Embedded Walking Algorithm for Biped Humanoid Robot with 17 Degrees-of-Freedom

Sherif Sherif¹, Jordan Kralev¹, Tsonyo Slavov¹ and Veselin Kunchev¹
¹Technical University of Sofia, Faculty of Automatics, Department of Systems and Control, Sofia, Bulgaria
E-mail: mr.shark94@abv.bg; jkralev@yahoo.com; ts_slavov@tu-sofia.bg; v.kanchev1993@gmail.com

Abstract. The realization of the control algorithm consists of two main stages: planning of a joint coordination sequence and implementation of an embedded walking algorithm in the Arduino Mega 2560 hardware platform. The joint coordination sequence consists of 13 steps in order for the humanoid robot to move forward without falling. The embedded walking algorithm is realized around the operating points of the servo motors. The software is developed in Arduino IDE in C language.

1. Introduction

Biped walking machines have been developed since 1970. Today popular biped robots are ASIMO of HONDA [1], WABIAN-2 of Waseda University [2], and HRP-3 of AIST [3]. The walking algorithm of a biped robot can be either static [4] or dynamic [5]. In static walking the robot stops its walking motion at each step in order to dampen the oscillations from the previous step. Dynamic walking is smoother because each parameter of the walking cycle is continuously varied such that centre of gravity (CG) translation speed is close to constant. The realization of a stable biped robot walking is a considerable challenge even in deterministic environment [6]. The key to successful walking requires the projection of CG onto the ground to lie within the supporting polygon of the two feet. In the case of dynamic walking the same condition is required for the zero moment point.

Considered system is laboratory robot with 17 degrees-of-freedom (DOFs) provided by servo motors LD-2015 mounted at the key joints. The microcontroller which controls the walking is ATmega2560 which is programmed with the Arduino IDE.

Figure 1. Laboratory model of biped robot with 17 degrees-of-freedom with 10 inputs and 6 outputs
The aim of the open-loop control for biped walking robots is to attenuate external disturbances acting on the mechanical system. In order to achieve such control the external disturbances have to be from deterministic origin. For the present laboratory model of humanoid robot the key external disturbances are: gravitational force, friction forces, inertial forces and CG motion.

The control actions are $\theta_{LL}$ and $\theta_{LR}$ are the angular offset in the left and right key joints, with $i = 1 \ldots 5$. The output signals $x, y, z$ represent the position for the robot CG and Euler angles $\varphi, \theta, \psi$ represent the robot orientation.

2. External disturbances

The external disturbance forces acting on the mechanical construction of the robot during walking can be pointed:

- **Gravitational force - $d_{GF}$**
  The gravitation is disturbing the stability on the robot when the incidence angle between the body and the ground surface is increased. If that angle goes beyond certain limit the robot cannot be stabilized with the control action from the servo motors alone.

- **Friction - $d_{FF}$**
  Successful walking requires a certain amount of contact force between the ground surface and robot feet which prevents foot slipping. However when a foot is translated over the surface developed friction force can disturb the orientation of the robot and consequently its trajectory.

- **Inertial forces - $d_{IF}$**
  These forces are fictitious forces representing the effect of kinematic variables upon the orientation of the body. If the frequency of the step cycle is increased beyond certain limit the abundant energy from uncompensated inertial forces cannot be dampen fast enough and leads to instability.

- **CG motion - $d_{DCG}$**
  Generally if the center of gravity is not projected upon the feet base then the mechanical construction becomes less stable and small uncompensated disturbance force would cause the robot to fall down.

The location of the robot CG is the most important factor which prevents the robot from falling during the execution of the walking step cycle. Location of the CG can be expressed with (1) by accounting for the mass action of the various components of the robot.

$$
\ddot{x}_{CG} = \frac{m_B}{M} \ddot{x}_{CG,B} + \frac{m_{HL}}{M} \ddot{x}_{CG,HL} + \frac{m_{HR}}{M} \ddot{x}_{CG,HR} + \frac{m_{LL}}{M} \ddot{x}_{CG,LL} + \frac{m_{LR}}{M} \ddot{x}_{CG,LR}
$$

(1)

where $\vec{x}_{CG}$ is the location vector of the CG with components

$$
\dot{x}_{CG} = \begin{pmatrix} x_{CG} \\ y_{CG} \\ z_{CG} \end{pmatrix}
$$

(2)

and $\vec{x}_{CG,i}$ is the location of CG of respective body element $i \in \{B, HL, HR, LL, LR\}$, $M$ is the total mass of the robot, $m_B$ is the mass of the trunk, $m_{HL}$ is the left hand mass, $m_{HR}$ is the right hand mass, $m_{LL}$ is the left leg mass and $m_{LR}$ is the right leg mass such that

$$
M = m_B + m_{HL} + m_{HR} + m_{LL} + m_{LR}
$$

(3)

