# Interactive Programming of a Humanoid Robot

Submitted: 16th April 2014; accepted: 29th May 2014

Mikołaj Wasielica, Marek Wąsik, Andrzej Kasiński

DOI: 10.14313/JAMRIS_3-2014/21

**Abstract:**
*This paper presents a control system for a humanoid robot based on human body movement tracking. The system uses the popular Kinect sensor to capture the motion of the operator and allows the small, low-cost, and proprietary robot to mimic full body motion in real time. Tracking controller is based on optimization-free algorithms and uses a full set of data provided by Kinect SDK, in order to make movements feasible for the considerably different kinematics of the humanoid robot compared to the human body kinematics. To maintain robot stability we implemented the balancing algorithm based on a simple geometrical model, which adjusts only the configuration of the robot's feet joints, maintaining an unchanged imitated posture. Experimental results demonstrate that the system can successfully allow the robot to mimic captured human motion sequences.*

**Keywords:** *bipeds, control of robotic systems, humanoid robots, legged robots, teleoperation*

## 1. Introduction

Programming humanoid robot movements is a difficult task. They are usually programmed with numerical optimization techniques or manually, which requires a lot of knowledge and skills about kinematics and dynamics of the robot. Whereas humanoid robot movements should be natural and human-like, human motion capture systems appear to be the preferred solution. However, differences between human and robot kinematics and dynamics, as well as high computational cost cause difficulties in the straightforward solution of this problem.

In our previous work [13] we presented the small-size humanoid robot M-bot (Fig. 1) designed from scratch as an alternative to robots built from commercially available kits [1] [2] [3]. As far as construction is concerned the main assumption was the low cost and relatively high number (23) of Degrees of Freedom (DOF). In our recent work [12] we also presented a manual programming method of the robot.

In this paper, we propose a simple, efficient and low-cost control system for our humanoid robot. The input device is Microsoft Kinect sensor [5], which is relatively cheap compared to professional motion capture (MoCap) systems and has an advantage in that it does not require sophisticated MoCap suits to wear. Kinect sensor and its included Microsoft Kinect Software Development Kit (SDK) provide a 3D Cartesian position of the joints of the observed person. We use all available joints simultaneously to provide the closest possible imitation of motion and static stability of the robot. Our mimicking strategy is based on optimization-free solutions and our balance controller uses a simple geometrical model. It should be noted that servomechanisms in M-bot have significant backlash, no feedback, limited resolution and control rate. Also construction of the robot was not precisely calibrated. Experimental results demonstrate that the controller can track human motion capture data while maintaining balance of the robot.

After introducing related work in the next section, we briefly summarize the overview of the system components in section 3. Section 4 provides a detailed description of the imitation strategy while a simple stability controller is presented in section 5. Experimental results are summarized in Section 6. The paper concludes with Section 7.

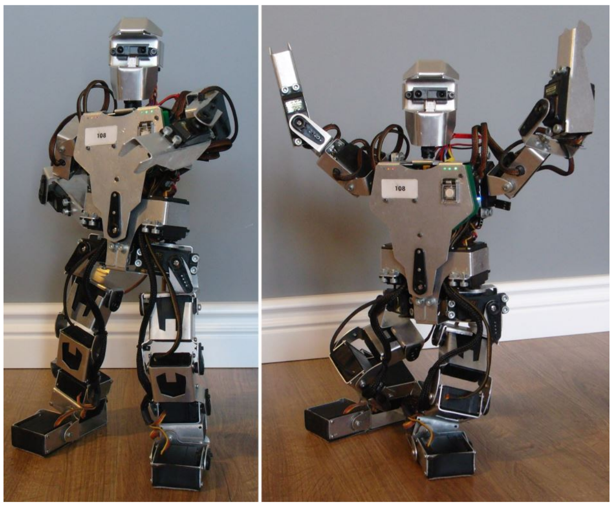
*Fig. 1. The robot (poses obtained from our mimicking system)*

