoi called the Bittle Dog has been used in this analysis to demonstrate the effects of various gaits and the instability models associated with them while traveling various surfaces (flat and uneven/angled). We set our goal to demonstrate that commercially available open-source robotic platforms can be used in research to explore and evaluate current and future proposed control algorithms. We make use of popular modeling software such as SolidWorks, Arduino IDE, MATLAB and Gazebo which encompasses the multi-disciplinary aspect involved in the design of robots.

In a review of quadruped walking robots by Biswal et al [1] various robot designs are discussed across various periods of time, and these show their origin and evolution. The time periods are divided as Early 1900's which consists of quadrupeds such as the Chebyshev mechanism, Mid 1900's which consisted of the Kumo – I and the Phoney pony. This was followed by the quadruped of the Early 1990's and Early 2000's, after which various modern quadrupeds are discussed like the famous Spot of Boston dynamics. Various details of the quadrupeds are tabulated such as their gait, dimensions and degree of freedom. Moreover, the studied quadrupeds are split into two categories based on the type of actuator as hydraulic and electric. The aspects of performance compared are normalized speed, payload capacity and normalized work. In an article by Sen et al [2], the inverse kinematic analysis of a quadruped with 3 degrees of freedom on each leg. The inverse kinematics are derived analytically in MATLAB and the mathematical formulation is described. Bhatti et al. [3] presents a detailed description of the Gaits of various animals in a systematic fashion. The purpose of this paper was to allow animators of movies and games to be able to animate the Gaits of four-legged creatures more accurately. The Gaits of lions, tigers, and house cats have been studied. The analysis incorporated clips of live animal motion from various sources over the internet. The clips which were obtained are categorized into video types and then broken-down frame by frame. The walking and running motions of horses, lions, tigers, and house cats were separated frame by frame and divided into cycles such

that the animal is in the same pose at the end of a cycle as it was in the beginning. These gaits have been divided into Symmetrical and Asymmetrical gaits, which refer to the pattern of limb movements. 2-beat, 3 beat and 4 beat gaits have been described. The paper concluded with the conclusion shortly by summarizing the previously described sections. Yoneda et al [4] discusses using a forward wave gait of variable duty factor and crab angle. Duty factor is defined as the percentage of the total cycle time during which each leg is in support phase. The crab angle is the angle between the heading of the Robot (direction of leg 1) and the motion direction. This enables the robot to track a curved trajectory. The gait planning algorithm is then described in the following section. The walking robot is taken in the 2D space where the z (height of robot) is considered constant. The maximum duty factor is found for a given motion command by taking into account the maximum leg velocity of the robot for a given motion. The foot trajectory planning is described with respect to the vertical and horizontal motion followed by the stability analysis. The walking algorithm is then experimentally validated.

II. Modeling

A. Physical Modeling

We first start by understanding the construction of the Petoi Bittle Dog and determining the number of Degrees of Freedom (DOF) and overall architecture. We identify that each leg is composed of two links, driven by two parallel motors, a main chassis containing the controller and battery and a rotary actuated head. For walking and running gaits only the legs will make a difference, so we decide to keep the head joint stationary and face the default direction. The robot link lengths are provided by the manufacturer as well as the joint numbers associated with each motor (Fig 2).

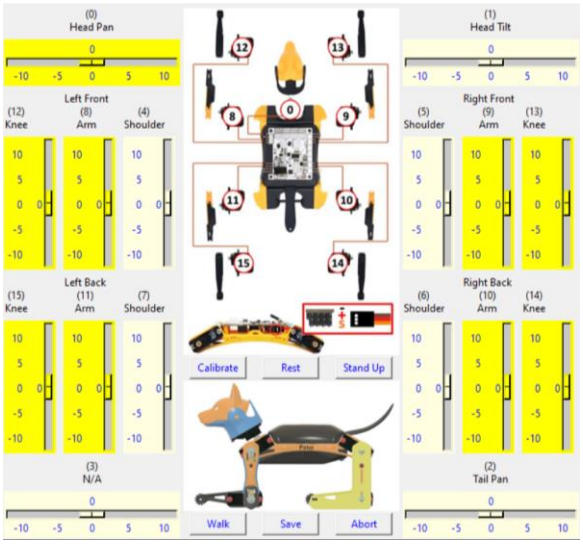


Figure 2: Joint number and assumed angle for home configuration

Since no 3D computer aided design (CAD) files were available, we searched GrabCAD and found a headless model in Standard for the Exchange of Product Data (STP) format created by Aparatum [5] based on Petoit pictures, videos and general dimensions provided by manufacturer's website. We reached out to the manufacturer, and they provided a head model in STP format to help us complete our model. Using SolidWorks 2021 we imported the models and assembled them into a single file. We validated the CAD file dimensions, by comparing them to the dimensions provided by the manufacturer as well as performing some physical measurements on the actual robot using a pair of calipers.

