2012 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 7-12, 2012. Vilamoura, Algarve, Portugal

Applying Human Motion Capture to Design Energy-efficient Trajectories for Miniature Humanoids

Kiwon Sohn*
Mechanical Engineering and Mechanics
Drexel University
Philadelphia, USA
ks948@drexel.edu

Paul Oh**
Mechanical Engineering and Mechanics
Drexel University
Philadelphia, USA
paul.yu.oh@drexel.edu

Abstract— **In this research, an approach to optimize motions for a humanoids is presented. Rapidly-exploring Random Tree(RRT) were used to plan an initial suboptimal motion. A reinforcement learning was then implemented to optimize the trajectories with respect to energy consumption, similarity to a human's natural motion and, physical limits. Energy cost was estimated by joint torque from a dynamic model, and validated against actual measured torque values using system identification (SID). With a motion capture system, human motions were collected for a given set of tasks, producing a representative "natural" motion, another cost for optimization. Physical limits of each joint ensured spatial and temporal smoothness of generated trajectories. Finally, an experimental evaluation of the presented approach was demonstrated through simulation using MiniHubo model in OpenRAVE.**

I. Introduction

As humanoids approach human size, strength and range of motion, they are expected to execute increasingly complex tasks and human-level performance. To meet this, kinematic path-planning are applied on high degree-of-freedom(DOF) robots. Algorithms based on potential field [1] are often employed with applications mostly limited to mobile robots.

Humanoids are high-DOF robots that involve large search spaces. As such, probabilistic-based approaches [2] like Rapidly-exploring Random Trees(RRT) [3] and its variants IKBiRRT [4] and CBiRRT [5] are often used. These algorithms are well-suited for a humanoid because the its end-effector can serve as the end-goal [6]. The concept of Task Space Region Chains(TSRs) introduced coupled with CBiRRT2 can even handle kinematic models in a dynamic environments. These search-based methods guarantee collision-free and quasi-static balance in generated paths. However, these algorithms do not consider energy-efficiency or "natural" appearance of the planned motions.

Another more direct approach for complex motion planning is to employ captured human-motion data. Such approaches are traditionally used in animation [7], but have also been applied to generate humanoid whole-body motions [8],[9]. Extracting motion primitives from captured data allows humanoids to create novel motions beyond simply mimicry of people [10],[11].

However, differences between human and humanoid joints and structures present significant challenges using motion-capture based approaches. [12] scales and limits motions to overcome such challenges by considering kinematic and dynamic constraints like joint angles and accelerations. Motion-capture can produce very natural-looking motions. However humanoid motions are often not optimized for energy efficiency. People possess core sets of muscles. Optimal (or near-optimal) motions may favor the use of different sets of joints. The net effect is that optimal motions for a human and a humanoid are often different.

Recently, a broad range of optimization techniques have been applied to humanoid motion planning. In early studies, many researchers focused on the kinematic structure of humanoids, trying to minimize error between planned trajectories and actual movements [13],[14]. Dynamic constraints like torque limits have been included to satisfy humanoid physical limits. For instance, [15] began with captured motion data and invoked Lagrange multipliers to limit motions within joint constraints. In other studies, they used a pre-calculated motion from path-planning algorithms and applied optimization techniques to satisfy those same physical limits [16]. Unlike [15], joint torque was minimized instead of kinematic differences. In other words, natural motion was not an emphasis. To address this [17] recorded instances of a particular motion, extracting "principle" ones that best represented the set. They then applied a bi-level optimization to minimize kinematic hand and joint jerks and the joint torque's time rate of change.

Another class of planning methods try to specifically generate life-like motions by accounting for human posture constraints. Khatib in [18] presented a framework for whole-body humanoid control, addressing multiple hierarchical constraints and contacts simultaneously. The framework ensures that constraints are not violated by projecting a given motion task into the null space of constraints. This powerful method requires torque control which is often not an option on many humanoids due to safety designs. [19] introduced a method for re-targeting robot motions to be more human-like by optimizing spatio-temporal correspondence. This method tries to produce an optimal set of temporal and spatial shifts but does not consider energy consumption as a cost.

This paper introduces a method to balance two goals: generate life-like humanoid motions and minimize joint energy consumption. To achieve these two goals simultaneously a reinforcement learning agent (Q-learning algorithm) was used. For a given task, multiple candidate motions which are collision-free and statically balanced are generated using

CBiRRT. At each time-step, the corresponding pose from each candidate motion is one possible state that the robot can attain. Thus, the goal of the learning agent is to choose an optimal sequence of states from these choices with respect to defined constraints.

Each state represents a joint angle set. The corresponding action reflects a change between the current state or pose and possible future states. Each state-action pair was given three penalty values, representing the costs of entering a given future state from the current one.