## 2. Related Work

Most of the available control frameworks for humanoid robots require professional motion capture systems, precise commercial robots, and sophisticated algorithms. For example, Yamane et al. [14] developed a system that allows a real force-controlled humanoid robot to maintain balance while mimicking a captured motion sequence. This system employs the model of a simplified robot dynamics to control the balance, a controller for joint feedback, and an optimization procedure yielding joint torques that ensure simultaneously balancing and tracking. However, this system does not allow tracking of motion sequences that include contact state changes, like stepping. In [15] this approach is extended, but only on a simulated robot, by adding a procedure that keeps the center of pressure inside the polygon of support, which varies as the robot moves its feet.

The motion-capture-based approaches to control a humanoid robot mostly use pre-recorded motion sequences. Only a few systems described in the literature can interact with the human operator in real-time. Such a system, described in [11], uses the Kinect sensor and the Nao small humanoid. This system implements balance control and self-collision avoidance by involving an inverse kinematics model and posture optimization. The captured human motion is split into critical frames and represented by a list of optimized joint angles. The similarity of motion is evaluated by the configuration of the corresponding links on the actor and imitator in the torso coordinate system. This work is most similar to our approach, but it was demonstrated on a more complicated robot that, unlike the M-bot, has reliable position feedback in the joints. Moreover, the solution presented in [11] requires numerical optimization, while our approach yields a feasible robot configuration in a single-step, using only geometric computations, and thus it is computationally efficient.

## 3. System Components

Our control system scheme is presented in Fig. 2. Input device of the system is Kinect sensor, which obtains a depth map of the observed scene, where the human being is located. Microsoft Kinect SDK beta 2 processes the cloud of points and returns a Cartesian position of 20 skeletal joints [6]. This data is an input to the trajectory forming algorithm, which converts the 3D position of human joints to angular configuration of the robot's servomechanisms. Configuration is then modified to provide static stability maintenance. Finally corrected information of joints angles is transferred to the robot.

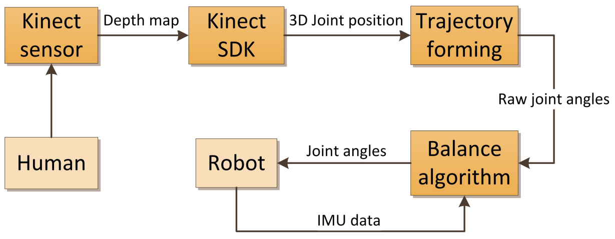

*Fig. 2. Kinect-based programming system overview*

### 3.1. Motion Capture System

Kinect is equipped with RGB camera (1280×1024 pixels for 63×50 degrees FOV, 2.9 mm focal length, 2.8µm pixel size), IR camera (1280×1024 pixels for 57×45 degrees FOV, 6.1 mm focal length, 5.2µm pixel size) [8], and IR pattern projector. Both IR camera and IR projector are used to triangulate points position in space. Minimal depth sensor range is 800 mm and maximal 4000 mm. However, the Kinect for Windows Hardware can be switched to Near Mode which provides range of 500 mm to 3000 mm. Resolution of obtained depth map is 11-bit, which provides 2,048 levels of sensitivity [7]. Highest possible frame rate is 30 fps, which is available for depth and color stream in 640×480 pixels resolution [4].

