Humanoid Throwing: Design of Collision-Free Trajectories with Sparse Reachable Maps

Daniel M. Lofaro[1], Robert Ellenberg[2], Paul Oh[2], and Jun-Ho Oh[3]
Electrical Engineering[1] and Mechanical Engineering[2] Dept., Drexel University, Philadelphia PA, USA
Mechanical Engineering[3] Dept., Korean Advanced Institute of Science and Technology, Daejeon, S. Korea
dml46@drexel.edu, rwe24@drexel.edu, paul@coe.drexel.edu, jhoh@kaist.ac.kr

***Abstract*— This work shows a method of creating trajectories to achieve end-effector velocity control for high degree of freedom position controlled, high-gain, robots. The focus of this work is throwing an object. It is shown that the full reachable area of the end-effector does not need to be known to achieve the desired velocity when a good collision model of the robot is available. The end-effector velocity (magnitude and direction) is specified as well as a duration of this velocity. A sparse map of reachable end-effector positions in free space and the corresponding poses in joint space is created using random sampling in joint space and forward kinematics. The desired trajectory in free space is placed within the sparse map with the first point of the trajectory being a known pose from the original sparse map. The Jacobian Transpose Controller method of inverse kinematics is then used to find the subsequent points in the trajectory. Each pose in the trajectory is checked against the collision model to guarantee no self-collisions. This method was tested on the 130cm tall full size humanoid Jaemi Hubo and its virtual representation.**

I. INTRODUCTION

This work focuses on creating and testing valid trajectories for high degree of freedom (DOF), high-gain, position controlled mechanisms that results in the desired end-effector velocity. Throwing and hitting are examples of end-effector velocity control. The goal is to have the end-effector moving at a specific rate in a specific direction. It is also a task that demands whole-body coordination. When the arm moves quickly, as in the case of pitching, such upper-body motions, if not coordinated with the lower-body, can cause the humanoid to lose balance. The overarching goal of this work is to create stable whole-body motions that reliably moves the end-effector at the desired velocity while retaining stability. Fig. 1 shows this overarching goal as an example of a stable human like pose when preparing to throw a ball. The focus of this work is the first required step for throwing with a high gain position controlled robot; throwing an object without self-collision with sparse knowledge of the full reachable area of the robot.

The resulting system is capable of creating trajectories for overhand and underhand throwing motions. This work illustrates the trajectory-based approach with underhand throwing, but can also be applied to overhand cases.

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

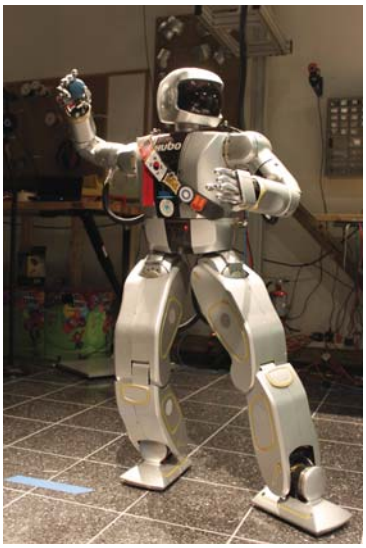

Fig. 1. Jaemi Hubo (Hubo KHR-4) demonstrating the initial pose for the overarching goal of creating a full body, human-like, stable, motion for throwing.

The location in space where the desired end-effector velocity occurs is important in instances such as tennis, ping-pong, and other hitting activities where the end-effector does not control the object throughout the entire motion. Throwing is an example of when the end-effector's velocity holds a higher priority over the position.

Mechanisms with only a single degree of freedom are restricted to throwing in a plane. 2-DOF mechanisms are able to throw in R^3 space with the correct kinematic structure. Such a mechanism can choose its release point or its end-effector velocity but not both. Mechanisms containing 3 or more DOF with the correct kinematic structure are able to throw in R^3 and choose both the release point and the end-effector velocity simultaneously.

Low degree of freedom throwing machines/robots are common. Typical throwing robots have between one and three degrees of freedom (DOF) [1]–[5]. All of these mechanisms are limited to throwing in a plane. Sentoo et al. [6] achieved an end-effector velocity of 6.0 m/s and can throw in R^3 space using it's Barret Technology Inc 4-DOF arm with a 360^o rotation base yaw actuator. These low degree of freedom throwing robots are either physically attached/planted

to the mechanical ground or have a base that is significantly more massive then the arm.

