Modeling and Dynamic Control for a Hexapod Robot

Dr. Ferat Sahin
Multi Agent Bio-Robotics Lab, Electrical Eng. Dept, Rochester Institute of Technology
Rochester, USA
feseee@rit.edu

Brian Stevenson
Electrical Eng. Dept, Rochester Institute of Technology
Rochester, USA
bds2625@rit.edu

***Abstract*—In the every expanding field of robotics, mobile robots come in a two primary variations: wheeled and legged. This paper will focus on the latter, specifically a hexapod with a circular body that has its six legs distributed axisymmetrically around the body. The core goal of the project discussed here is to allow the robot to dynamically change foot placement based on an input vector in addition to controlling the position and orientation of the body. Specific attention is also given to a simulation model that can mimic this control of the robot in a virtual environment. Having a high level of control enhances the robot's ability to move in a very smooth and purposeful manner.**

Keywords—modeling; dynamic control; hexapod; kinematics; Webots™

I. Introduction

This paper will present the methodology behind the development and dynamic control of the hexapod, nicknamed TigerBug. The research presented in this paper investigates how to generate a locomotive solution to a hexapod that is enhances the fluidity of the robots' movements. To test this solution, a virtual model of this robot is created in software that is simulated in a close to real-world environment.

Inverse kinematics and differential motion will be used to establish the basis for a walking method that allows TigerBug to change its leg direction mid-step. This motion is the breakdown of a larger complex motion into smaller, more intuitive steps, much like the walking process mentioned above. Additionally, this motion should be unrelated to the position of the other legs, and also move each of its legs independently and based on instantaneous vectors of motion. Furthermore it will be able to rotate its body with respect to the central x, y, and z axes, creating the roll, pitch, and yaw motions. The allowance of this much freedom will enhance the ability of a walking gait to steer in the exact direction that is given, but be able to change its direction instantaneously.

A model of TigerBug is created in Cyberbotics powerful Webots™ robotic simulation software. This is a development environment than can model, program, and simulate custom robots. One of the greatest benefits of this package is its ability to test the robots in physically realistic worlds using the editable physics, sensors, motors, and shapes of each robot component. The model will mimic the physical TigerBug in all of its movements.

This project was sponsored by the Rochester Institute of Technology Multi-Agent Bio-Robotics Lab (RIT MABL)

II. Backround

While building robots and hexapods is a thrilling undertaking, it can be costly, time consuming, and impractical. This is where modeling comes in. One of the primary advantages is that a model is usually built on a computer with far greater processing power than they actual model. This allows the model to generate much more data feedback that can provide useful information about the behavior or the future states of the system. In the case of the hexapod, the kinematic equations, joint characteristics (positions, velocities, torques, etc.), and sensor feedback all require adequate processing power to make sure the algorithm computes all variables and states correctly. As you will see later in this paper, the calculations and data for the hexapod movement will get lengthy quite quickly. Add in the output of the sensors into different algorithms, and the complexity increases even further. Modeling additionally enables a user to experiment with different or more complex algorithms without fear of destruction.

The final test of a computer generated model robot is to see if using the same algorithm both the model robot and the physical robot. If they exhibit the same behavior then that model can be justified as an accurate depiction of the robot. By getting to this point, the model builder learns much more about the system and environment than before. The subtleties of the system come to light in a model much more than a physical implementation. If a constraint on a robot is missing in the model and not in the physical robot, that difference will be seen and the model can be corrected.

A further benefit that comes from modeling robots, especially mobile robots, is the potential development and use of swarm intelligence. By being able to create and control one hexapod robot in simulation, it can be extended to control many of them much easier compared to the hardware equivalent. The uses of swarm intelligence are powerful and its implementations vast. However, in respect to this paper, it is a concept yet to be exploited, but foreshadows possible gains.