To determine the link limits and positive rotation direction we used the Petoi's main interface software UI to rotate each link. A remote control is also provided where sequences can be pre-saved onto specific buttons. When we first tried a pre-saved motion, we noticed that one of the joints was not moving and had to replace the damaged motor. We then tested again and noticed that the joint was not moving the same as the other legs. We determined that the robot needed to be re-calibrated to account for any differences in the new motor and assembly tolerances. A gauge fixture was provided to set each leg to a 90-degree angle where the motor encoder should read zero (Fig 3). A software offset can be applied at this point, or physically loosening the motor and re-attaching at the preferred orientation. Nategh et. al [6] explored a calibration method in a hexapod robot by using best line fit when comparing the desired and actual poses near the working space boundary of a stationary robot to determine actual values of kinematic parameters. With this approach an external camera was needed as well as a minimum number of measurements based on the number kinematic parameters. In our case, since a physical gauge was provided this greatly facilitated the calibration process and we decided not to implement such an approach.

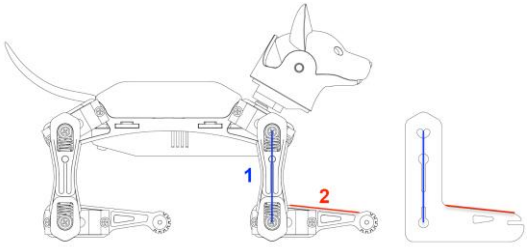


Figure 3: Calibration Jig for leg

To translate the 3D data into MATLAB or Gazebo a Unified Robotics Description Format (URDF) file is needed. Currently there is a URDF plugging for SolidWorks 2021 [7], but not 2022 which forced us to not use the latest software version. Using the plugin in SolidWorks, we were able to define which bodies constitute a specific link, set the coordinate frame for each joint as well as globally, organize the links in the correct kinematic chain order, set type of joint (prismatic or revolute), export surfaces and dynamic parameters such as moment of inertias (Fig 4).

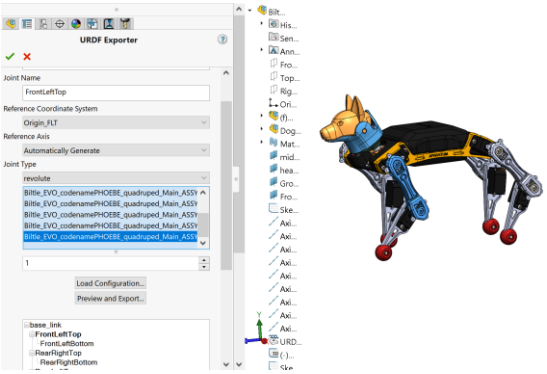


Figure 4: Joint model tree definition in SolidWorks

Once the Gaits are simulated and verified in MATLAB, we can import these into the robot controller using the Arduino IDE or Python interface. This interface takes as input the angles for each leg joint and are looped into a continuous motion for gaits, until an interrupt stops the motion (Fig 5). For a Gait, the first row represents the number of rows (35), desired pitch (0), desired roll (0) angles for the body, and the joint ratio (1) describing 1 gait cycle. (Note the pitch and roll angle is only functional if the on-board gyroscope is enabled) The Joint ratio is set to 1 to indicate a 1 to 1 conversion between the desired and the commanded joint angle. The 2nd and nth row represent the angles in degrees for each leg joint, starting with joint 8 (Left Front Arm) sequentially up to joint 15 (Left Back Knee) (Fig 2 & 5).

```
const char bk[] PROGMEM = {
  35, 0, 0, 1,
  46,  54,  46,  54,  -5, -23,  -5, -23,
  43,  58,  43,  58,  -5, -24,  -5, -24,
  ......
  52,  43,  52,  43,  -5, -21,  -5, -21,
  50,  48,  50,  48,  -5, -22,  -5, -22,
  47,  53,  47,  53,  -5, -23,  -5, -23,
};
```