The first penalty value was the torque at each joint. The torque was calculated using a humanoid dynamic model developed with ProPac [20]. The humanoid used, called Mini-Hubo is a scaled-down version of Drexel's full-sized Jaemi-Hubo. Mini-Hubo was developed to serve as a tool to prototype humanoid motions [21]. The second penalty value considered differences between a planned motion and a human's natural motion. Here, a set of human motion trajectories for many sample goal positions was captured. Then, from this collection, trajectories for new goal positions were regenerated using nearest-neighbor algorithms. The resulting regenerated trajectory was then converted to be compatible with Mini-Hubo's joint structure. The third penalty value employs joint limits to produce temporally and spatially smooth trajectories. Here, limits on joint angles and velocities, and temporal change of joint torque, were assigned. The net result is an overall penalty for the learning agent that was a weighted sum of these three penalties.

Section II describes optimization of the overall architecture based on reinforcement learning. Section III shows how the initial set of planned paths were produced from CBiRRT. The Mini-Hubo dynamic model and joint torque calculations are then given in Section IV. Section V explains how the database of human movements using a motion capture system is constructed and how the recorded trajectories were used to find a natural motion prototype for a given task. Lastly, experimental results using a model of Mini-Hubo in OpenRAVE [22] are given in SectionVI.

II. System Overview and Reinforcement Learning Agent

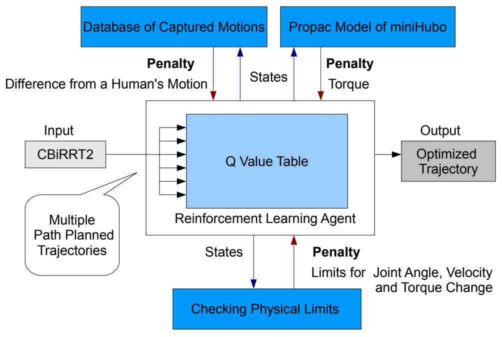

Database of Captured Motions
Propac Model of miniHubo
Penalty
Difference from a Human's Motion
States
Penalty
Torque
Input
CBiRRT2
Q Value Table
Output
Optimized Trajectory
Multiple Path Planned Trajectories
Reinforcement Learning Agent
States
Penalty
Limits for Joint Angle, Velocity and Torque Change
Checking Physical Limits

Fig. 1. Trajectory-Optimizing System based on Q-Learning

Figure 1 illustrates the proposed reinforcement learning based trajectory optimization architecture. In this paper, the Q-learning algorithm for the reinforcement learning agent is used. Q-learning uses an action-value function to compute the expected rewards of taking a given action in a given state [23]. This function generates a fixed policy. Equation 1 shows how Q values in the agent are updated

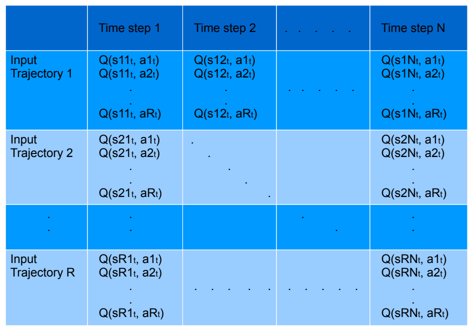
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t(s_t, a_t) * \qquad (1)$$

$$[P_{t+1} + \gamma \arg\max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

At every learning time t, there are multiple states s_t and a set of actions(a_t) which exist per state. α is the learning rate and P is the penalty value. γ is a discount factor for the maximum future Q value.

The CBiRRT(Constrained Bidirectional RRT) planner from the Constrained Manipulation Planning Suite (CoMPS) [24] was used to generate multiple solution trajectories for a given manipulation task. Example tasks include box grasping and target reaching. For such tasks, Mini-Hubo was modeled and OpenRAVE was used to check collisions and stability. A generated trajectory consists of a set of angles from every joint of MiniHubo during movement.

Every planned trajectory was initially divided into a set of joint angles at every time step, and each set of angles became all different states. In this study, 10 $msec$ time steps were used to synchronize the frequency of human motion capture. Figure 2 shows the Q value table for t When there are R input RRT trajectories, and the necessary time for finishing the input trajectory is $N * 10$ $msec$, $R * N$ different states exist in the Q value table. Therefore, each state in a row of the Q table is a set of joint angles at a given time step. These states correspond to a particular solution trajectory. Each input trajectory may take a different amount of time to finish. Hence each trajectory's had to be equalized. Therefore, the longest solution determined the length of the Q value table, N. Shorter trajectories were padded with their final state to fill in the remaining time steps in the Q table.