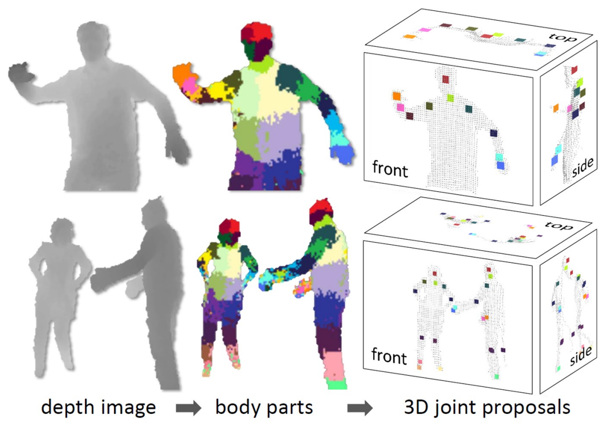

*Fig. 3. Microsoft Kinect SDK Motion Capture process [10]*

Microsoft Kinect SDK is able to track users providing detailed information about twenty joints of the user's body in the camera field of view. Fig. 3 shows an overview of this process. First, from single input depth image human silhouettes are extracted (because it uses depth map it is no longer necessary to wear special MoCap suits). Then a per-pixel body part distribution is inferred. Each color indicates the most likely part labels at each pixel. Finally local modes of each part distribution are estimated to give confidence-weighted proposal for the 3D location of body joints [10].

### 3.2. The Robot

Our robot is a proprietary construction. It was made of scale model servomechanisms and hand-bended 1mm aluminium sheet (Fig. 1). The robot weights 1.8 kg and is 42 cm tall. Total number of DOFs is 23. It has 7 servos in each leg. Usually a robot leg has 6 DOFs, but we added bended toes to improve walking possibilities. Three DOFs are located in each arm, one in the trunk, and two in the head. Inside the robot is located custom made printed circuit board equipped with ATXMega microcontroller, 3-axial accelerometer and 3-axial gyroscope. All 23 servomechanism are directly connected to motherboard. We added a bluetooth module to enable wireless communication with the host computer. This link is used to boost computational capabilities of the microcontroller and to allow the addition of external sensors like Kinect.

## 4. Motion Imitation Problem

Kinect sensor provides MoCap data in 3D Cartesian coordinates form, but our robot's actuators are angular position controlled. In this situation, since we operate in two different coordinate systems we have to define how to understand the adequacy of a human pose by the robot. We considered scaling the operators joints position to compare it with the robot's one, but we realized that our robot has different proportions to a human and also the operator will not always be the same person. As a result we would have to adjust the scale for each joint and for each operator change, which will not guarantee pose adequacy. We observed that humans learn new choreography by imitating pose angular configuration rather than Cartesian position of joints, e.g. children learning some new poses from adults. Therefore we decided to represent human posture as rotational joints configuration and then transfer it directly to the robot's servomechanisms. Operating in multi-angular space guarantees the important configuration-based scale invariance.

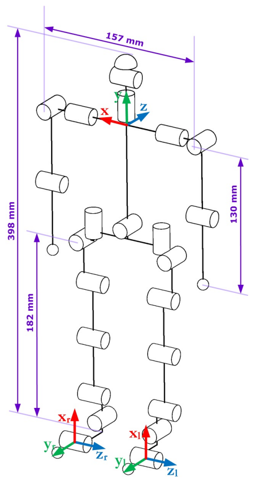

*Fig. 4. Kinematic structure of the robot*

We also wanted to avoid singular poses while still achieving accurate pose imitation to avoid computationally costly iterative calculus. In order to get this we assumed that the number of human DOFs and relative orientation of the rotation of their axes are the same as robot's (Fig. 4). Distance between DOFs is unrestricted and depends on human proportions. As origin of the Coordinate System (CS) we chose the intersection of spine and shoulders axes. Our skeleton is treated as 5 independent kinematic chains (legs, arms and head). Then we provided structural reduction of DOFs. Namely, we consider a kinematic chain divided into several sections with maximum three DOFs each. Knowing the 3D position of 20 human joints from Kinect we are able to obtain inverse kinematics (IK) equations for each particular chain section. These IK equations are simple algebraic calculations and thanks to this, we avoid much slower numerical computations. Because the position of each body part given by Kinect is used, we obtain exact representation of human body configuration, avoiding singular poses at the same time. Moreover this solution is optimization free, because the obtained robot's configuration is practically close to the operator's.