Haddadin et al. [7] used their 7-DOF arm and a 6-DOF force torque sensor with standard feedback methods to dribble a basket ball. In addition Zhikun et al. [8] used reinforcement learning to teach their 7-DOF planted robot arm to play ping-pong. Likewise Schaal et al. [9] taught their high degree of freedom (30-DOF) humanoid to hit a tennis ball using an on-line special statistical learning methods. Visual feedback was used in the basketball throwing robot by Hu et al. [10] achieving accuracy of 99%. All of the latter robots were fixed to the ground to guarantee stability.

Kim et al. [11], [12] takes the research to the next level with finding optimal overhand and sidearm throwing motions for a high degree of freedom humanoid computer model. The model consists of 55-DOF and is not fixed to mechanical ground or a massive base. Motor torques are then calculated to create both sidearm and overhand throws that continuously satisfies the zero-moment-point stability criteria [13].

The highly articulated 40-DOF full size humanoid Jaemi Hubo KHR-4 (Fig. 1) is the platform focused on in this work. Jaemi Hubo is a high-gain, position-controlled biped humanoid weighing 37 kg and standing 130 cm tall. It is designed and made by Dr. Jun-Ho Oh Director of the Hubo Lab at the Korean Advanced Institute of Science and Technology (KAIST). Jaemi has been located at the Drexel Autonomous Systems Lab (DASL) since Fall 2008. DASL has extensive experience with the Jaemi Hubo KHR-4 platform in key areas needed to complete this work. Balancing was explored when developing a real-time zero moment point (ZMP) preview control system for stable walking [14]. A full-scale safe testing environment designed for experiments with Jaemi Hubo was created using DASL's Systems Integrated Sensor Test Rig (SISTR) [15]. Additionally all algorithms are able to be tested on miniature and virtual versions of Jaemi Hubo prior to testing on the full-size humanoid through the creation of a surrogate testing platform for humanoids [16].

The goal of this work is to show the creation of collision free trajectories for end-effector velocity control, the first step in our overarching goal of creating a system with the ability to throw objects and retain balance. Towards this, Section II-A will discuss the paper's method of detecting self collisions. Section II-B describes the creation of the robot's sparse reachable map (SRM), a map in R^3 of reachable points. Section II-C shows the creation of a throwing trajectory in R^3 and placing it withing the robot's reachable area using the SRM. Section II-D explains the inverse kinematics used to convert the throwing trajectory in R^3 to joint space for this high degree of freedom humanoid. Section II-E describes the creation of the setup trajectory from the initial pose of the robot to the starting pose of the throwing trajectory using a variant of trapezoidal motion control to keep within the actuators' physical limitations. Section III features experiments to demonstrate the successful execution of this paper's goal of throwing. Section V concludes the paper and comments on future work.

II. METHODOLOGY

To create a valid throwing trajectory for a high-DOF, high-gain, position controlled robot, a desired line in R^3 in the direction of the desired velocity must be created. Each point in the line is temporally separated by the robot's command period T_r. All points in this line must be reachable. Each point in the line must have poses that do not create a self-collision. A valid throwing trajectory is created when the latter criteria are met.

A. Self-Collision Detection

Self-collision is an important when dealing with a high DOF robot. Unwanted self-collisions can cause permanent damage to the physical and electrical hardware as well as causing the robot not to complete the given task.

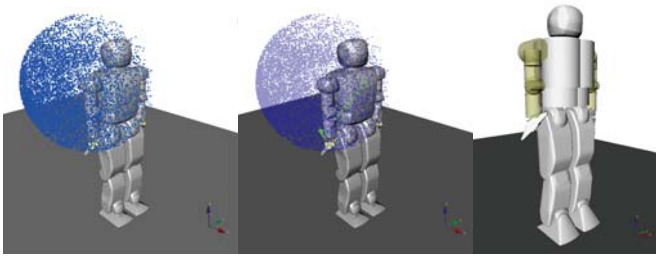

Fig. 2. OpenRAVE model of Hubo KHR-4. Left: Model with SRM of right arm. Center: SRM (blue) with setup and velocity phase trajectories (green) Right: Collision Geometry