Figure 5: Gait angles for Robot interface

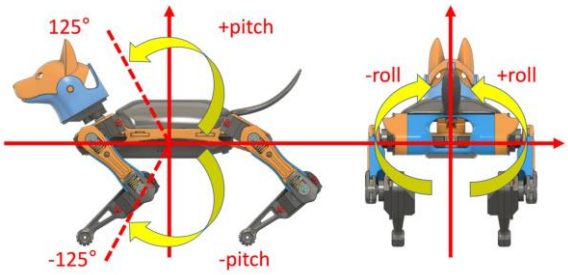


Figure 6: Body rotation of robot

To provide the inputs for a single pose, a similar methodology is used. For a pose, the joint angles start with the head joint (joint 0), is not looped, and the pose can be described with a single row of joint angles (Fig 7). Note that for a pose the initial pose angles are also included, from previous to new pose.

```
const char sit[] PROGMEM = {
1, 0, -30, 1,
    0,   0, -45,   0,  -5,  -5,  20,  20,  45,  45, 105, 105,  45,  45, -45, -45,};
```

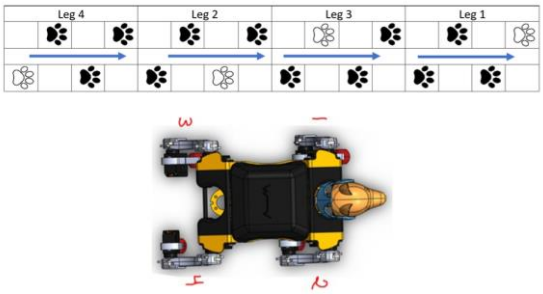
Figure 7: Single pose angles

B. Simulation Modeling

We first start with simulating the robot in MATLAB. Initially our aim is to make the quadruped robot walk in a straight line in MATLAB and then check it on Gazebo simulation and the hardware. For simulation in MATLAB, the kinematic model is used as shown in Fig.8.

Figure 8: MATLAB robot representation

The timestep used for simulation is 1 ms. First the gait phase diagram is designed for the quadruped robot using a Duty factor of $\beta = 0.75$. This means that the leg will be in support phase for 75% of the cycle time and in transfer phase for 25% of the cycle time. Fig.2 shows the kinematic phase diagram of the quadruped for $\beta=0.75$. The white area represents transfer phase, and the grey area represents support phase. This gait is very stable as it only moves one leg at a time, and the length of the support cycle is long. For walking robots, a $\beta > 0.5$ is typically considered minimum to maintain stability.



	β=0.75											
Leg 1												
Leg 2												
Leg 3												
Leg 4												
	0.08	0.17	0.25	0.33	0.42	0.50	0.58	0.67	0.75	0.83	0.92	1.00

Figure 9: Kinematic phase diagram of the quadruped robot

From the Calibration procedure mentioned in physical modelling, the following parameters of the robot are extracted and used for MATLAB simulation:

- L_1 = Arm Joint Length = 4.6 [cm]
- L_2 = Knee Joint Length = 4.8 [cm]
- L = Length of the robot = 10.5 [cm]
- W= Width of the robot = 9.7 [cm]

The body frame of the robot is located on the ground, at the center of the base. The world frame is located at [0,0,0]. Initially the body frame and the world frame are the same. The legs number are assigned in the following way with the robot facing front.

Leg 1 = Front Left

Leg 2 = Front Right

Leg 3 = Back Left

Leg 4 = Back Right

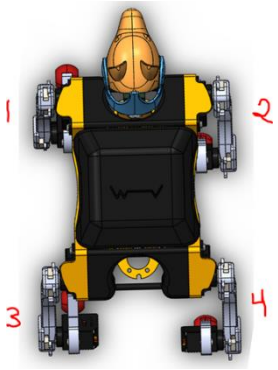


Figure 10: Robot leg order

Additionally, the following values are defined/calculated for the simulation:

- Velocity of the robot = 0.1 [m/s]
- Stride length of the robot = 0.1 [m]
- Initial joint values of the robot in Transfer phase: [0 0.7854] radians. These values were chosen arbitrarily.
- Initial joint values of the robot:
  - ***Support phase:*** [-0.77864, 0.59528] radians.
  - This value is also used for home configuration when spawning the robot in MATLAB. This was chosen as the end joint values of the transfer phase as this also marks the beginning of the support phase.
- ***Position of foot with respect to Body frame*** in support phase: In support phase the position of foot remains constant with respect to world and is given as follows:
  - Foot 1 = [9.3558 4.8500 -0.0000] radians
  - Foot 2 = [9.3558 -4.8500 -0.0000] radians
  - Foot 3 = [-1.1442 4.8500 -0.0000] radians
  - Foot 4 = [-1.1442 -4.8500 -0.0000] radians
  - These values were calculated using the home configuration and the length and the width of the robot.
- ***Position of arm joint with respect to Body frame***: The position of the arm with respect to the body remains constant through the motion.
  - Arm 1 = [5.2500 4.8500 7.9941] radians
  - Arm 2 = [5.2500 -4.8500 7.9941] radians
  - Arm 3 = [-5.2500 4.8500 7.9941] radians
  - Arm 4 = [-5.2500 -4.8500 7.9941]
  - These were calculated using the length and the width of the robot.
- ***Max height which the robot can reach***
  - Transfer phase = 0.7994 [cm]
  - This is chosen to be ($1/10^{th}$) of the height of the robot.

Instead of performing the Forward Kinematics (FK) and Inverse Kinematics (IK) for the whole robot, we performed these for each Leg separately treating it as a serial RR-robot (robot with two parallel revolute joints) see Fig 11. The equations used for the FK and IK are shown in Fig. 12 and Fig 13 respectively. For the IK we make use of the cosine laws to determine the joint angles. The foot positions of the leg with respect to the arm joint is denoted by [X_c, Y_c]. These are extracted from the trajectory profile we want the foot to follow (See Figure 14). This trajectory is ideal for either moving over obstacles or in non-even terrain, as the foot moves up vertically, then translating sideways and then moves back down in a vertical manner. The motion was segmented at the points of inflection for graphing simplicity.

NOTE: The joint lengths are denoted as L_1=a_1=arm joint length and L_2=a_2=knee joint length

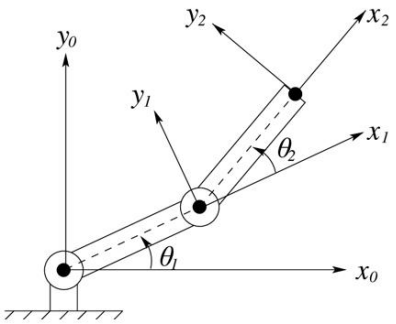


Figure 11: RR-robot representation

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} L_1 * \cos(\theta_1) + L_2 * \cos(\theta_1 + \theta_2) \\ L_1 * \sin(\theta_1) + L_2 * \sin(\theta_1 + \theta_2) \end{bmatrix} = \begin{bmatrix} h_1(\theta_1, \theta_2) \\ h_2(\theta_1, \theta_2) \end{bmatrix}$$

Figure 12: RR-robot Forward Kinematics

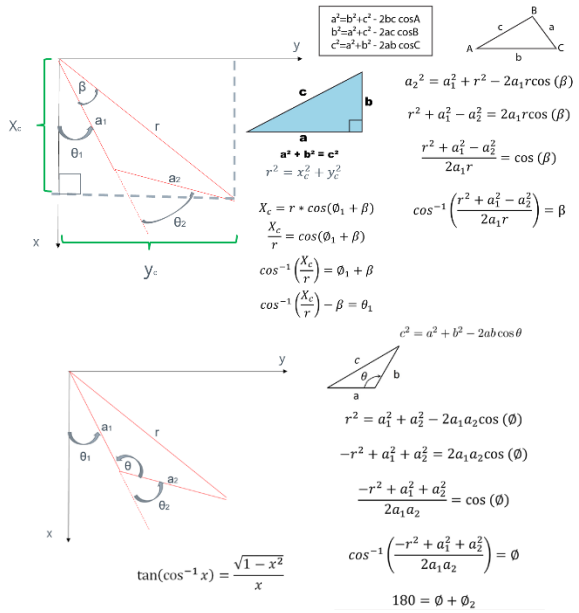
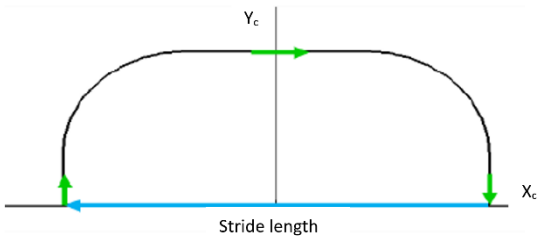