### 4.1. Arms

According to the above assumption we considered our upper limb as made of two sections. The shoulder section-starts in origin of the CS with 2 DOFs and arm section. We calculated particular sections configuration with Eulerian rotation matrices representation. However the arm of our robot has 3 DOFs, while a human one has 4 DOFs, excluding the wrist. In this situation we had to compensate for this disadvantage to a obtain visually acceptable configuration. The advantage of that robot construction is that the elbow can be bended over the straight angle. Therefore we are able to minimize dead zones (Fig. 5). Also, when servo reaches angular limit, configuration for this joint changes with hysteresis.

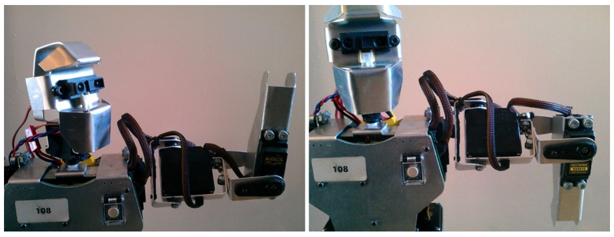

*Fig. 5. Approximating human arm configurations with the limited number of DOFs in the robot*

### 4.2. Head and Torso

Microsoft Kinect SDK beta 2 do not provide human head orientation, therefore we implemented a controller, which directs the robot's "eyes" to look on the operators head. Knowing relative position of the robot head to the Kinect sensor and relative position of the operator head, we were able to triangulate configuration of 2 DOFs of the robot's neck.

Single DOF of the robot torso enables it to tilt. Its angular position is defined as roll angle between origin of CS and the hip CS of the operator.

### 4.3. Legs

We divided the leg kinematic chain into two sections: hip and knee. We do not consider the foot orientation, because information extracted from the Kinect data is usually highly inaccurate (in section 5 we explain how to obtain ankle joint configuration). To define hip joint configuration, we took into account hip to heel vector orientation instead of thigh orientation. This results from the assumption that representation of the operator's leg length is expressed as the proportional distance between hip and heel, which varies from 0 to 100% of the maximal leg length. Therefore knee angular position is calculated with the law of cosines. All of this is necessary to improve the balancing algorithm of the robot. Since the hip joint has 3 DOFs it is critical to define one more vector, perpendicular to hip to heel vector, to obtain three Euler angles. To do this we defined a vector, which is a cross product of the thigh and calf orientation. We used it, rather than feet orientation, because Kinect data for the calf and thigh area is much more accurate.

### 4.4. Configuration Correction

As result of the algorithm calculations we obtained configuration of all joints, a set of 23 elements. Sometimes in this data outliers occur, caused by Kinect reading errors. To smooth the robot movements we average the last 6 measurements for each joint angle. Number of samples was manually adjusted (the greater the number, the less dynamic movement). We also provided simple self-collision avoidance methods. The algorithm limits the range of the angle of each servomechanism, according to the eq. 1:

$$\hat{\Theta}_i^t = \begin{cases} C_{i_{min}} & if\ \Theta_i^t \leq C_{i_{min}}, \\ \Theta_i^t & if\ C_{i_{max}} \leq \Theta_i^t \leq C_{i_{min}}, \\ C_{i_{max}} & if\ \Theta_i^t \geq C_{i_{max}}, \end{cases} \quad (1)$$

where $C_{i_{min}}$ and $C_{i_{max}}$ are angular limits for extreme position of each servomechanism.

### 4.5. Operating Modes

Additionally we implemented two operating modes: a show and face to face. In the first mode the robot and the operator are turned in the same direction i. e. to the audience. In this case configuration of the operator can be directly transferred to the robot. But in second mode, where the robot faces the operator, his configuration has to be mirrored. We change location of configuration in our data set of the left side of the robot to the right and vice versa. If it is necessary we change the angle sign.

## 5. Stability Maintenance

Described in section 4 robot configuration does not account for the different human mass distribution and inertia. Because of Kinects significant measurement error of human feet position, the obtained ankle configurations are encumbered with errors, which is essential for pose stabilization. In this situation we need to implement a separate balance algorithm in the robot. It is based on the static model and controls actual COM position relatively to the robot's supporting feet. When a subsequent pose indicates loses of the robot static stability the control system adjusts COM position by using only the robot feet joints.