To aid in the detection of self-collisions a detailed model of the Hubo KHR-4 was made in the widely used open-source robot simulation environment OpenRAVE [17]. The model was created by exporting the three dimensional schematics that the physical robot was created with, to a format that OpenRAVE can use. This was done in order to ensure an accurate and detailed model. For these experiments we needed the external boundaries only; the internal geometry was replaced with a simplistic representation. The external shell is the only part now visible, see Fig 2 (Center). The Proximity Query Package (PQP) was used to detect collisions between any two pieces of the robot's external shell. Due to the high polygon count of the external shell the computation time of detecting a collision was on the magnitude of seconds. It is advantageous to reduce this time if the system is to run live on the robot. Computation time is decreased significantly when boundary/collision geometries are simplified due to the lower polygon count. The collision geometries were further simplified to decrease computation time by making them primitives such as spheres, cylinder and boxes, see Fig 2 (Right).

Joint limitations are added to the model to mimic the physical robot. The model can be commanded the same configurations as the physical robot. A pose is commanded to the model, PQP searches for any collisions. With the simplified collision geometry self-collisions are detected on the order of milliseconds. If there are no collisions then the pose can be applied to the physical robot. A 5% increase

in volume between the simplified collision geometry and the high polygon geometry was added to ensure all of the physical robot's movements will not collide due to minor calibration errors.

B. Reachable Area

The desired end-effector velocity must be achieved with all joint limits and self-collision constraints satisfied at all times. Typical methods of determining reachability is to move each joint through its full range of motion for each DOF [18], [19]. Due to the high DOF of the Hubo KHR-4 this method is not desirable. A sampling method described in this work is similar to Geraerts et al. [20]. It was used to accommodate the high DOF system. Both active and static joints must be defined to calculate the reachable area of a manipulator at a discrete time N. The static joints are assumed to hold a fixed position at time step N. Active joints are free to move to any position as long as it satisfies the joint angle limitations and does not create a self-collision. A uniform random number generator is used to assign each active joint with an angle in joint space. Each random angle assigned is within the valid range of motion of the respective joint. The self-collision model described in Section II-A is used to determine the self-collision status with the randomly assigned joint angles. If there is no self-collision the end-effector position and transformation matrix T are calculated using forward kinematics.

$$\chi_i = \begin{bmatrix} R_i & \Gamma_i \\ 0 & 1 \end{bmatrix} \tag{1}$$

$$T = [\chi_1 \cdot \chi_2 \cdot \ldots \cdot \chi_n] \tag{2}$$

where χ_i is the transformation between joint $i-1$ and i, R_i is the rotation of joint i with respect to joint $i-1$ and Γ_i is the translation of joint i with respect to joint $i-1$, and n is the number of joints in the kinematic chain.

The end-effector position and the joint angles used are recorded. This process is repeated multiple times to form a sparse representation of reachable end-effector positions in R^3 and the corresponding joint angles in joint space. The resulting representation is called the Sparse Reachable Map (SRM). Fig. 3 shows a cross section of the SRM about the right shoulder between -0.40 m to 0.40 m on X, -0.40 m to 0.40 m on Z, and -0.21 to -0.22 m on Y. The blue points show valid end-effector locations with known kinematic solution in joint space. Fig. 2 shows the SRM of the entire right arm. The SRM is used to calculate valid movement trajectories.

C. Trajectory Generation

An end-effector velocity, $\vec{V}_e$, is chosen based on target location, the well known equations of projectile motion, and the required velocity duration t_e. $\vec{V}_e$ must be held for a time span of t_e. The release point must be within the time span t_e. The magnitude of the velocity in the direction of $\vec{V}_e$ immediately preceding time span t_e must be less than or