III. Literature Review

A. Hexapods

Hexapods are one of the most popular legged robots. Its popularity stems largely from investigations into biologically inspired locomotion. They are widely investigated when legged robots can contribute substantially more to a locomotion solution due to their ability to move quickly without

978-1-4799-7611-9/15/$31.00 ©2015 IEEE

compromising their stability [1, 2]. Rough terrain is prime example of where a wheeled or tank-treaded robot would fall short of an optimal stable solution of motion. Legged robots have a greater ability to overcome obstacles and climb over barriers much easier than their wheeled counterparts. Whereas a wheeled robot's axle is the upper limit for vertical obstacles, a limbed/legged robot can climb vertical obstacles higher than itself [3, 4].

In some of the most extreme environments, it would be reasonable to assume some damage may occur to the robots. However, it has been shown that leg faults or even leg loss can be managed by changing the walking mechanism [5]. This results in a slight loss of control, but the important take away is that it's adaptable. The same cannot be said for wheeled robots in the same scenario.

There are also a large number of walking schemes available for a hexapod to implement. With at least three points of contact with the ground, the robot's center of mass stays consistently controllable and therefore stable [6]. This is especially true with hexapods with circular bodies and their legs equally spaced radii, like the one discussed in this paper. Other research has demonstrated that these robots utilize the optimal number of legs for control and that a larger number of legs will not increase walking speed [7]. Hexapods at the apex of the optimal speed vs. stability trade off. However, in order to get non-gaited leg motions, existing techniques for motion planning can be computationally expensive.

B. Dynamic Movements

Most current methods of walking are rudimentary and simplistic. They involve a set movement of picking up the joints, rotating the joints, and placing them down again. The same can be used to control a hexapod's body. While the elementary controls of walking and body movement can occasionally meet the requirements of a situation, more refined algorithms are often needed. Creating a walking scheme with dynamic movement is a problem that has been solved with a number of methods.

1) Dynamic Movements

A very basic way to move the legs was to use forward kinematics based on a position vector that is input into the system [8]. The robot moves according the specified path and determined by the vector. This method is typical of many current hexapods and has many implementations in off-the-shelf commercial products.

One design used a z-stepping technique, similar to the one in this paper, in order to move the leg from one point to another. Using inverse kinematics and the final position, the leg was moved towards its goal. The quadratic error from the final position was calculated after each movement, fed back in order to make its next movement, to reduce further error [9]. Another method used a trajectory planning to accomplish a walking movement [10]. A starting point, mid-point, and end point were defined and the robot calculates the best way to reach each of these points in succession using inverse kinematics. The robot used force and torque feedback to determine the loads on the joints to calculate the necessary movements to accomplish the specific movements in its trajectory. A newer method of synthesizing robot motion that has been a gaining distinction implements obstacle avoidance to place feet [11]. Obstacle avoidance is just one example, and the constant change and activity in the surrounding environment means that the robot should be able to act as such to adapt.

2) Artificial Intelligence and Machine Learning

There are very powerful machine learning techniques that are also used for dynamic movement solutions in robots. Of the many methods, this paper discusses how to apply fuzzy logic controllers (FLCs) to quadruped walking robots in order to learn and execute soccer-playing behaviors [12]. Part of this scheme is learning how to walk and move effectively. The approach discusses a hybrid architecture that combines reactive behaviors with deliberative reasoning.

In this instance, Particle Swarm Optimization (PSO) is used to identify the best structure of the robot limb configuration before the manufacturing process [13]. Using a simulation of possible performance, it optimizes the goal of walking the longest stride per time unit, with minimum energy input.

Many times, more than one of these techniques is implemented together. This paper proposes a new learning approach for evolving dynamic gaits of a hexapod using fully connected recurrent neural networks (FCRNNs). Moreover, the FCRNN parameters are found using a symbiotic species-based particle swarm optimization (SSPSO) algorithm [14]. The designed controller is then successfully applied to achieve a forward gait. This generates a somewhat slower speed of motion but is no less of a stable solution.

3) Force Sensing

The challenge of using force sensing for hexapod movement over uneven surfaces has been studied as a method of foot placement as well. This method of control increases the adaptability of the robot vehicle to irregular terrain and to distribute the foot forces during locomotion over both hard and soft ground [15].