	Time step 1	Time step 2	. . . . .	Time step N
Input Trajectory 1	$Q(s11_t, a1_t)$ $Q(s11_t, a2_t)$ . . $Q(s11_t, aR_t)$	$Q(s12_t, a1_t)$ $Q(s12_t, a2_t)$ . . $Q(s12_t, aR_t)$	. . . . .	$Q(s1N_t, a1_t)$ $Q(s1N_t, a2_t)$ . . $Q(s1N_t, aR_t)$
Input Trajectory 2	$Q(s21_t, a1_t)$ $Q(s21_t, a2_t)$ . . $Q(s21_t, aR_t)$	. . . . .		$Q(s2N_t, a1_t)$ $Q(s2N_t, a2_t)$ . . $Q(s2N_t, aR_t)$
. .	. .		. .	. .
Input Trajectory R	$Q(sR1_t, a1_t)$ $Q(sR1_t, a2_t)$ . . $Q(sR1_t, aR_t)$	. . . . .	. . . . .	$Q(sRN_t, a1_t)$ $Q(sRN_t, a2_t)$ . . $Q(sRN_t, aR_t)$

Fig. 2. Q Value Table at Learning Time t

In the Q table of Figure 2, state smn_t is a joint angle set from the time step n of the mth input trajectory. The Q value table also shows that each state has its own set of actions, a_t. When the current state is smn_t, a possible action for this state is defined as a transition between state smn_t and any of the states in time step $n+1$. For example, if the present learning time is t, then the current state is $sm1_t$. This current state is a set of joint angles at time step 1 corresponding to the m^{th} input trajectory. If one selects aj_t as an action for the current state, the next state becomes $sj2_{t+1}$. This state is a set of joint angles at time step 2 of the j^{th} input trajectory. With the Q value updating equation above, $Q(sm1_t, aj_t)$ can be updated based on the previous value of $Q(sm1_t, aj_t)$ and the predicted maximum future Q value (2).

$$\arg\max_{a_{t+1}} Q(sj2_{t+1}, a_{t+1}) \tag{2}$$

At the next learning time t+1, a set of state $sj2_{t+1}$ and selected action a_t repeats the whole process. When states in the last column of the Q value table became updated, the state which should be updated is automatically redirected to one of the states which are in the first column of Q table. This process repeats until convergence is reached.

To update Q values at each set of state and action, 3 different penalty values were defined as follows: (1) the sum of torque values of each joints in the current state were assigned as a penalty value. Previous studies showed that joint torques can be an effective method of measuring energy consumption [25]. For calculating torque values, a dynamic model of Mini-Hubo was built using ProPac, and the torques of each joint in MiniHubo were calculated for all states. For more accurate prediction of torques over joints, the dynamic model was refined with system identification using real torque data collected from the actual Mini-Hubo. (2) the differences between planned trajectories and a human's natural motions were also penalized. Human motions for various tasks were captured and converted into ones which met Mini-Hubo's kinematic constraints. After building a database of recorded trajectories, a trajectory which can reach a desired goal under a given task was calculated using a nearest-neighbor algorithm. Due to the kinematic differences between the human and Mini-Hubo, a direct comparison of joint positions is not always meaningful. The normalized joint velocities were used instead as a means of comparison. The differences between normalized velocities for the human motion and a given state formed another penalty value. (3) movements also received penalty values when they exceeded joint angle, velocity and torque over time constraints. This joint penalty ensured that the final trajectory met the physical limits of the robot. Penalizing the angle and velocity of joints ensures a smooth trajectory over time. Penalties also limit the rate of change of torque, thus ensuring spatial smoothness. By choosing suitably small time steps, the resulting trajectory is also kept collision-free and statically-stable.

By iterating with the weighted sum of penalties, Q values became updated until they converged. Also, the learning agent generated a new trajectory which minimizes penalty at each time step. Since every state in the Q value table came from the RRT trajectories which complete the task, states in the last column of Q value table meet the goal.

Possible state transitions are currently defined only between the current state and states in the immediate future. It is theoretically possible to move to distant future states in a single action, since all states are simply a set of joint angles. However, it is difficult to assure that this action will be collision-free and stable due to the potentially large change in joint angles. The large number of possible states also increases both computation and convergence times for the Q table. To add variety to the states, a large number of solution trajectories from the RRT algorithm was generated.

By assigning different weights on each penalty value, the properties of the output trajectory could be controlled. For example, a trajectory which focuses on minimizing joint torques instead of a more natural appearance could be found by increasing the weight of the torque penalty values. Since the Q learning algorithm explores all possible states from the set of input RRT trajectories, the final output trajectory from our learning agent is globally optimized.