Figure 13: RR robot Inverse Kinematics



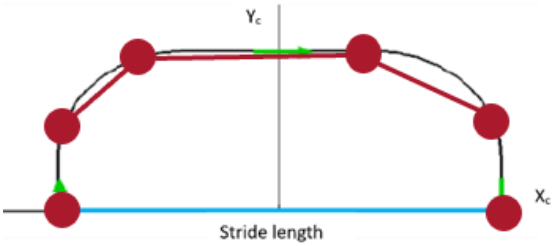


Figure 14: Foot trajectory with respect to ground

To correlate the joint velocities with respect to the foot tip velocities, we make use of the Velocity Jacobian (Jv) see Figure 15 equations for the RR-robot.

$$J_{v_i} = \begin{cases} z_{i-1} \times (o_n - o_{i-1}) & \text{for revolute joint } i \\ z_{i-1} & \text{for prismatic joint } i \end{cases}$$

$$J_v = \begin{bmatrix} \frac{\partial h}{\partial \theta} & \frac{\partial h}{\partial \theta} \\ \frac{\partial h}{\partial \theta} & \frac{\partial h}{\partial \theta} \end{bmatrix} = \begin{pmatrix} -L_2 \sin(\theta_2 + \theta_1) - L_1 \sin(\theta_1) & -L_2 \sin(\theta_1 + \theta_2) \\ L_2 \cos(\theta_1 + \theta_2) + L_1 \cos(\theta_1) & L_2 \cos(\theta_1 + \theta_2) \end{pmatrix}$$

Figure 15: Equation for Jacobian

III. RESULTS

The robot foot position with respect to the body and the ground are computed, discretized at the inflection points of the move profile in figure 14 and graphed for 1 leg. It's important to know that these would be identical from leg to leg, with just a phase shift based on the specific gait. The velocity is also computed and graphed (figure16).

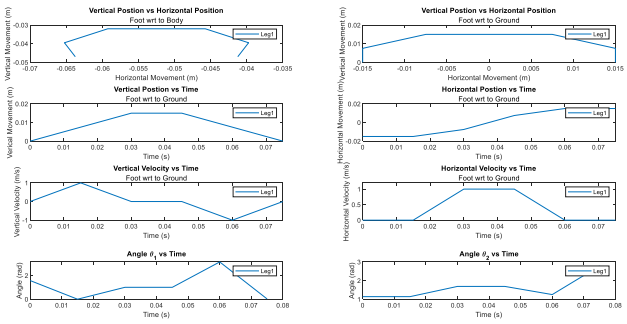


Figure 16: Foot Position and velocity graphs

These angles and velocities were then used in Gazebo to simulate the robot. For this section, a simplified base model from Daemiac called DreamWalker [8] was used. The reason for selecting this package was to utilize the main controller for path following. Following Parameters were first measured:

- L_1 = Arm Joint Length = 5.5 [cm]

- L_2 = Knee Joint Length = 6.5 [cm]
- L = Length of the robot = 18.9 [cm]
- W= Width of the robot = 16.2 [cm]

There were 2 major issues faced while simulating the forward wave gait in gazebo. The first issue was tuning the PID parameters. The High PID values caused too much torque which resulted in robot flipping over. After tuning the PID values, we found these set of values to work: P =, 1 I = 0, D = 0. The second problem was the time gap between the commanded values. If the time gap is too low, it will result in large velocity and hence large torque. If it is too low, the robot will just fall in the transfer phase. For the simulation we choose 0.01 second as the wait time between each commanded joint values, which resulted in the desired behavior. The video of the robot walking in gazebo is attached in the presentation.