One solution for hexapod motion over uneven terrain uses the combination of designed force threshold-based foot motion and center-of-mass (CoM) omnidirectional movement and a process called Environment Trailed Trajectory (ETT). The culmination of these methods allows the robot to intelligently solve for stable foot placement solutions when presented with uneven terrain [16]. Another paper talks about many of the ways that force sensing can be performed in hexapod locomotion as compared to real six-legged insects [18]. It was found from this paper that force sensing can be used to detect force and the rate of change in force, or jerk, during locomotion. This method can provide information about the direction of force while standing still, detect sudden force decreases such as slipping or impacts, and enhance muscle feedback while moving [17]. The overall algorithm development gives a much more biological viewpoint of hexapod movements. One force sensing technique was to use current measurement through shunt resistors. The voltage drop across the resistor is amplified, passed through an ADC, and finally processed by a digital control loop. The other solutions discussed in the paper include induction measurement for current estimation and the use of magnetic field sensors [18].

4) Modeling Robots

Illustrating control with models has also been used and with great success in many studies on hexapods. Using models and techniques to make the model more like the real robot it personifies can show great improvements to algorithms.

Complexity in the calculations of the gait and the control variables is large enough that if not managed and modeled properly, some solutions cannot be found. This paper uses SolidWorks as well as MSC ADAMS to simulate and analyze the prototype model of the robot [19]. This model also helps find a methodology to get the robot inverse solution in ADAMS, and simplify the theoretical calculation. In the end, the simplification further improves the efficiency of the design. Another modeling method uses imagined mechanical components to create virtual forces, which are applied through the joint torques of real actuators [20]. The proposed method of controlling joint torques for movement facilitates the use of a high level control system which can be used above the low level virtual model controllers to modulate the parameters of mechanical mechanisms quite successfully. Similar to the research seen in [13], the paper in [21] works out a way to optimize a robot's feet spacing using the numerical analysis method. This technique calculates the robot's degree of agility utilizing virtual prototyping technique and completely measures the flexibility of both the robot's leg and body.

IV. Description of System

TigerBug was original built to be a bigger and badder model of another hexapod with the same radial configuration, at the RIT Multi-Agent BioRobotics Lab (MABL). The fundamental goal of the new design is to create a new robot with a more advanced leg placement technique.

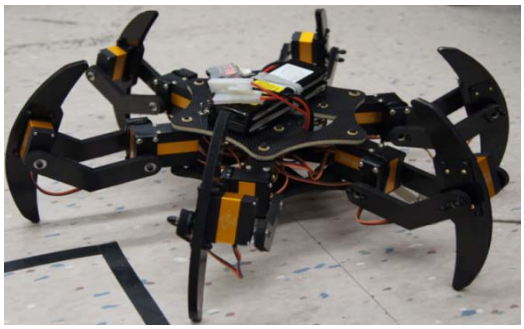

Fig. 1. The physicalTigerBug

Using the properties established in the construction of TigerBug, the model is generated. Everything that can be quantified is loaded into the model including dimensions, servo specifications, weights, and coefficients of friction [22, 23]. Additional aspects are included in the model such as sensors and data collection that does not yet exist on the physical TigerBug. An accelerometer, compass, gyroscope, GPS, and Inertial Mass Unit (IMU) were all added for future data processing and algorithm development within the model. The final model is shown in Figure 2 here. The model is used for most of the algorithm testing and development. The simulation is also setup to be able to run the model and the physical robot simultaneously. This is a huge benefit it will prove the models ability to mimic the behavior exactly.

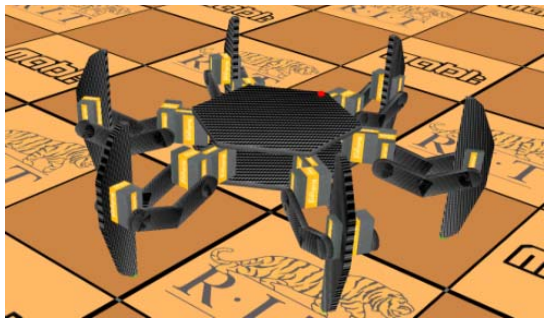

Fig. 2. Final Model of TigerBug in Webots simulation software, in its home position. Red dot indicates Leg 1

V. Proposed Approach

The basic approach allows a user to input commands for walking and body manipulation using an Xbox 360 controller. By reading in these commands as very small but precise movements, the robot will move much more seamlessly than if under pre-defined motions. This method will utilize a quadruped gait sequence, meaning that two of the legs will be off the ground at one time during the walking algorithm.