III. Path Planning using Rapidly-exploring Random Tree

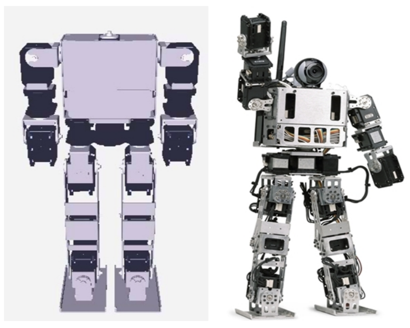


Fig. 3. Virtual Hubo and MiniHubo

To plan a trajectory which is an input for our optimizing system, the Constrained Bidirectional Rapidly-exploring Random Tree (CBiRRT)[6] was used. This path planning algorithm found a set of joint angles at every time step until it placed the end effectors in sample positions of a goal area. This algorithm was developed with OpenRAVE plugins, namely the Constrained Manipulation Planning Suite (CoMPS)[24]. Consequently Mini-Hubo was built in OpenRAVE. The actual and virtual Mini-Hubo is shown in Figure 3. Virtual Mini-Hubo is designed to have same kinematic and dynamic properties of MiniHubo.

Various kinds of manipulation, like reaching a box, could be implemented using CoMPS and Virtual Mini-Hubo. In the authors' research, there was a box in front of Virtual Mini-Hubo. Here, the humanoid's arms were commanded to reach each side of the box. Both the virtual and actual Mini-Hubo do not have hands. Thus, tips of both lower arms were assigned as end effectors. Figure 4 shows a motion of Virtual Mini-Hubo along a generated RRT trajectory.

For this reaching task, multiple RRT trajectories were generated and each planned motion commanded Virtual Mini-Hubo's end effectors to reach each side of the box while conserving task constraints [6]. After collecting multiple

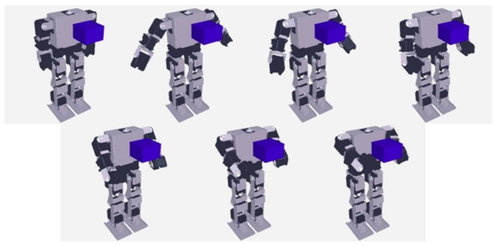


Fig. 4. Reaching a Box using a Trajectory from CoMPS

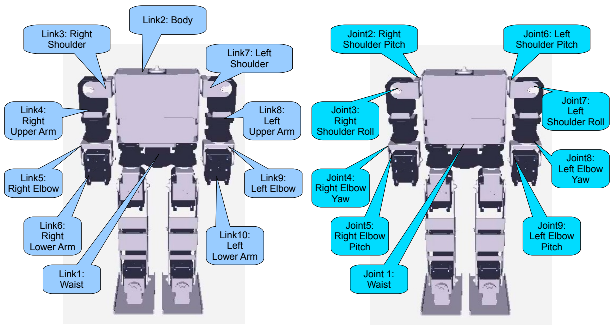


Fig. 5. Defined Links and Joints in MiniHubo Model

planned trajectories, each one was divided along the time coordinate. As explained in Section II, each time-divided trajectory became an individual state at the corresponding time step and input trajectory of the Q value table.

IV. Prediction of Energy Consumption

To predict energy consumption of each state in the Q value table, it was necessary to calculate Mini-Hubo's torque values. The generated RRT trajectory was originally expressed as joint angles during motion. To explicitly relate joint angles to torques, a Mathematica package called ProPac [20] was used. ProPac supports the assembly of simulation models for mechanical systems such as robots. ProPac generated an explicit set of non-linear equations that model Mini-Hubo.

To build Mini-Hubo's model with ProPac, all necessary data for individual joints and links, such as part masses, centers-of-mass, and moments of inertia were collected. For such data, the CAD toolkit Open Inventor was employed. A system interconnection structure was created from the joint structure of Mini-Hubo. Figure 5 shows all the defined links and joints for Mini-Hubo. Only the motion of the upper body from Mini-Hubo was considered in this paper. As such, 10 links and 9 joints were defined for this process. Mini-Hubo's waist defined the model's reference frame.

Poincare equations of motion was formulated on the Mini-Hubo model. Equation 3 shows the dynamic equations generated with ProPac. q is the generalized coordinate vector of the joint angle and p is a quasi velocity of a joint.

$$M(q)\dot{p} + C(q,p)p + F(q) = B_p \tag{3}$$

where

$$C(q,p) = -[\frac{\partial M(q)p}{\partial q}V(q)] + \frac{1}{2}[\frac{\partial M(q)p}{\partial q}V(q)]^T$$
$$+[\sum_{j=1}^{m} p_j {X_j}^T]V^{-T}$$
$$F(q) = V^T(q)\frac{\partial u(q)}{\partial q^T}, T_p = V^T(q)B$$

$u(q)$ is the potential energy function and M is a spatial inertia matrix. B_p denotes the generalized forces represented in the p-coordinate frames and B denotes the generalized forces in the velocity of q coordinate frame. In combination with Equation 4, kinematic equations of each joint and link, these equations provide a closed set of equations.

$$\dot{q} = V(q)p \tag{4}$$

ProPac calculates all components of Poincare equation and all of Mini-Hubo's joint angles, velocities and accelerations are known. One can thus compute B which is a set of applied generalized motor forces of each joint. Torque values for the motors at each joint can then be easily calculated by dividing each generalized force by the known joint velocities. Using this method of inverse dynamics, one could generate torque-calculation functions for Mini-Hubo's joints under a given trajectory with Mini-Hubo's forward kinematic model.