### 5.1. Center of Mass

To maintain stability of the robot, first we considered using a stabilizer based on the Zero Moment Point (ZMP) [9]. Servomechanisms in our robot do not provide any feedback information, even about angular position. The robot does not have pressure sensors in it's feet too, therefore implementation of ZMP was not possible. To maintain the static stability we finally decided to implement a controller based on the position of the Center of Mass (COM) projection on the ground plane.

We obtained the mass and relative position of COM of each robot segment from CAD data. We calculated absolute COM position using configuration data and the relative COM position of each segment, respectively to the robot Coordinate System located in the torso. Assuming that the robot consists of $N$ rigid-body links in three dimensions, the absolute COM position is given as

$$COM_{x,y,z} = \frac{\sum_{i=1}^{N} m_i [x_i, y_i, z_i]'}{\sum_{i=1}^{N} m_i}, \quad (2)$$

where $m_i$ is a mass and $[x_i, y_i, z_i]'$ is a position of each robot segment.

Because used servomechanisms have significant backlash and positioning error of about $\pm 1.5°$, we do not consider COM as point, but as a ball. We assume that the ball represents punctual COM with embedded measurement uncertainty as its radius. This fundamental assumption implicates that to maintain static stability, projection of the COM to ground plane does not have to lie in the foot supporting area, but on the line segment connecting centers of the feet. Therefore space of existing solutions was reduced from two to one dimension.

### 5.2. Simple Stability Model

The main task of our system is to imitate the operating person accurately, therefore any obtained robot's configuration modifications are not recommended. Also the operator's foot orientation readings are of poor quality especially since the robot's ankle joints are essential to balance. Combining these arguments we propose a simple solution.

Position of the robot's COM is calculated before executing each posture, 50 times per second. To adjust COM position we use only feet orientation, rest of the body joints are fixed. Changing feet orientation has negligibly small influence to mass distribution, therefore we can assume that the position of the robot's COM is constant according to the robot's CS in current frame. Thanks to that, the robots pose is same as the operators. Also this solution is optimization free, because no iterations are needed to estimate COM position for each configuration change.

The only problem to solve is the calculation of orientation of the robots foot according to its CS. To do this we propose a simple geometrical model presented in fig. 6. First we assumed that the ground on which the robot stands is flat, so the normal vector to the surface is parallel to gravity vector all over the place. Then we can introduce vector $\vec{V}$, which has to be parallel to the normal of the support surface to maintain static stability. Since the feet are robot's support surface, with its centers in $P_p$ and $P_l$ points, the $\vec{V}$ should start in line segment $\overline{P_p P_l}$ and end in the COM. Also observing that $P_p$ and $P_l$ should lie on the same surface, $\vec{V}$ has to be perpendicular to the $\overline{P_p P_l}$. Combining this assumptions we obtained equation 3 to calculate $\vec{V}$ orientation.

$$\vec{V} = \overrightarrow{P_L P_P} \times (\overrightarrow{P_L COM} \times \overrightarrow{P_L P_P}) \quad (3)$$

Knowing $\vec{V}$ orientation according to the robot's CS we can substitute it into inverse kinematics equations. Because the ankle joint has two DOFs, only a single 3D vector is needed to define its orientation, and we can easily calculate its configuration using explicit algebraic transformations.

### 5.3. Side fall prevention

Introduced ankle strategy guarantees stability maintenance only when projection of COM to the line segment will be inside this segment. In other words