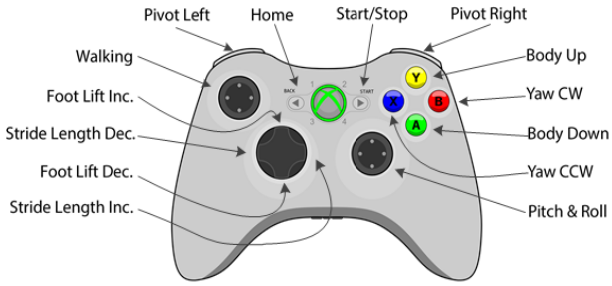

Fig. 3. Xbox 360 commands used for TigerBug

A. Kinematics

In this approach, inverse kinematics and the robot Jacobian matrix are used to determine the differential rotations of the servos from the differences in Cartesian space. That is to say there are small differential changes in the x, y, z plane as well as the differential changes in the roll, pitch, and yaw movements. These movements from the controller are mapped to the differential rotations of each servo in the robot. Mapping of these motions starts by deriving the frame transformations and the Denavit-Hartenberg (D-H) Table. This process will be derived such that it applies to all legs given a leg number.

It is important to note that the orientation of the z-axis of the frame at each foot must be parallel to the frame at the center of the robot in order for the team's control method to work. For this reason, the D-H Table has more transformations than joints due to the readjustment of the foot frame back to match the center frame orientation. This is absolutely critical to make sure that each foot moves in the correct direction with respect to the body at each and every instance in time. The D-H Table is derived in Table 1. The first row in the table is a preliminary joint to move the center frame to frame of the leg. The transformation from the center, C, to the foot frame, F, is the multiplication of all of the frame transformations of the joints.

TABLE I. THE COMPLETE D-H TABLE FOR TIGERBUG

Frame/ Joint	D-H Parameters			
	θ	d	a	α
${}^{C}T_{L}$	$\Theta_L = (Leg - 1)*60°$ [a]	0	L_1	0
${}^{L}T_{H}$	0	$-L_2$	0	0
${}^{H}T_{K}$	Θ_1	0	L_3	90°
${}^{K}T_{A}$	Θ_2	0	L_4	0
${}^{A}T_{F1}$	$\Theta_3 - 90°$	0	L_5	0
${}^{F1}T_{F2}$	-90°	0	L_6	0
${}^{F2}T_{F3}$	$180° - \Theta_2 - \Theta_3$	0	0	-90°
${}^{F3}T_{F}$	$\Theta_1 - \Theta_L$	0	0	0

[a] center frame to a leg via a multiple of 60 degrees. L=Leg, H=Hip, K=Knee, A=Ankle, F=subframe

The D-H Table then allowed the team to calculate the Jacobian matrix, *J*, needed for applying the inverse kinematics and differential motion. This matrix is calculated by taking the derivative of the corresponding kinematic equation with respect to one of its variables [24].

Knowing the Jacobian, the following equations can now be setup the equations that relate the differential rotations of the servos from the differences in Cartesian space. To find the differential change in the motor angles, D_θ, the inverse of the Jacobian is pre-multiplied by the differential in movement in Cartesian space, *D*, as seen in Equation 1. The vector D_θ is the angular change that will be made in the current step. Finally, the differential change in servo position is added to the current angle, using Equation 2. The robots, both the realized and model, are then moved to the new current angle. This happens for each leg of the robot for each iteration.

$$D_\theta = J^{-1} * D, \ \ where\ D_\theta = \begin{bmatrix} d\theta_1 \\ d\theta_2 \\ d\theta_3 \end{bmatrix} \ and\ D = \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} \quad (1)$$

$$\theta_{CURRENT} = \theta_{CURRENT} + D_\theta\ , where\ \theta_{CURRENT} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} \quad (2)$$