Torque-calculation functions were refined using the MATLAB System Identification (SID) toolkit to more accurately predict motor energy consumption. SID is software which constructs mathematical models of dynamic systems from measured input-output data. To use this toolkit, Mini-Hubo was first commanded a defined sample trajectory and each joint's actual torque data was recorded. Next, a new SID dynamics model was constructed using previously formed ProPac torque-prediction functions. These functions were combined with the recorded torque data.

After defining SID model parameters to be refined, like the center of mass, Levenberg-Marquardt estimation was used to construct a new torque-calculation model for each joint. Such estimation could optimize parameter values in the dynamic model that was constructed. This made the computed torque values from the model become more similar to those measured from Mini-Hubo's actual motors.

Figure 6 shows calculated torque values of Mini-Hubo before and after parameter estimation. Here, Mini-Hubo's right arm was commanded to rotate counter-clockwise. The torque for the motor at the right shoulder pitch joint was measured. The top figure shows both the measured and predicted (curved line) torque output data using the constructed dynamics model before parameter estimation. The bottom figure shows the predicted torque values from the model after parameter estimation. One sees that the model became more similar to measured data after parameter estimation.

A refined torque-prediction model of Mini-Hubo was used and energy consumption of each state in the Q-value table was calculated. Penalty values were assigned by weighting energy consumption. For simplicity, Mini-Hubo's head was

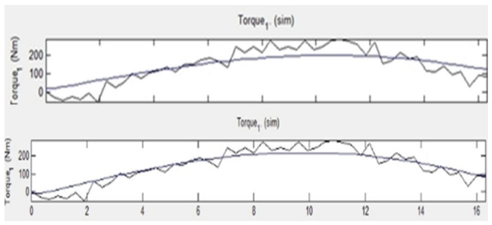


Fig. 6. System Identification

not modeled. This is a reasonable assumption given that the head's mass with respect to the rest of the robot, is small. Given this low mass, the impact on Mini-Hubo's dynamics is likely to be small. Future work will incorporate the head's mass and size to Mini-Hubo's model. It is expected that this addition will lead to even better system identification.

V. Estimation of Human's Motion Pattern

A human's natural motion pattern and a state in the Q value table to be updated, were compared to measure similarity. For this, repetitive motion's of a human were captured using 18 Optitrack FLEX:V100R2 cameras and Arena software.

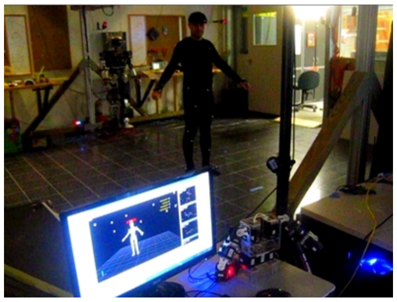


Fig. 7. Human and MiniHubo in Motion Capture System

NatNet SDK in Arena software provides rotational data in quaternion format for skeletal links of the human body. For this paper's experiments rotations were converted to Euler angles. This enabled direct assignment of angles for Mini-Hubo's joints. Different sequences of Euler rotation (roll, pitch and yaw) could result in Mini-Hubo having different kinematic poses of links. Therefore, for each of Mini-Hubo's links, a sequence of Euler rotations was identified. This identification ensured that the Euler and quaternion representations resulted in the same kinematic pose. Next, all quaternion angle data from human motion capture data was converted to Euler form to compute trajectories for Mini-Hubo. Figure 8 shows the conversion.

The authors observe that humanoids mainly uses its upper body for repetitive tasks that include: 1)Reaching for an object, 2)Moving an object, and 3)Mimicry of a human motion such as dance. With this observation, human motion data in the above categories was captured and a database for each task was built. For example, 27 different motions were captured to build a database of motions for the reaching for a box-related task. For each motion, a box was located in all of

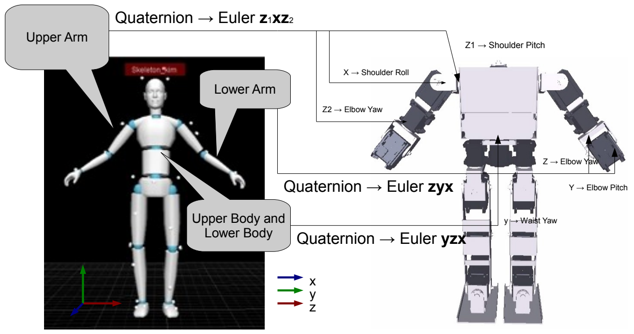


Fig. 8. Conversion from Rotation Data of a ARENA Human Model to Joints Angles of Mini-Hubo

the different goal positions. Figure 9 (right and left) show 27 sample positions of an object that was located during motion capture. Within a limited area in which the human does not need to walk, turn or bend the knees to reach an object, an object was located in one of the 27 sample positions which have uniform distances with other sample positions.