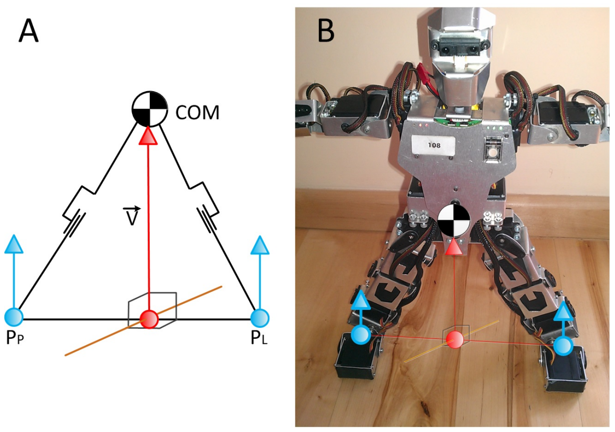

*Fig. 6. Stability maintenance strategy: schematics (A) and visualization on the robot (B)*

we have to detect the possibility of exceeding outer boundary of the foot by COM and adjust configuration to prevent against falling to one side. We decided to detect the possibility of falling by measuring the angle between two vectors (eq. 4):

$$\angle(\vec{x},\vec{y}) = \arccos \frac{(\vec{x},\vec{y})}{|\vec{x}|\cdot|\vec{y}|}. \tag{4}$$

For each vectors pair angles are specified in eq. 5:

$$\begin{aligned} \alpha &= \angle(\overrightarrow{P_P COM},\vec{V}),\\ \beta &= \angle(\overrightarrow{P_L COM},\vec{V}),\\ \gamma &= \angle(\overrightarrow{P_P COM},\overrightarrow{P_L COM}). \end{aligned} \tag{5}$$

To find out if next pose is stable we have to solve eq. 6:

$$Pose = \begin{cases} Stable & \text{if } \alpha+\beta=\gamma \wedge \alpha \neq 0 \wedge \beta \neq 0,\\ Stability\ limit & \text{if } \alpha = 0 \vee \beta = 0,\\ Unstable & \text{if } \alpha+\beta > \gamma. \end{cases} \tag{6}$$

We simply calculate whether vector $\vec{V}$ is inside our modelled triangle (Fig. 6(A)). If it is inside the pose will be stable and no adjustments are needed. Configuration can be directly transmitted to the robot. If it is on the stability boundary, configuration also can be transmitted. In the third case we have to modify existing configuration to guarantee pose stability. However doing this we should as little as possible modify pose to maintain it's similarity to the operator pose.

Fig. 7 presents scheme of adjusting pose. The aim is to transform the unstable configuration (A) to a stable one (D). First (A) we calculate the proper angle by using eq. 5. This angle informs us about vertical deviation. Rotating our model by this angle (B), we can calculate the intersection point of ground and the edge of the triangle. Knowing the position of this point and position of vertex we are able to calculate the distance between them (C). We can simply shorten the appropriate leg, as in a method of adjusting leg length in section 4.3. Finally we obtain a stable pose (D), which can be transferred to the robot, after updating the foot orientation.

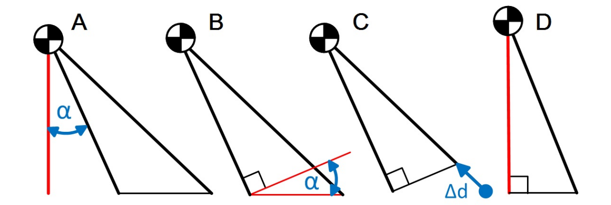

*Fig. 7. Illustration of the side fall prevention strategy (from left to right)*

### 5.4. Single Leg Standing and Correction from IMU

Our presented balance algorithm is also able to provide stabilization on a single leg. It requires defining vector $\vec{V}$ as $\vec{V} = \overrightarrow{P_P COM}$ or $\vec{V} = \overrightarrow{P_L COM}$. Then the balancing algorithm will be adjusting orientation of robot's ankle so that the COM will be always over center of supporting feet.

By adding the inclination angles obtained from the IMU to the orientation of vector $\vec{V}$ we can also compensate to some degree the ground tilting.

This additional features have not been implemented, but theoretical considerations imply that the idea is correct.