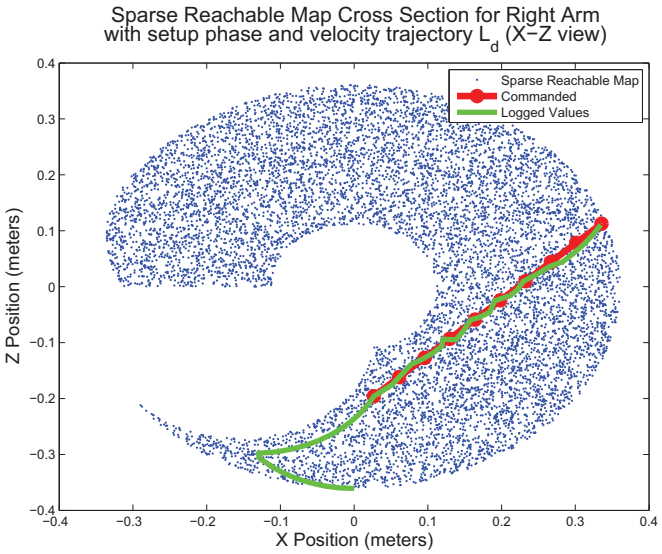

Fig. 3. Cross section of the SRM about the right shoulder between -0.40 m to 0.40 m on X, -0.40 m to 0.40 m on Z, and -0.21 to -0.22 m on Y. (Blue) show valid end-effector locations with known kinematic solution in joint space. (Red) Commanded right arm end-effector position in R^3. (Green) The logged joint space values converted to R^3 using forward kinematics.

equal to the magnitude of $\vec{V}_e$ during t_e. t_e must be an integer multiple of the robot's actuator command period T_r.

A line $\vec{l}_d$ in R^3 that passes through (X_0, Y_0, Z_0) in the direction of $\vec{V}_e$ is created. $\vec{L}_d$ is the discrete representation of $\vec{l}_d$. Each point in $\vec{L}_d$, (X_0, Y_0, Z_0), (X_1, Y_1, Z_1) $\cdots$ (X_n, Y_n, Z_n), are separated by a time span T_r.

The desired velocity is defined as

$$\vec{V}_d = [V_x\hat{i}, V_y\hat{j}, V_z\hat{k}] \tag{3}$$

The line $\vec{L}_d(n)$ is defined as

$$\vec{L}_d(n) = [X_n\hat{i}, Y_n\hat{j}, Z_n\hat{k}] \tag{4}$$

where n is the current zero based time step index value for the time span t_e. The change in $\vec{L}_d$ between time step 0 and n must be equal to our desired velocity $\vec{V}_d$.

$$\frac{\Delta\vec{L}_d|_0^n}{n \cdot T_r} = \vec{V}_d \tag{5}$$

thus

$$\vec{V}_d = \frac{\vec{L}_d(n) - \vec{L}_d(0)}{n \cdot T_r} \tag{6}$$

The line $\vec{L}_d$ at time step n can now be defined in terms of $\vec{V}_d$, T_r, the origin $\vec{L}_d(0)$, and the current zero based time step index value n.

$$\vec{L}_d(n) = n \cdot T_r \cdot \vec{V}_d + \vec{L}_d(0) \tag{7}$$

where

$$\vec{L}_d(0) = [X_0, Y_0, Z_0] \tag{8}$$

The line $\vec{L}_d$ is the trajectory the robot's end-effector must follow during the time span t_e. The starting point $\vec{L}_d(0)$ must be found so that $\vec{L}_d$ is within the reachable area. $\vec{L}_d(0)$ is set to a random starting points chosen within the SRM.

$$\vec{L}_d(0) \in SRM \tag{9}$$

All subsequent points in $\vec{L}_d$ must fall within some Euclidean distance d from any point in SRM. If one of the points in $\vec{L}_d$ fails this criteria a new random point is chosen for $\vec{L}_d(0)$ and the process is repeated.

Once an $\vec{L}_d$ is found that fits the above criteria the inverse kinematic solution must be found for each point and checked for reachability. Smaller values of d will increase the probability $\vec{L}_d$ is within the reachable area defined in the SRM however more iterations will be required to find a valid $\vec{L}_d$. Larger values of d will decrease the number of iterations needed to find a valid $\vec{L}_d$ however the probability of $\vec{L}_d$ being in the reachable area is decreased. In addition larger values of d decreases the system's ability to properly map near sharp edges in the SRM. Increasing the number of samples in the SRM will allow for larger values for d.