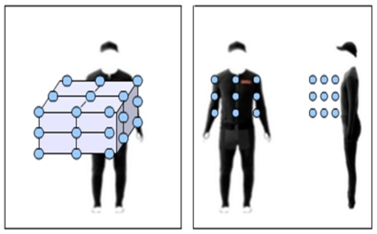


Fig. 9. 27 Goal Positions for a Reaching a Box Task

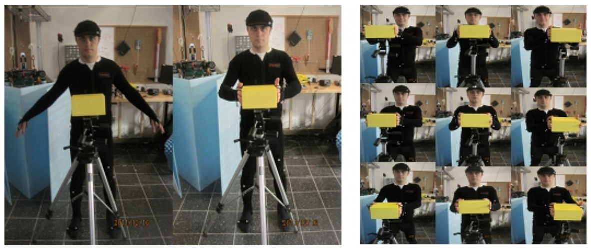


Fig. 10. Initial and Last Pose of a Human for One Sample Goal Position of Box(Left) and for Various Goal Positions(Right)

In case that a given task for MiniHubo is to reach an object which is located in one of 27 sample positions in database, we could use a saved trajectory in database directly. However, if a goal space is not same with any sample goal positions in 27 pre captured motions, a weighted nearest neighborhood algorithm was used for estimating a human's trajectory for a given goal space. 4 neighboring sample positions which have nearest distances from a given goal position were selected among 27 sample positions in the database. Trajectories of motions for selected sample positions were gathered, then the average of gathered trajectories was calculated using different weighting factors for each trajectory. To give more weights on a trajectory from the nearest object, the inverse value of Euclidean distance between a given goal object and the nearest object was used for a weighting factor. Because of dimensional differences between a human body and MiniHubo, 4 nearest neighborhoods were chosen based on ratio of body length and the lengh, width, height, and goal location of the object.

Since the size of an object which MiniHubo should reach for in a given task can vary, a nearest neighborhood algorithm was implemented individually for each arm. This approach generated different goal positions for each arms of MiniHubo and made it possible for MiniHubo to have trajectories which can reach objects of various sizes. In case of objects ouside of the boundary for a captured area, additional actions such as turning of waist joint or bending knees were necessary.

Figure 11 shows a trajectory of MiniHubo which was generated from database of human's motion. This shows that people bend both arms at the same time when they try to reach a goal object. And it also demonstrates that people can reach each side of an object using the shortest path.

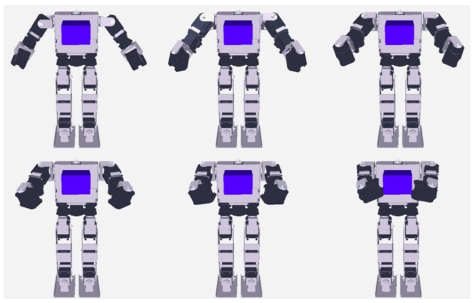


Fig. 11. Reaching a Box using a Trajectory from Database of Human's Motion

At each state in Q-value table, velocity value between current state and previous state was calculated and compared with velocity of trajectory which was estimated from human's motion database. Then, differences between two velocity values were combined with a weighting factor and assigned as a penalty value.

In this proposed learning agent, captured trajectory from the human in MoCap have just a role as penalty values in Q learning. And final output trajectory from learning agent is a mixture of each state in Q-value table which is initially from each time step of path planned trajectory for MiniHubo by RRT. Since the RRT trajectory was calculated for MiniHubo to reach a desired goal position, the output trajectory from our learning agent which is combination of each state also could make it possible for MiniHubo to reach a target object. Therefore, there was no need to worry about different kinematic structure and different length of the arm links between the human and MiniHubo.

In our experiments for this paper, motions were captured and collected from only one person who performed the given tasks in the MoCap system, on the assumption that human subjects share very similar motion patterns under similar mission and goal positions. This assumption will be tested and further analyzed in future studies. Optimizing between motion patterns from several different human subjects and its combination with dynamics of humanoids robot can give us many future directions for this study.