B. Algorithm Structure

The defining equations so far have set up the leg work for making the robot make small differential movements into small joint angle adjustments. However, the differential movement vector, *D*, must take into account both the walking difference of the foot as well as the body manipulation. To formulate this vector properly, it must use the walking vector direction, foot height, the *n*, *o*, and *a*, (roll, pitch, and yaw motion from the controller), as well as the lateral movement x, y, z. Using the kinematic equations and these vector inputs, the *D* vector can be fully defined as seen in Equation 3.

$$D = \begin{bmatrix} D_0 \\ D_1 \\ D_2 \end{bmatrix} = \begin{bmatrix} dy \\ dx \\ \Delta_F + dz \end{bmatrix} + {}^{T}\Delta_{\boldsymbol{F}}$$

$$where \quad {}^{T}\Delta_{\boldsymbol{F}} = {}^{C}\boldsymbol{T}_{\boldsymbol{F}}^{-1} * \mathrm{c}_\Delta * {}^{C}\boldsymbol{T}_{\boldsymbol{F}} \quad and \quad (3)$$

$$\mathrm{c}_\Delta = \begin{bmatrix} 0 & -\delta z & \delta y & dx \\ \delta z & 0 & -\delta x & dy \\ -\delta y & \delta x & 0 & dz \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

The main loop of the algorithm constantly reads the inputs of the Xbox controller. It then finds the differential change from those commands, tells each of the legs what the controls are, calculates the differential rotations, and updates the servo angles with those differences.

1) Body Manipulation

First, we consider the kinematics without walking. All of the buttons on the right side of controller supply the variables that control the body movements like the height and rotation of the center body. These are the *dz* and a_z, respectively in the equations. These control values are small but allows for smooth movements in those directions. Similarly, the n_x and o_y values are very small and controlled by the analog joystick. A linear equation to fit the joystick data to rotational change in the x and y axis is required. The linear trend is applied in both the positive and negative direction. To represent the deadzone feedback of the stick, a threshold is applied. All motion is now deliberate and purposeful and not a result of drift error.

2) Walking

The movement of the legs is an iterative procedure that continuously updates with the current state of the controller. At each update, the differential change is applied to the robot, derived from the above commands. As long as a valid walking command is given, each leg goes through a finite number of differential movements, k_{MAX}, using a sub-stepping variable, *k*, for one gait cycle. It should be noted that the robot can also still take in any other command during walking, which is crucial for modeling realistic motion.

When the variable reaches its maximum value, the legs will have travelled its full path and the next legs in the gait sequence starts moving and so-on. While the x and y of the leg positions are determined by the input vector, the z position of the foot is defined with a function that is based on the current sub-step value. The change in the z-position of the leg is fed directly into the differential motion equations and carried through with the input vector from the joystick.

Walking has four different components that need to be addressed by this approach. They are direction, stride length, foot lift height, and sub-step size. The actual walking is done by applying the lift to a pair of legs while pushing with the other four. The values of *dx*, *dy*, and Δ_F are the variables that change during the process of walking. It is a simple addition to the differential Cartesian vector seen in Equation 3. Direction is solved using the same trend fitting line process as the rotation joystick change above. It uses a slightly different dead zone, but is also scalable. The maximum step size in one direction shouldn't be too large otherwise the Jacobian inverse and differential assumption of the kinematics begin to fail.

$$\Delta_F = lift_{mag} * \frac{\left(0.5 - \frac{k}{k_{MAX}}\right)}{1000}\ , where\ k \stackrel{\text{def}}{=} 0, 1, \ldots, k_{MAX} \quad (4)$$

The most critical part of the method is application of the change in foot height, Δ_F. This change in foot height equation is seen in Equation 4. The plots in Figure 4 demonstrate the

Δ_F values versus iteration. Also shown is the true foot path based on the change in foot height equation.