D. Inverse Kinematics

The trajectory $\vec{L}_d$ has one point with a known kinematic solution in R^3 and in joint space, $\vec{L}_d(0)$. The joint space kinematic solutions for points $\vec{L}_d(1) \rightarrow \vec{L}_d(n)$ are unknown. Mapping the robot's configuration $\vec{q} \in Q$ to the desired end-effector goal $\vec{x}_g \in X$, where Q is the robot's configuration space and X is in R^3, is done using Jacobian Transpose Controller used by Weghe et al. [21]. Weghe shows the Jacobian as a linear map from the tangent space of Q to X and is expressed as

$$\dot{\vec{x}} = J\dot{\vec{q}} \tag{10}$$

The Jacobian Transpose method is used because of the high DOF of the Hubo KHR-4. Under the assumption of an obstacle-free environment the Jacobian Transpose Controller is guaranteed to reach the goal. A proof is shown by Wolovich et al. [22].

To drive the manipulator from its current position $\vec{x}$ to the goal positions $\vec{x}_g$ the error $\vec{e}$ is computed and the control law is formed.

$$\vec{e} = \vec{x}_g - \vec{x} \tag{11}$$

$$\dot{\vec{q}} = kJ^T\vec{e} \tag{12}$$

where k is a positive gain and self-collisions are ignored. The instantaneous motion of the end-effector is given by

$$\dot{\vec{x}} = J\dot{\vec{q}} = J(kJ^T\vec{e}) \tag{13}$$

The final pose $\vec{q}$ for our goal position $\vec{x}_g$ can now be found.

The Jacobian Transpose method works best when there is a small difference between the current position $\vec{x}$ and the goal position $\vec{x_g}$. $\vec{L}_d(0)$ is known both in X and in Q and is the starting point.

$$\vec{x} = \vec{L}_d(0) \tag{14}$$

$$\vec{q_0} = SRM\left(\vec{L}_d(0)\right) \tag{15}$$

The goal position $\vec{x}_g$ is set to the next point in $\vec{L}_d$

$$\vec{x}_g = \vec{L}_d(1) \tag{16}$$

The pose $\vec{q}_1$ can now be calculated

$$\vec{q}_1 = \vec{q}_0 + \dot{\vec{q}}_0 = \vec{q}_0 + kJ^T\vec{e}|_{\vec{x}}^{\vec{x}_g} \tag{17}$$

where $\vec{x}_g = \vec{L}_d(0)$ and $\vec{x} = \vec{L}_d(1)$. $\vec{L}_d(1)$ is now known both in X and in Q. Now $\vec{x} = \vec{L}_d(1)$ and the process is repeated until all points in $\vec{L}_d$ are known both in X and Q.

E. On-Line Trapezoidal Motion Profile

The robot's starting position $\vec{x}_0$ is not guaranteed to be the same as the first point in the velocity trajectory $\vec{L}_d$. To avoid over large accelerations when giving this step input from $\vec{x}_0$ to $\vec{L}_d(0)$ an on-line trapezoidal motion profile (TMP) was used to generate joint space commands with the desired limited angular acceleration and velocity. The TMP was only active during the setup phase where the robot's end-effector moves from $\vec{x}_0$ to $L_d(0)$. This is because the TMP's inherent nature has the potential to adversely effect the desired velocity in R^3 under high angular velocity and acceleration conditions in joint space.

The TMP was designed to limit the applied angular velocity and acceleration in joint space and to prevent over-current/torque. An important advantage over simply limiting output velocity and acceleration is that the TMP has little to no overshoot. When a clipped and rate-limited velocity profile is integrated, the resulting position trajectory may over or undershoot due to this non-linear system behavior. The TMP accounts for the imposed limits inherently, and will arrive at a static goal without overshoot. Table I describes the three regions that make up the TMP.

TABLE I
TRAPEZOIDAL MOTION PROFILE REGIONS

Region 1	Accelerate at maximum acceleration in direction of goal
Region 2	Achieve and hold maximum velocity
Region 3	Decelerate to zero velocity to reach goal

The area under the velocity trapezoid in region 1-3 is the total displacement achieved by the profile. By shaping this profile based on initial and goal conditions, any goal position can be precisely reached, even if velocity clipping occurs. The shape of the profile can be challenging to identify, since it is not always a trapezoid. For large velocity and acceleration limits and small displacements, the profile will only reach a fraction of maximum velocity, and will be triangular. The varying shape of the profile means that

calculating and storing complete motion profiles for each update may be required. This paper's method removes the need for complete profile generation and storage.