VI. Experimental Result

Q values in learning agent were updated until every value in Q value table became converged. Q values were updated according to Equation 1 and P which is a penalty value included both energy consumption (Section IV) and difference from human's natural pattern (Section V) to mix two different desired features. In our experiment, an iteration of 85500 updates was required to meet this criterion. For learning rate and discounting factor, 0.5 was assigned for both factors and this made a proposed agent has moderate dependency on recent information and future rewards. Since different weighting factors could be assigned to each penalty values, we could get several different kinds of output trajectories from learning agent with a given set of input RRT trajectories. We produced one output trajectory which minimized an estimated amount of torque consumption for each joint of MiniHubo and another trajectory which had the highest priority on mimicking human's natural motion. Then, a trajectory which tries to satisfy both goals above was produced. Table below shows estimates of energy consumption from each produced trajectory. Dividing sum of torque values from whole joints of MiniHubo by execution time of corresponding trajectory, average torque values from each trajectory were calculated. This table shows that there was almost 10 percent energy reduction when a learning agent has the highest priority on minimizing torque.

Highest Learning Factor	Average Torque
Mimicking a Human's Motion	0.260Nm
Minimizing Torques	0.232Nm
Minimizing Torques while Mimicking a Human's Motion	0.237Nm

Figure 12 and 13 demonstrate movements of MiniHubo at the first case and the last case of table above. In Figure 12, MiniHubo implements an output trajectory which was produced when differences from a human's natural motion had the highest weight for penalty values in Q learning process. Like a motion captured trajectory in Figure 11, MiniHubo starts bending both arms at the same time(Green Line) and the tip of the arm reached each side of box diagonally with the shortest path(Red Line). In Figure 13, a penalty value for torque consumption had the higher weight than penalty value for differences from a human's motion during Q learning stage. Since minimizing energy consumption was emphasized in generating a trajectory, an output trajectory has a different pattern compared to a motion captured trajectory. MiniHubo bent each arms at different time step(Green Line) and did not move lower arms diagonally to each side of box(Red Line). Rather than moving its arms diagonally like captured trajectory, MiniHubo tried to reduce the angle of the shoulder roll joints first, then rotated the shoulder pitch joints just enough to reach the box. This movement resulted in reduction of torque in shoulder roll joints of MiniHubo. Since both arms were kept close to torso of MiniHubo, there was not much torque consumption for the shoulder roll joint while lifting arms. In case of trajectory from captured motion, additional torque was spent in the shoulder roll joint since both arms had non-zero roll angles from torso of MiniHubo during lifting motion.

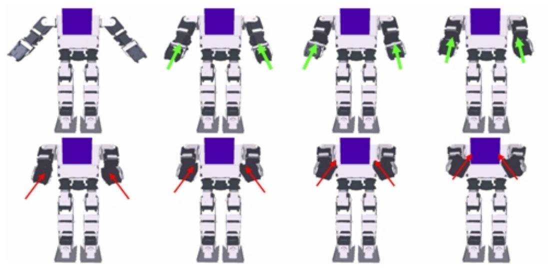


Fig. 12. When Mimicking a Human's Motion Had the Highest Priority

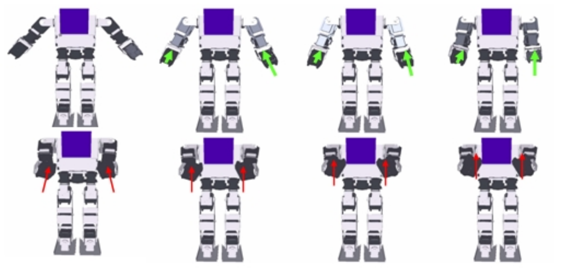


Fig. 13. When Minimizing Torques Had the Highest Priority and Mimicking a Human's Motion Had the Next Highest Priority

VII. Conclusion

This paper presented an approach to optimize humanoid motions. With multiple statically stable input RRT trajectories, a reinforcement learning agent generated a motion trajectory which minimized a set of penalty values. Penalty factors were weighted to generate trajectories that minimized each joint's energy consumption. Weighting also was selected to command the humanoid's joints to move similarly to a person's natural motions. ProPac and SID were used to build a torque model of Mini-Hubo to measure energy costs of each joint. Human motion capture was collected for a given set of tasks and a database was created to similarity cost functions. The converged Q value table generated an optimized trajectory that minimized penalty values. Repeatability was another outcome of this approach; one the Q value table was generated it could be used repeatedly whenever initial kinematic configuration of Mini-Hubo is one of states in the table and a given task are the same. The Q value table could generate an optimized trajectory which has minimum penalty values that the given initial state can achieve.

VIII. Acknowledgments

This project was supported by a Partnerships for International Research and Education (PIRE) grant #0730206, sponsored by the the U.S. National Science Foundation.