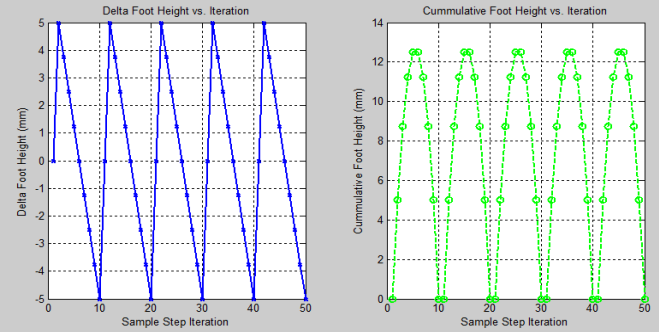

Fig. 4. Delta foot height for maximum step of 10 and the cumulative foot height to show how the foot will react give these delta values. (Using this lift magnitude of 10, maximum height the foot will be off the ground is 1.25 cm.)

By increasing the maximum step value, the , Δ_F steps become smaller and even smoother. Increasing it too much can result in slow gait. In order to walk in the desired direction, the lifted pair of legs is moved along the direction vector. At the same time, the legs that are still on the ground move in the opposite direction, pushing the robot along the desired vector. We will call the desired direction values from the controller V_x and V_y and use them to then determine the differential change of that foot. Table 2 shows how the change in X-Y Cartesian plane is found from the lateral command from the controller.

TABLE II. CALCULATED DIFFERENTIAL CHANGE IN X AND Y DIRECTIONS OF THE LIFT LEGS AND PUSH LEGS

Lift Legs		Push Legs	
$dx = \frac{V_x}{k_{MAX}}$	$dy = \frac{V_y}{k_{MAX}}$	$dx = \frac{-V_x}{2 * k_{MAX}}$	$dy = \frac{-V_y}{2 * k_{MAX}}$

The push legs need to have the factor of two in the denominator due to the quadruped gait. These legs are pushing for 2 cycles before they are lifted. Once the lift pair touches down again, they become pushers while another set of legs becomes lifted. This process continues while the user is applying an input to the walking joystick.

VI. RESULTS

After its development, the proposed approach was tested on the simulation robot and the physical robot simultaneously. This not only demonstrated the method but also verified that the model built in the Webots simulation software was as created as close as possible to the real thing. This was judged by close visual inspection of the robots after different commands. Any action generated by the controller caused an identical reaction in each of the robots, proving the model is successful. The different body manipulations of the robot can be seen in the next few figures.

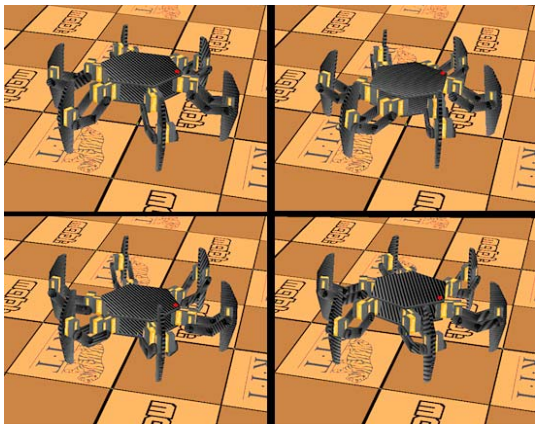

Fig. 5. Body Manipulation in the z-axis; positions are body up, rotation counterclockwise, rotation clockwise, body down (clockwise from top left)

The motions seen in both Figure 5 and Figure 6 were arrived at very smoothly, just as designed. All of the motions control just the body of the robot. The kinematic equations that map the robot can be proved accurate by assuring that all feet are also still on the ground. If this were not the case, the robot would become unstable and show the model and/or equations were incorrect. All of these motions were created using the right half of the controller seen in Figure 3.

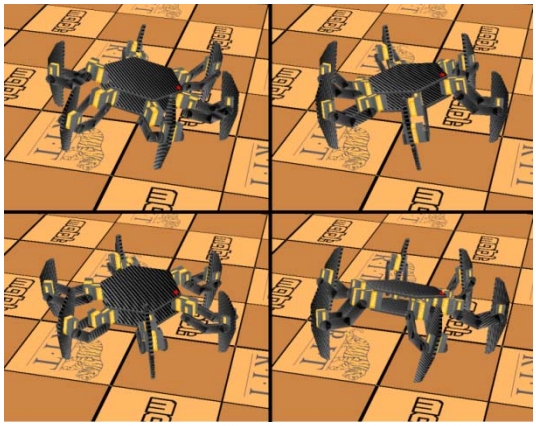

Fig. 6. Body rotations in the x and y direction; positions are tilt forward, tilt backward, tilt left, tit right (clockwise from top left)

Even more important is that any and all combinations of the body movement can occur at the same time. Because of the differential movement of the proposed approach, each movement can be executed seemingly simultaneously, one small step at a time in any configuration. An example of this can be seen in Figure 7.