The CG of each component of the robot at time instant $k$ is shifted with respect to CG location at time instant $k - 1$. 
\[ \vec{x}_{CG,i}(k) = \vec{x}_{CG,i}(k - 1) + \vec{\Delta}_i \]  

with \( \vec{\Delta}_i \) representing the CG displacement caused by joint reconfiguration during walking. Then the displacement of the robot CG \( \vec{\Delta}_{CG} \) will be a weighted sum over the displacements \( \vec{\Delta}_i \) of CGs of its components with weights \( m_i/M \). Hence by rotating a joint in particular direction the robot CG is translated in this direction. In order to keep the upper stance the projected location of the robot CG onto the ground surface should be kept within the supporting foot base.

3. Walking algorithm

At initial position the robot joint angles are set to their trim values. They are experimentally tuned to guarantee its upper standing position and are summarized in Table 1.

| Joint                  | Notation          | Value [deg] |
|------------------------|-------------------|-------------|
| Left ankle roll        | \( \theta_{1, L, \text{Trim}} \) | 97          |
| Left ankle pitch       | \( \theta_{2, L, \text{Trim}} \) | 74          |
| Left knee pitch        | \( \theta_{3, L, \text{Trim}} \) | 6           |
| Left hip pitch         | \( \theta_{4, L, \text{Trim}} \) | 85          |
| Left shoulder pitch    | \( \theta_{5, L, \text{Trim}} \) | 155         |
| Right ankle roll       | \( \theta_{1, R, \text{Trim}} \) | 79          |
| Right ankle pitch      | \( \theta_{2, R, \text{Trim}} \) | 82          |
| Right knee pitch       | \( \theta_{3, R, \text{Trim}} \) | 142         |
| Right hip pitch        | \( \theta_{4, R, \text{Trim}} \) | 70          |
| Right shoulder pitch   | \( \theta_{5, R, \text{Trim}} \) | 87          |

The joint coordination sequence consists of 13 steps in order for the humanoid to move forward without falling. The starting position of the walking robot is denoted as "A" when its center of gravity is distributed on both legs as in figure 1.

In stage "B" as figure 2 shows the center of gravity projection is shifted to the left leg by applying ankle roll and right shoulder pitch.
Figure 5. Stage “E”  
Figure 6. Stage “F”  
Figure 7. Stage “G”  
Figure 8. Stage “H”  
Figure 9. Stage “J”  
Figure 10. Stage “K”  
Figure 11. Stage “L”  
Figure 12. Stage “M”  
Figure 13. Stage “N”
In step "C" (figure 3.), the right leg is moved forward but the center of gravity projection remains on the left foot and keeps the stability of the robot. In this stage the right leg pitch is performed in ankle, knee and hip joints.

In step "D" given in figure 4., it can be noticed that the right leg is exerted a little further from the left, which is due to the changes in the previous stage (Stage "C" – figure 3.). Also in step "D" center of gravity projection is again distributed on both legs. To accomplish this step, three changes are made to the current values of the servo motors – roll of both ankles and pitch of the right ankle.

In the next step "E" (figure 5.), the center of gravity projection is shifted to the right foot by rotating four of the joints – ankle’s roll and shoulder’s pitch.

During the sixth stage "F" shown in figure 6, the left foot of the robot is moved forward by performing ankle, knee and hip pitch. The center of gravity projection then lies on a right leg which is static to ensure the stability of the robot. The difference between this stage compared to stage "C" is in supporting leg where the center of gravity projection is located.

In step "G" (figure 7.), the weight of the robot is distributed over the both legs, but it is not in the starting position because the right knee joint, the right shoulder joint and some of the left leg joints are still displaced. To achieve this step roll of both ankles is applied and also a pitch of the left shoulder.

The step “H” from figure 8, shifts the center of gravity projection to the left leg. The difference here with respect to stage “B” is that the two servos in the knee of the right leg are not in their initial position and also the servo in the shoulder joint of the right hand. The other difference is in the left leg and in particular in the servo motor for ankle pitch and the two servo motors in the knee, which in the current stage ("H") are not in their initial position.