Regions one and two of the velocity profile are bounded by the maximum acceleration, a_m, and maximum velocity, v_m, respectively. In these regions the joint moves towards the goal as fast as the limits allow. In region three the joint has reached a deceleration distance d_s from the goal. It now accelerates at $-a_m$. When the velocity reaches zero, the joint has exactly arrived at the goal position. d_d is the integral of the velocity profile in region three, given by (18).

As long as the distance to the goal d_g and d_s are equal then the controller needs to decelerate at the maximum rate to come to rest at the goal. Conversely, for the current goal distance, there is a critical velocity v_c such that, if the joint began moving at this velocity in the following time-step τ, it could decelerate at a_m to reach the position goal. The controller minimizes the error between v_c and v_0 at each time-step.

Since the joint is moving with velocity v_0 during a current time-step, some initial distance d_i (19) is traveled before the joint can be affected. Defining $\hat{u}$ as the sign of the distance to the goal, v_c is related to d_g and d_i quadratically in (21). This equation assumes simple trapezoidal integration. Solving for v_c using the quadratic formula generally produces complex roots due to the possibility of negative v_0 or d_g. In (22), $v_0 \cdot \hat{u}$ is the current velocity relative to the goal direction, producing a positive term if the signs of both terms match. This result will always produce a real value for v_0 and d_g.

$$d_s = \frac{v_0^2 sign(v_0)}{2a_m} \tag{18}$$

$$d_i = v_0\tau + \frac{v_c - v_0}{2}\tau \tag{19}$$

$$\hat{u} = sign(\theta_g) \tag{20}$$

$$v_c^2 = 2a_m\left(\theta_g - \theta_{err}\right) \tag{21}$$

$$v_c = \hat{u}a_m\left(\sqrt{\frac{a_m\tau^2 - 4\hat{u}v_0\tau + 8|d_g|}{4a_m}} - \frac{\tau}{2}\right) \tag{22}$$

III. EXPERIMENT

The goal of throwing a projectile was set to 3.0 m, the maximum usable distance in the SISTR system as described Ellenberg et al. [15]. The target is placed directly in front of the robot. Using the SRM the release point for the projectile motion calculations was set to the mean of the right arm's reachable area, 0.8m from the ground in z. This resulted in a throwing velocity of 4.9 $\frac{m}{s}$ at $[\Theta, \Psi, \Phi] = [0.0^o, 45.0^o, 3.8^o]$. This velocity was required to be sustained for 0.1 sec due to the speed and accuracy of the gripper's release. This velocity duration and direction forced the trajectory to produce an underhand throwing gesture. The projectile is a standard racquetball measuring 57 mm in diameter and weighting 42.7 g. The light weight racquetball ball was chosen to assist in not causing instability. $\vec{L}_d$ and the setup trajectory are created using the method shown in Section II and following all specified constraints. Fig. 3 shows $\vec{L}_d$ and the setup trajectory plotted within the SRM.

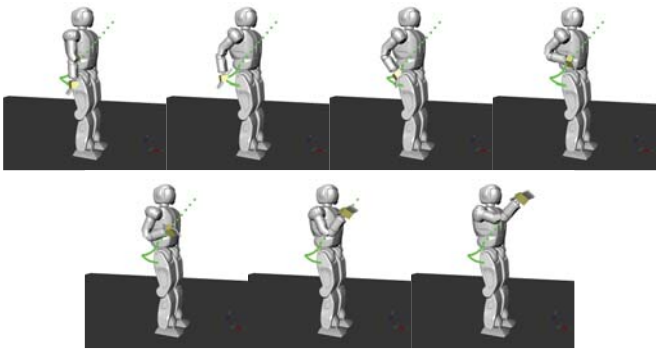

Fig. 4. Jaemi Hubo running throwing trajectory $\vec{L}_d$ immediately after the setup phase is completed. $\vec{L}_d(0)$ is top left. Frames are read left to right and have a Δt of 0.15 sec

The trajectory was run on Jaemi Hubo with a position command period T_r of 0.01 sec. Fig 5 shows the side profile of the Jaemi Hubo successfully running the trajectory. The trajectory shown in Fig 5 is considered an underhand throw, overhand and sidearm throws are also created with this method.