References

[1] J. Barraquand and J.C. Latombe, *Robot motion planning: A distributed representation approach*, Int. J. Robot. Res., 10(6):628-649, 1991.
[2] L. Kavraki, et al, *Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces*, IEEE Trans. on Robotics and Automation, 12-4, 566-580, 1996.
[3] S. LaValle and J. Kuffner, *Rapidly-exploring random trees: Progress and prospects*, WAFR, 2000.
[4] D. Berenson, S. S. Srinivasa, D. Ferguson, A. Collet, and J. Kuffner, *Manipulation planning with workspace goal regions*, ICRA, 2009.
[5] D. Berenson, S. S. Srinivasa, D. Ferguson, and J. Kuffner, *Manipulation planning on constraint manifolds*, ICRA, 2009.
[6] Dmitry Berenson, Joel Chestnutt, Siddhartha S. Srinivasa, James J. Kuffner, Satoshi Kagami, *Pose-Constrained Whole-Body Planning using Task Space Region Chains*, Humanoids09, 2009.
[7] Zordan, V.B., Hodgins, J.K., *Tracking and Modifying Upper-body Human Motion Data with Dynamic Simulation*. Computer Animation and Simulation 99, Eurographics, pp 13-22, 1999.
[8] M. Riley, A. Ude, C. Atkeson, *Methods for Motion Generation and Interaction with a Humanoid Robot: Case Studies of Dancing and Catching*, AAAI and CMU Workshop on Interactive Robotics and Entertainment, Pittsburgh, Pennsylvania, April 2000.
[9] Daisuke Matsui, Takashi Minato, Karl F. MacDorman, and Hiroshi Ishiguro, *Generating Natural Motion in an Android by Mapping Human Motion*, Intelligent Robots and Systems(IROS), pp3301-3308, 2005.
[10] M. J. Mataric, *Getting humanoids to move and imitate*, IEEE Intelligent Systems, vol. 15, no. 4, pp. 18-24, 2000.
[11] Lige ZHANG, Qiang HUANG, Shusheng LV, You SHI, Zhijie WANG, and Ali Raza JAFRI, *Humanoid Motion Design Considering Rhythm Based on Human Motion Capture*, IEEE/RSJ IROS, 2006.
[12] Nancy Pollard, Jessica K Hodgins, M.J. Riley, and Chris Atkeson, *Adapting human motion for the control of a humanoid robot*, IEEE International Conference on Robotics and Automation, 2002
[13] L. Guilamo, J. Kuffner, K. Nishiwaki, and S. Kagami, *Manipulability optimization for trajectory generation*, IEEE ICRA, 2006.
[14] A. Safonova, N. Pollard, and J. Hodgins, *Optimizing human motion for the control of a humanoid robot*, Applied Mathematics and Applications of Mathematics, 2003.
[15] Wael Suleiman, Eiichi Yoshida, Fumio Kanehiro, Jean-Paul Laumond and Andre Monin, *On Human Motion Imitation by Humanoid Robot*, International Conference on Robotics and Automation, 2008.
[16] Wael Suleiman, Eiichi Yoshida, Jean-Paul Laumond and Andre Monin, *On Humanoid Motion Optimization*, IEEE-RAS 7th International Conference on Humanoid Robots, Pittsburgh, PA, USA, 2007
[17] S. Albrecht, K. Ramrez-Amaro, F. Ruiz-Ugalde, D. Weikersdorfer, M. Leibold, M. Ulbrich and M. Beetz, *Imitating human reaching motions using physically inspired optimization principles*, IEEE-RAS 11th International Conference on Humanoid Robots, 2011
[18] Oussama Khatib, Luis Sentis, and Jae-Heung Park, *A Unified Framework for Whole-Body Humanoid Robot Control With Multiple Constraints and Contacts*, Springer Tracts in Advanced Robotics - STAR Series, European Robotics Symposium, 2008.
[19] Michael J. Gielniak, Andrea Lockerd Thomaz, *Spatiotemporal correspondence as a metric for human-like robot motion*, HRI, 2011
[20] Harry G. Kwatny and Gilmer Blankenship, *Nonlinear Control and Analytical Mechanics: A Computational Approach (Control Engineering)* , Birkhauser Boston, 1 edition, 2000
[21] Robert Ellenberg, David Grunberg, Paul Y. Oh", Youngmoo Kim, *Using Miniature Humanoids as Surrogate Research Platforms*, International Conference on Humanoid Robots, 2009
[22] Rosen Diankov and James Kuffner. *OpenRAVE: A Planning Architecture for Autonomous Robotics*, tech. report CMU-RI-TR-08-34, Robotics Institute, Carnegie Mellon University, July, 2008
[23] Watkins and Dayan, C.J.C.H., *Q-learning.Machine Learning*, ISBN : 8:279-292, 1992
[24] Dmitry Berenson and Siddhartha Srinivasa and James Kuffner, *Task Space Regions: A Framework for Pose-Constrained Manipulation Planning*, International Journal of Robotics Research, 2011
[25] Ray G. Burdett, Gary S. Skrinar, and Sheldon R. Simon, *Comparison of Mechanical Work and Metabolic Energy Consumption During Normal Gait*, Journal of Orthopaedic Research, vol. 1, pp:63-72, Raven Press, New York, 1983