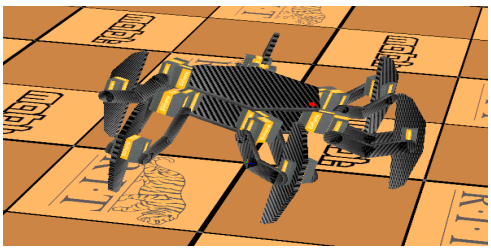

Fig. 7. Combination of the body manipulations; up, clockwise, and tilt forward

The heart of the proposed approach however, is the ability to establish a smooth walking algorithm. In both the physical and simulation robots, the smooth balanced walking was observed. The body presented only with very small jerks on occasion. Similarly, the walking algorithm can occur with almost any pre-determined body position as well as a changing body position. The only limitation for walking occurs when the body manipulation movement has already caused the servos to reach their physical limits.

The added ability to change variables like the foot height maximum, and stride length were also demonstrated on both robots. The ability to increase stride length showed that speed could be increased, at a slight loss of smooth and stable movement. The foot height change showed that the robot could function equally as well with different walking foot heights. After experimenting with different combinations of foot heights and stride lengths, it was found that the most balanced and smooth approach came when the stride length was around 4 cm, and the foot height was 1.75 cm off the ground.

VII. Conclusion

The hexapod robot proposed control scheme was implemented very successfully on the TigerBug hexapod robot and the Webots TigerBug model. It allowed the robot to walk, move and orient itself correctly in almost all situations and positions. The robot was able to walk with very smooth motions and maintained its stability. It was also able to perform rotations around all three of its central axes. Its ability to switch the direction of the body mid-step was also demonstrated with success. It was evident that the instantaneous changes in direction allowed the robot move freely and dynamically, unlike other hexapod robot walking methods. This novel method of movement was versatile and lends itself well to future work. Overall, the hexapod robot method of walking was extremely effective.

Acknowledgment

This could not have been done without the team work and contributions of Andy Anthony, Sander Idelson, and Alexander Yevstifeev. Additional thanks for their assistance is given to Ryan Bowen and Shitij Kumar.

References

[1] Espenschied, K.; Quinn, R.D.; Chiel, H.J.; Beer, R.D., "Biologically-Inspired Hexapod Robot Project: Second Robot," *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on* , vol.3, no., pp.V11,, 21-27 May 1995

[2] Jin Bo; Chen Cheng; Li Wei; Li Xiangyun, "Design and Configuration of a Hexapod Walking Robot," *Measuring Technology and Mechatronics Automation (ICMTMA), 2011 Third International Conference on* , vol.1, no., pp.863,866, 6-7 Jan. 2011

[3] Ignell, Nils Brynedal, Niclas Rasmusson, and Johan Matsson. "An overview of legged and wheeled robotic locomotion."

[4] Freedberg Jr. , S, 2012, 'Why The Military Wants Robots With Legs (Not To Run Faster Than Usain Bolt)' Breaking Defense, September 7. <http://breakingdefense.com/2012/09/why-the-military-wants-robots-with-legs-robot-runs-faster-than/>

[5] Ding, X., Rovetta, A., Wang, Z. and Zhu, J. M. 2010. Locomotion analysis of hexapod robot. In: *Climbing and Walking Robots* (ed: B Miripour) pp 291 - 310. InTech, Italy

[6] Spirito, Carl P., and Daniel L. Mushrush. "Interlimb coordination during slow walking in the cockroach: I. Effects of substrate alterations." *The Journal of Experimental Biology* 78.1 (1979): 233-243.