Fig. 5. Jaemi Hubo running throwing trajectory $\vec{L}_d$ immediately after the setup phase is completed. $\vec{L}_d(0)$ is top left. Frames are read left to right and have a Δt of 0.15 sec

During the experiments the actual position of each joint was recorded. The total time from the start of the setup phase to the end of the velocity phase is 0.31 sec.

IV. RESULTS

The system successfully generated and ran valid trajectories. The commanded trajectory produces the desired velocity of 4.9 $\frac{m}{s}$. The joint position of the physical joints

were logged during run-time. The logged and commanded trajectories are seen plotted over the SRM in Fig. 3.

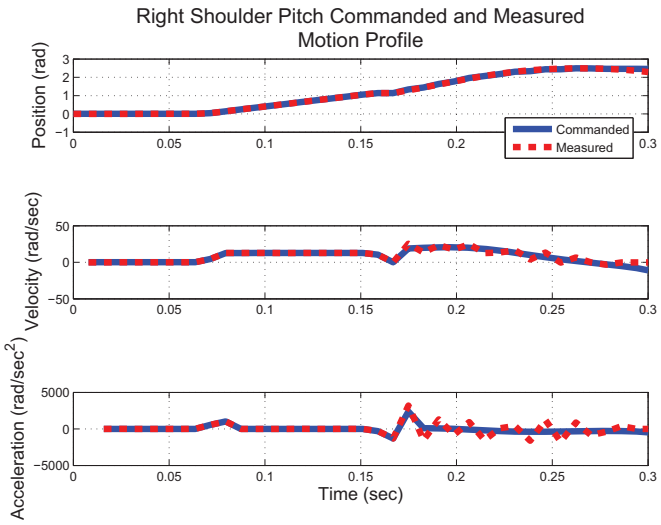

Fig. 6. Right shoulder pitch commanded and measured motion profile; Position (top), Velocity (middle), Acceleration (bottom). This is the result of running the trajectory shown in Fig. 4 and Fig. 5

The end-effector has large accelerations present in the physical system because some of the actuators are commanded with accelerations and torques beyond the capabilities of the physical actuator. The angular velocity and acceleration of the right shoulder pitch joint can be seen in Fig. 6. The large accelerations combined with the inertia of the arm caused the joint to overshoot the commanded position. This caused over-torque on the pitch joint causing the joint to shutdown in slightly less then 10% of the trials.

V. CONCLUSION AND FUTURE WORK

As the results in Fig. 3 in Section IV show the approach presented in this paper was successful for underhand throwing. This approach also preforms overhand throwing if the velocity direction, duration, and magnitude falls within the SRM. While not presented in this paper, results of overhand throwing will be presented in future dissemination of this work. This work is also constructed in such a way that it is easily applicable to other low and high DOF robots.

The paper described a solution that coincides with our future efforts. The next logical step is to incorporate full body motion and balancing to the velocity trajectory calculations to further advance the overarching goal of full body end-effector velocity control.

REFERENCES

[1] N. Kato, K. Matsuda, and T. Nakamura, "Adaptive control for a throwing motion of a 2 dof robot," in *Advanced Motion Control, 1996. AMC '96-MIE. Proceedings., 1996 4th Int'l Workshop on*, vol. 1, Mar 1996, pp. 203 –207 vol.1.

[2] K. M. Lynch and M. T. Mason, "Dynamic nonprehensile manipulation: Controllability, planning, and experiments," *Int'l Journal of Robotics Research*, vol. 18, pp. 64–92, 1997.

[3] W. Mori, J. Ueda, and T. Ogasawara, "1-dof dynamic pitching robot that independently controls velocity, angular velocity, and direction of a ball: Contact models and motion planning," in *Robotics and Automation, 2009. ICRA '09. IEEE Int'l Conference on*, May 2009, pp. 1655 –1661.

[4] T. Nakamura, "Search guided by skill in motion planning using dynamic programming," in *Advanced Motion Control, 1996. AMC '96-MIE. Proceedings., 1996 4th Int'l Workshop on*, vol. 2, Mar 1996, pp. 711 –716 vol.2.