In step "J" (figure 9.) the left foot of the humanoid begins its return to its starting position. To prevent the robot from falling this is done in two stages ("J" and "K"). The need for two steps is due to the fact that the current angle of the left knee joint is twice as large as that of the right knee. The difference in this stage with respect to "F" stage is the foot on which supports the center of gravity projection. In order to accomplish this step, three changes were made to the current values of the joints. The ankle and knee servos are still not returned to their starting position. Their current angels are set to an intermediate values with respect to stages "F" and "K".

In the tenth stage "K" (figure 10.) the process of returning of the ankle and knee joints of the left leg to initial position is completed. Also, at this stage, the right shoulder joint is set to its trim value.

In step "L" shown in figure 11, the center of gravity projection lies on both robot's legs, but it is still not in the starting position because the right foot is exerted slightly further than the left. The right leg is ahead of the left due to its ankle and knee joints. In order to realize this stage, there are changes in the current values of two servo motors, which are located in the feet joints.

In stage "M" (figure 12.) the center of gravity projection is shifted to the right leg. In order to accomplish this step, changes are made to the current values of three servo motors pointed in the figure.

In next step "N" (figure 13.) the robot's right foot returns to the starting position. For the completion of this stage right hip and knee joint are rotated. Also the left shoulder joint is rotated. After the rotations the servo motors in the knee of the right leg and the servo motor in the left shoulder are set to their trim values.

A sub-step before returning to stage “A” after the last stage "N" is executed in order to return all joints to their trim values. In this sub-step, the weight of the robot is distributed over both legs. Then the robot completes the joint coordination sequence and its location is one step ahead.

4. Implementation in Arduino
The robot control algorithm is embedded in Arduino Mega 2560 hardware platform which is attached at the back of the body element together with a signal distribution board (figure 14.). The software is developed in Arduino IDE in C language. Also created special shield, which plays a roll on the distribution board. The shield allowed easily installation of servo motors and the connection of components to the battery.
Figure 14. Location of the controller on the back of the robot’s body

The signal line of each servo motor is connected to a digital output pin of the Arduino platform. The trim values of the joints $\theta_{ij,trim}$ are declared as constants. And the current offset values $\theta_{ij}$ around the trim constants are calculated according to the presented step sequence from the previous section. The angular position of the joint is commanded to the servo motors as a pulse width modulated signal, with duration $u_{ij}$ of the pulse between 1 [ms] to 2 [ms] corresponding angles from 0 to 180 degrees.

The progress through the step sequence is monitored by a dedicated temporal progress variable $\tau \in [0, T_{step}]$, where $T_{step}$ determines the necessary time for performing a single step. The value of $T_{step}$ is tuned experimentally and it is set to 2760 [ms]. If a shorter period is used the humanoid robot becomes unstable. The current time is obtained through the processor internal clock $t$ and the progress variable $\tau$ is calculated as $\tau = t - T_{start}$, where $T_{start}$ is the variable for storing the initial timing of the current step. The progress variable $\tau$ can be represented in percentage scale for convenience so when 100 [%] are reached the step sequence is completed the start time $T_{start}$ is set to the current value of the clock $t$.

The programming of each stage of a single step form the walking sequence requires a particular configuration of the robot joints. For example let focus on the stage “B” from figure 2. This stage occurs when $0.83 \% \leq \tau \leq 5 \%$. After checking that condition the respective servo motors are rotated by setting a new value to $\theta_{ij}$. The pulse width is then calculated as (eq. (5))

$$ u_{ij}(t) = 0.001 \left( 1 + \frac{\theta_{ij,trim} + \theta_{ij}(t)}{180} \right) $$

Figure 15. and figure 16. shows the control signal send to the servo motor in order to achieve the desired joint angular position. It is a pulse width modulated signal with varying pulse duration between 1 to 2 [ms]. The walking algorithm requires all 17 joint servo motors to receive appropriate signal. According to the servo communication standard the period of the command signal should be $T_{pwm} = 20$ [ms]. Since the correctness of the servo commands depend critically on timing it have to be precisely synchronized with the rest of the tasks within the Arduino loop() function. The algorithm for synchronization is schematized in figure 17.