[7] Preumont, A.; Alexandre, P.; Ghuys, D., "Gait analysis and implementation of a six leg walking machine," *Advanced Robotics, 1991. 'Robots in Unstructured Environments', 91 ICAR., Fifth International Conference on* , vol., no., pp.941,945 vol.2, 19-22 June 1991

[8] Graca, P.; Zimon, J., "Mechanical construction and kinematic calculations of the six-legged walking machine ARTHRON," *Electrodynamic and Mechatronics, 2009. SCE 11 '09. 2nd International Students Conference on* , vol., no., pp.23,24, 19-21 May 2009

[9] Seljanko, F., "Towards omnidirectional locomotion strategy for hexapod walking robot," *Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on* , vol., no., pp.143,148, 1-5 Nov. 2011

[10] Sorin, M.-O.; Mircea, N., "The modeling of the hexapod mobile robot leg and associated interpolated movements while stepping," *System Theory, Control and Computing (ICSTCC), 2012 16th International Conference on* , vol., no., pp.1,5, 12-14 Oct. 2012

[11] Seyed Mohammad Khansari-Zadeh and Aude Billard. A dynamical system approach to realtime obstacle avoidance. Autonomous Robots, 32(4):433–454, March 2012.

[12] Dongbing Gu; Huosheng Hu, "Integration of Coordination Architecture and Behavior Fuzzy Learning in Quadruped Walking Robots," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* , vol.37, no.4, pp.670,681, July 2007

[13] Burkus, E.; Odry, P., "Mechanical and walking optimization of a hexapod robot using PSO," *Computational Cybernetics (ICCC), 2013 IEEE 9th International Conference on* , vol., no., pp.177,180, 8-10 July 2013

[14] Chia-Feng Juang; Yu-Cheng Chang; Che-Meng Hsiao, "Evolving Gaits of a Hexapod Robot by Recurrent Neural Networks With Symbiotic Species-Based Particle Swarm Optimization," *Industrial Electronics, IEEE Transactions on* , vol.58, no.7, pp.3110,3119, July 2011

[15] Schmucker, U., et al. "Force sensing for walking robots." *Proceedings of the Third International Symposium on Adaptive Motion in Animals and Machines, AMAM*. 2005.

[16] Irawan, A.; Nonami, K., "Force Threshold-Based Omni-directional Movement for Hexapod Robot Walking on Uneven Terrain," Computational Intelligence, Modelling and Simulation (CIMSiM), 2012 Fourth International Conference on , vol., no., pp.127,132, 25-27 Sept. 2012.

[17] Sathya Kaliyamoorthy, Roger D. Quinn, and Sasha N. Zill. "Force Sensors in Hexapod Locomotion." The International Journal of Robotics Research July 2005 24: 563-574

[18] Ziegler, S.; Woodward, R.C.; Iu, H.H.-C.; Borle, L.J., "Current Sensing Techniques: A Review," Sensors Journal, IEEE , vol.9, no.4, pp.354,376, April 2009

[19] Luo Qingsheng; Zhang Hui; Han Baoling; Zaho Xiaochuan, "Research on biologically inspired hexapod robot's gait and path planning," *Information and Automation, 2009. ICIA '09. International Conference on* , vol., no., pp.1546,1550, 22-24 June 2009

[20] Torres, Ann L. *Virtual model control of a hexapod walking robot*. Diss. Massachusetts Institute of Technology, 1996.

[21] Baoling Han; Qingsheng Luo; Qiuli Wang; Xiaochuan Zhao, "A Research on Hexapod Walking Bio-robot's Working Space and Flexibility," *Robotics and Biomimetics, 2006. ROBIO '06. IEEE International Conference on* , vol., no., pp.813,817, 17-20 Dec. 2006

[22] Webots 7.4.0 [computer program]. Lausanne, Switzerland: Cyberbotics S.à r.l.; 2014.

[23] Cyberbotics. *Webots User Guide.* Lausanne, Switzerland: Cyberbotics S.à r.l.; 2014.

[24] Cyberbotics. *Webots Reference Manual.* Lausanne, Switzerland: Cyberbotics S.à r.l.; 2014.

[25] Niku, Saeed B.. "Differential Motions and Velocities - 3.8 Calculating the Jacobian." *An introduction to robotics analysis, systems, applications*. Upper Saddle River, N.J.: Prentice Hall, 2001. Print.