[5] A. Sato, O. Sato, N. Takahashi, and M. Kono, "Trajectory for saving energy of a direct-drive manipulator in throwing motion," *Artificial Life and Robotics*, vol. 11, pp. 61–66, 2007.

[6] T. Senoo, A. Namiki, and M. Ishikawa, "High-speed throwing motion based on kinetic chain approach," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ Int'l Conference on*, Sept 2008, pp. 3206 –3211.

[7] S. Haddadin, K. Krieger, M. Kunze, and A. Albu-Schaffer, "Exploiting potential energy storage for cyclic manipulation: An analysis for elastic dribbling with an anthropomorphic robot," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ Int'l Conference on*, Sept 2011, pp. 1789 –1796.

[8] Z. Wang, C. H. Lampert, K. Mulling, B. Scholkopf, and J. Peters, "Learning anticipation policies for robot table tennis," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ Int'l Conference on*, Dec 2011, pp. 332 –337.

[9] S. Schaal, S. Vijayakumar, S. D'Souza, A. Ijspeert, and J. Nakanishi, "Real-time statistical learning for robotics and human augmentation," in *Int'l Symposium of Robotics Research (ISRR01)*. Springer, 2001, pp. 117–124.

[10] J.-S. Hu, M.-C. Chien, Y.-J. Chang, S.-H. Su, and C.-Y. Kai, "A ball-throwing robot with visual feedback," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ Int'l Conference on*, Oct 2010, pp. 2511 –2512.

[11] J. Kim, "Motion planning of optimal throw for whole-body humanoid," in *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS Int'l Conference on*, Dec 2010, pp. 21 –26.

[12] J. H. and Kim, "Optimization of throwing motion planning for whole-body humanoid mechanism: Sidearm and maximum distance," *Mechanism and Machine Theory*, vol. 46, no. 4, pp. 438 – 453, 2011.

[13] M. Vukobratovic, "How to control artificial anthropomorphic systems," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 3, no. 5, pp. 497 –507, sept. 1973.

[14] Y. Jun, R. Ellenberg, and P. Oh, "Realization of miniature humanoid for obstacle avoidance with real-time zmp preview control used for full-sized humanoid," in *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS Int'l Conference on*, Dec 2010, pp. 46 –51.

[15] R. Ellenberg, R. Sherbert, P. Oh, A. Alspach, R. Gross, and J. Oh, "A common interface for humanoid simulation and hardware," in *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS Int'l Conference on*, Dec 2010, pp. 587 –592.

[16] R. Ellenberg, D. Grunberg, P. Oh, and Y. Kim, "Using miniature humanoids as surrogate research platforms," in *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS Int'l Conference on*, Dec 2009, pp. 175 –180.

[17] R. Diankov, "Automated construction of robotic manipulation programs," Ph.D. dissertation, Carnegie Mellon University, Robotics Institute, Aug 2010.

[18] Z.-Y. Ying, Y.-G. Xi, and Z.-H. Zhang, "Test of the reachability of a robot to an object," in *Robotics and Automation, 1989. Proceedings., 1989 IEEE Int'l Conference on*, May 1989, pp. 490 –494 vol.1.

[19] Z. Xue and R. Dillmann, "Efficient grasp planning with reachability analysis," in *Intelligent Robotics and Applications*, ser. Lecture Notes in Computer Science, H. Liu, H. Ding, Z. Xiong, and X. Zhu, Eds. Springer Berlin / Heidelberg, 2010, vol. 6424, pp. 26–37.

[20] R. Geraerts and M. Overmars, "Reachability analysis of sampling based planners," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE Int'l Conference on*, Apr 2005, pp. 404 – 410.

[21] M. Vande Weghe, D. Ferguson, and S. Srinivasa, "Randomized path planning for redundant manipulators without inverse kinematics," in *Humanoid Robots, 2007 7th IEEE-RAS Int'l Conference on*, 29 2007-dec. 1 2007, pp. 477 –482.

[22] W. A. Wolovich and H. Elliott, "A computational technique for inverse kinematics," in *Decision and Control, 1984. The 23rd IEEE Conference on*, vol. 23, dec. 1984, pp. 1359 –1363.