In the beginning of the loop() function all of the servo pins are set to high level by the digitalWrite() routine. Then the starting time of this ‘high’ event is stored in the variable start_time. Then while all servo command pins are help high the joint configuration is updated if required with a function Calculate_Servo_Commands(). The updated servo pulse width are stored in variables $u_{ij}$. They are consequently filled in the array_pulse[] array. Another array named array_pins[] will held
the corresponding to \( u_{ij} \) values pin numbers. These two array are then sorted in an ascending order of the pulse width by the function `Sort_Pins_By_Pulse_Width(pins, angles)`. Then the pin with the shortest pulse width is driven to low state first. The duration of the pause for the consequent pins is calculated as a temporal difference between the pulse widths with the function `First_Difference()`.

```
loop()
{
    digitalWrite(pin_1L,HIGH);
    ...
    digitalWrite(pin_5R,HIGH);
    start_time = Get_Time();
    Calculate_Servo_Commands();
    array_pins[0] = pin_1L; array_pulse[0] = u_1L;
    ...
    array_pins[9] = pin_5R; array_pulse[9] = u_1R;
    Sort_Pins_By_Pulse_Width(array_pins,array_pulse);
    array_pulse_d = First_Difference(array_pulse);
    while (Get_Time() - start_time < array_pulse_d[0]);
    digitalWrite(array_pins[0],LOW);
    ...
    while (Get_Time() - start_time < array_pulse_d[9]);
    digitalWrite(array_pins[9],LOW);
    while (Get_Time() - start_time < 20ms);
}
```

**Figure 17.** Servo command synchronization algorithm

On figure 18. and figure 19. the servo control actions from both sides of the robot are presented. From these figures it can be determined how many steps the humanoid robot is making forward and the distance that travels for 50 [s] can be estimated. Noting that one step ahead is 2.76 [s] in duration, and the one-step distance is 2.14 [cm] the distance covered for period of 50 [s] by the robot which performs 18 steps is approximately 39 [cm].
Figure 18. Joint angles in radians of the left side of the robot during the walking algorithm

Figure 19. Joint angles in radians of the right side of the robot during the walking algorithm

Figure 20. Block scheme of Embedded Walking Algorithm
On figure 20. is shown block scheme of part from embedded walking algorithm for biped humanoid robot. As shown, the program starts with initialization of the low level drivers of the servo motors, the input-output ports for pulse width modulated (PWM) signals, the absolute (Trim) angular values, the current values and the offset values over the operating point, the time period for one step $\tau$, and measured time for one step $T_{\text{start}}$ and the clock value $t$.

After initializing starts a procedure of comparing $\tau$ with the percentage of the time elapsed for one single step. If the value is bigger than 100 [%] it is necessary to assign zero values to $\tau$ and $t$. We check then if $\tau$ is between 0.83 [%] AND 5 [%]. If the compare return flag is TRUE, then the Left ankle roll, Right ankle roll and the Right shoulder pitch are rolled with the accordingly offsets. This procedure is according to Stage "B". The other stages are analogic, excluding the percentages for compare, which are different for every stage. Last part of the cycle is to check $\tau$ and if the condition is fulfilled, Left ankle roll and Left ankle roll are the only variables with current values, so it is necessary to assign the absolute values to them.

5. Conclusions
Biped walking robot will have an important part in many future activities of humanity. The paper presented a laboratory model of a humanoid biped platform with 17 degrees of freedom. However, only 10 of the servo motors are necessary for achieving stable process of walking. The paper presented in detail the developed walking algorithm together with its implementation in a commonly used in hobby robotics Arduino platform.

Acknowledgements
The authors would like to thank the Research and Development Sector at the Technical University of Sofia for the financial support.

References
[1] J. Chestnutt, M. Lau, G. Cheung, et al., *Footstep Planning for the Honda ASIMO Humanoid*, International Conference on Robotics and Automation (ICRA), 2005 Barcelona, Spain, doi: 10.1109/ROBOT.2005.1570188
[2] Yu Ogura, H. Aikawa, K. Shimomura, et al., *Development of a new humanoid robot WABIAN-2*, International Conference on Robotics and Automation (ICRA), 2006 Orlando, USA, doi: 10.1109/ROBOT.2006.1641164
[3] K. Kaneko, K. Harada, F. Kanehiro, et al., *Humanoid robot HRP-3*, International Conference on Robotics and Automation (ICRA), 2008 Nice, France, doi: 10.1109/IROS.2008.4650604
[4] F. Plestan, J.W. Grizzle, E.R. Westervelt, et al. *Stable walking of a 7-DOF biped robot*, IEEE Trans on Robotics and Automation, vol 14(4), 2013, doi: 10.1109/TRA.2003.814514
[5] M. Gienger, K. Loffler, F. Pfeiffer, *Towards the design of a biped jogging robot*, Int. Conf. on Rob. and Autom. (ICRA), 2001 Seoul, South Korea, doi: 10.1109/ROBOT.2001.933265
[6] D. K. Pratihar, L. C. Jain, *Intelligent Autonomous Systems: Foundations and Applications*, Springer 2010, ISBN 9783642116759