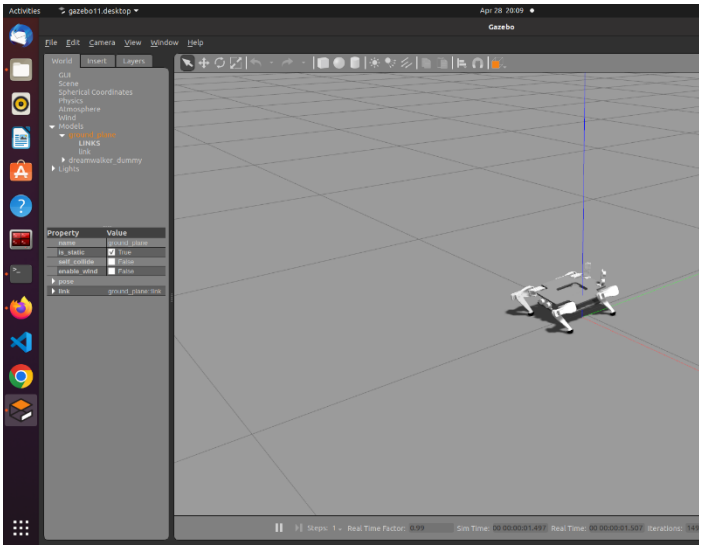


Figure 17: The Dream Walker Robot spawned in gazebo

IV. CONCLUSION

In this study the results of a walking gait was successfully developed using MATLAB and Gazebo and implemented into a commercially available quadruped robot. The platform of the Petoi robot allows for development of various controls architectures to be developed in any platform, and then outputted as a set off executable angles for each joint in a basic controlled such as the modified Arduino board used by the robot. We faced various challenges implementing the walking gait on robots in both simulation and on the Petoi robot dog. For instance, we needed to tune the PID controller to ensure that the torque to control joint angle is not high as increased values of joint motor torques would result in the flipping of the robot in gazebo. We also faced hardware issues relating to the servo motors of the front legs of the Petoi robot. Future work would involve further tunning these parameters as well as incorporating the gyroscope in the robot to perform additional stability control under different gaits.

V. REFERENCES

[1] Biswal, Priyaranjan, and Prases K. Mohanty. "Development of quadruped walking robots: A review." *Ain Shams Engineering Journal* 12, no. 2 (2021): 2017-2031. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[2] Sen, Muhammed Arif, Veli Bakircioglu, and Mete Kalyoncu. "Inverse kinematic analysis of a quadruped robot." *International journal of scientific & technology research* 6, no. 9 (2017): 285-289.Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[3] Bhatti, Z. "Gait analysis and biomechanics of quadruped motion for procedural animation and robotic simulation." *Bahria University Journal of Information & Communication Technologies (BUJICT)* 10, no. 2 (2017)

[4] Yoneda, Kan, Kenji Suzuki, Yutaka Kanayama, Hidetoshi Takahashi, and Junichi Akizono. "Gait and foot trajectory planning for versatile motions of six-legged robot." *Journal of Robotic Systems* 14, no. 2 (1997): 121-133.

[5] Petoi robot model by Aparatum (https://grabcad.com/library/petoi-bittle-evo-quadruped-robot-rev-engineering-1)

[6] Y M.J. Nategh a,M.M.Agheli " A total solution to kinematic calibration of hexapod machine tools with a minimum number of measurement configurations and superior accuracies " International Journal of Machine Tools & Manufacture 49(2009)1155–1164

[7] URDF Solidworks Plug In (http://wiki.ros.org/sw_urdf_exporter)

[8] https://github.com/Daemiac/Dreamwalker

[9] Alexander Dettmann, Daniel Kühn, Frank Kirchner "Control of Active Multi-Point-Contact Feet for Quadrupedal Locomotion" International Journal of Mechanical Engineering and Robotics Research Vol. 9, No. 4, April 2020

[10] Alexander Shkolnik and Russ Tedrake "Inverse Kinematics for a Point-Foot Quadruped Robot with Dynamic Redundancy Resolution" Proceedings 2007 IEEE International Conference on Robotics and Automation, Rome, Italy, 2007, pp. 4331-4336, doi: 10.1109/ROBOT.2007.364146.

[11] Felix Grimminger, Avadesh Meduri, Majid Khadiv, Julian Viereck, Manuel Wüthrich, Maximilien Naveau, Vincent Berenz, Steve Heim, Felix Widmaier, Thomas Flayols, Jonathan Fiene, Alexander Badri-Spröwitz, Ludovic Righetti "An Open Torque-Controlled Modular Robot Architecture for Legged Locomotion Research" arXiv:1910.00093 [cs.RO], https://doi.org/10.48550/arXiv.1910.00093