## 6. Experimental Results

### 6.1. Controller Implementation

We implemented the controlling framework on a standard PC computer, which is connected to the robot by bluetooth. Each robot actuator is controlled by the internal position feedback loop running at 50 Hz for position control. Therefore our implementation of the posture controller runs at this rate. Our control system uses a configuration extracted from Kinect data twice because the Kinect provides skeletal data at about 25 Hz. Also it should be noted, that information about each actuator position is not visible for our main controller. This results from the lack of any feedback information from the low-cost servomechanisms used.

The robot mass distribution is obtained from a CAD model. Similarly the angular position of gears is evaluated by eye, so it approximates only configuration based on the appearance of the robot. Also the support platform is not levelled. All these factors cause the inconsistency with the real data. To compensate for this, while the robot is in initial pose, we manually adjust the foot joints until the robot is able to stand upright on its own. Then we add these constant offsets to the reference joint angles in order to match the initial reference robot pose with the actual initial pose.

### 6.2. Control of the System

The control framework is designed to be user-friendly. Because the system was presented with audience participation, we implemented an operator chosen algorithm. From humans in the Kinect field of view, the person who raises their hand will take control of the robot (Fig. 8 A). From this moment the affection is fixed on that person.

Controlling the robot utilises the whole operators body, so it is not possible for him to manage the control application in a traditional way by using a mouse or keyboard. Therefore we implemented voice commands in the robots control system. Thus it is possible to establish a connection to stop or to turn off the robot. Mainly this mode is used to switch a controlling person.

### 6.3. Tracking Human Motion Data

In our experiment the operator presents simple and the hardest poses for the robot involving both upper-body and legs. Snapshots from the trial[1] are shown in fig. 8. Each pose is presented in frontal view, to minimize occluding of body parts, which result in Kinect readouts errors. If the error occurs as a significantly fast joints position step, which is faster than people are able to perform, the robot trying to follow such a movement in some cases can fall down, because of the high value of dynamic forces.

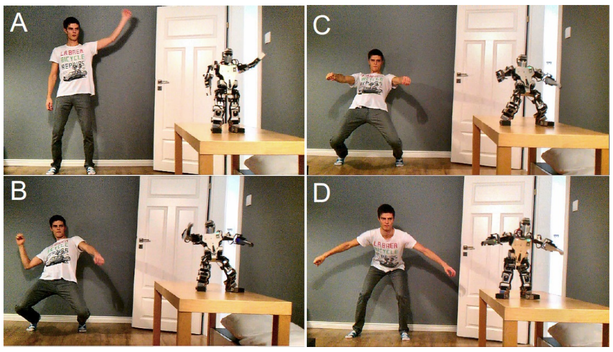

*Fig. 8. Snapshots from the experimental validation: taking the control (A), various postures achieved by the robot (B,C,D)*

Analysing captured video frame by frame we observed insignificant lag when mimicking motion using Kinect. This lag is mainly caused by the Kinect SDK calculations time. However, the robot movements stay as dynamic as the operator. Both feet and hands of the robot have the same maximum speed, which does not significantly affect the robots stability, thanks to the robust stability controller. This is an original property of the robot compared to other solutions [15] [11]. Sometimes when the robot joint reaches its angular limit, it performs additional corrective movements, which are not performed by the operator. Such movements help the robot to avoid singular configurations and improve robot-human pose similarity limitation. Finally, the overall motion sequence is very similar to the reference motion performed by the operator.

## 7. Conclusion

In this paper we have proposed a straight-forward method to build a real time full body human imitation system on the proprietary humanoid robot. First we suggest to rely on the angular representation of a human pose as it is independent of human size, uses a full set of available joints data, and needs no optimization. We also propose a straight-forward balance control strategy based on ankle joints control and have proved it to be efficient. Experimental results demonstrate that our implementation successfully controls a humanoid robot so that it tracks human motion capture data while maintaining the balance.

## ACKNOWLEDGEMENTS

This research was funded by the Faculty of Electrical Engineering, Poznan University of Technology grant, according to the decision 04/45/DSMK/0126.

## Notes

[1] A supplemental video is available at: http://lrm.cie.put.poznan.pl/ROBHUM.mp4

## AUTHORS

**Mikołaj Wasielica*** – Poznan University of Technology, Institute of Control and Information Engineering, ul. Piotrowo 3A, 60-965 Poznań, e-mail: mikolajwasielica@gmail.com, www: http://www.cie.put.poznan.pl/.
**Marek Wąsik** – Poznan University of Technology, Institute of Control and Information Engineering, ul. Piotrowo 3A, 60-965 Poznań, e-mail: wasik.m@gmail.com, www: http://www.cie.put.poznan.pl/.
**Andrzej Kasiński** – Poznan University of Technology, Institute of Control and Information Engineering, ul. Piotrowo 3A, 60-965 Poznań, e-mail: Andrzej.Kasinski@put.poznan.pl, www: http://www.cie.put.poznan.pl/.

*Corresponding author

## REFERENCES

[1] Aldebaran Robotics, 2012 (online product specification, link: www.aldebaran-robotics.com).

[2] I. Ha, Y. Tamura and H. Asama, "Development of open humanoid platform DARwin-OP", *Advanced Robotics*, vol. 27, 2013, no. 3, pages 223–232, DOI:10.1080/01691864.2012.754079.

[3] Kondo Kagaku Co., Ltd, 2012 (online product specification, link: www.kondo-robot.com).

[4] Microsoft, "Kinect for Windows Sensor Components and Specifications", 2012 (online documentation, link: http://msdn.microsoft.com/en-us/library/jj131033.aspx).

[5] Microsoft, "Kinect for X-BOX 360", 2010 (online product specification, link: http://www.xbox.com/en-US/kinect).

[6] Microsoft Kinect SDK, "Getting Started with the Kinect for Windows SDK Beta from Microsoft Research", 2011, pages 19–20, (online document, link: http://www.microsoft.com/en-us/kinectforwindowsdev/Downloads.aspx).

[7] Microsoft, "Kinect Sensor", 2012 (online documentation, link: http://msdn.microsoft.com/en-us/library/hh438998.aspx).

[8] OpenKinect, "Protocol Documentation", 2012 (online document, link: http://openkinect.org/wiki/Protocol/_Documentation/#Control/_Commands;a=summary).

[9] P. Sardain and G. Bessonnet, "Forces acting on a biped robot. Center of Pressure – Zero Moment Point", *IEEE Trans. Systems, Man, and Cybernetics, Part A*, vol. 34, 2004, no. 5, pages 630–637.

[10] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman and A. Blake, "Real-time human pose recognition in parts from single depth images", *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Providence, USA (2011), pages 1297–1304, DOI:10.1109/CVPR.2011.5995316.

[11] F. Wang, C. Tang, Y. Ou and Y. Xu, "A real-time human imitation system", *Proc. 10th World Congress on Intelligent Control and Automation*, Beijing, China (2012), pages 3692–3697, DOI:10.1109/WCICA.2012.6359088.

[12] M. Wasielica, M. Wąsik, A. Kasiński and P. Skrzypczyński, "Interactive Programming of a Mechatronic System: A Small Humanoid Robot Example", *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)* Wollongong, Australia (2013), pages 459–464, DOI:10.1109/AIM.2013.6584134.

[13] M. Wasielica, M. Wąsik and P. Skrzypczyński, "Design and applications of a miniature anthropomorphic robot", *Pomiary Automatyka Robotyka*, vol. 2, 2013, pages 294–299.

[14] K. Yamane, S. Anderson and J. Hodgins, "Controlling humanoid robots with human motion data: Experimental validation", *Proc. IEEE/RSJ, Int. Conf. on Humanoid Robots*, Nashville, USA (2010), pages 504–510, DOI:10.1109/ICHR.2010.5686312.

[15] K. Yamane and J. Hodgins, "Control-aware mapping of human motion data with stepping for humanoid robots", *Proc. IEEE/RSJ ,Int. Conf. on Intelligent Robots and Systems*, Taipei, China (2010), pages 726–733, DOI:10.1109/IROS.2010.5652781.