Digital Servo Calibration and Modeling

Ethan Tira-Thompson
Writing Qualifier, December 15, 2008

Abstract—**The introduction of digital microcontrollers into "hobby" servos opens new doors for consumer and educational robotics. However, the new operational modes, parameters, and sensory feedback also add complexity. This paper will analyze the capabilities of these servos, and describe methods of calibration and motion modeling for accurate planning and control. As much as possible, these methods will avoid use of precision rigs or expensive measurement devices to remain accessible to the classrooms, laboratories, and garages which these servos target.**

I. Survey of Servo Models and Applications

Servo manufacturers currently recognize two major market segments: the industrial servo market and the radio-controlled (RC) servo market. Niche markets are also found in hard drives and optics, which require small scale, precision actuation.

The industrial market caters to large, powerful, expensive motors in factories, airplanes, automobiles, etc. These are often calibrated by the manufacturer, with well defined (and/or customized) operational parameters.

RC servos or "hobbyist" servos are typically intended for use in small remotely controlled devices such as model boats or planes. Due to their standard control interface, low cost, and wide variety of sizes, torques, and materials, these RC servos have become a common component of indoor robotics, particularly among prototype and research oriented devices.

Unfortunately, their original intended use in teleoperated devices with limited range of motion and small number of actuators implies several design choices which constrain autonomous capabilities, and are less than ideal for robotics in general. The lack of middle ground between RC servos and industrial counterparts has driven the creation of projects such as OpenServo [1] and other efforts [2] to replace standard RC servos brains with custom electronics.

The large manufacturers (e.g. Hitec, Futaba) have begun to address the robotics market by "digital servos", which provide better response time and torque management. However, the "digital" refers to a digital microcontroller within the servo, not the communication method, which is still the same analog pulse-width modulation (PWM) interface used by hobby servos.

The restriction to analog communication is at the root of a number of problems:

- Static controller parameters - servos, by their nature, are sent target positions, and the servo determines the torque needed to get there. Different tasks may require different levels of compliance, speed, or other operational modes.
- Unknown position - upon startup, the controlling software has no notion of the servo's current position. Thus, the servo will snap directly to its first position command, resulting in large and sudden motion. This is not an issue for small linkages like a pan/tilt camera or flaps on a model plane, but is a serious problem in robotics where arms and legs may have perilous internal or external collisions.
- Unknown load - users cannot detect collisions or measure weight and pressure distribution. Safety features that prevent burnout or excessive wear cannot be implemented without load feedback.
- Cabling - Each analog signal requires a dedicated signaling wire. A well actuated manipulator will need many wires to carry the control signal out to each individual servo, increasing weight and risk of snagging or pinching wires.

Some of Hitec's HSR series of servos (HSR-5990, HSR-8498) do provide communication options beyond PWM. Designated as Hitec Multi-protocol Interface (HMI), this allows the same cable to be used as a common PWM interface, or a 19200 baud serial-TTL line.

The PWM interface is bidirectional, where a set of special pulse lengths can switch between different pre-defined control parameter sets, or request the servo to send pulse width response to indicate its current position. However, the manufacturer's documentation for the PWM queries notes: "Because the positional feedback [...] operates in conjunction with the PWM control function, there is a chance that a communication error will occur 10% of the time." [3] The PWM read signal also interrupts the torque control (putting the servo into a back-drivable state until the next position command is received), and requires servo controllers to add sensing circuitry to the signaling pin for each servo to read the result. This mode is neither well implemented nor widely available.

The serial protocol, on the other hand, is more promising for flexible digital communication, but it is currently almost entirely unpublicized. For example, most of the servo specification sheets list PWM as the only communication interface, even if it may be noted elsewhere that the servo supports HMI. Documentation is incomplete regarding commands needed for tasks such as setting the PID parameters or changing servo IDs, relegating this functionality to Hitec's Digital Servo Programmer hardware tool.

A more attractive alternative is the Dynamixel line of servos from Robotis, which has a well documented protocol and a 1 megabaud serial TTL connection. These servos provide comparable torque, and provide speed, load, and

position feedback, yet are priced lower than Hitec's. Controller parameters can also be configured. The servos allow 300° range of travel for position control, and can be switched into a "free spin" mode for continuous rotation under speed control, which can pass through the 60° dead zone.

Due to these advantages, the Dynamixel servo was selected for further study as to its feasibility for precision control and dynamic operation.

II. Dynamixel Features

	7 volt	10 volt
Holding Torque	12.0 kgf·cm	16.5 kgf·cm
No-Load Speed	0.269 sec / 60°	0.196 sec / 60°

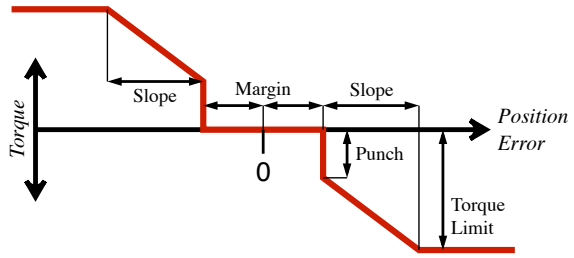
Table 1: published specifications for the AX-12

This paper is based on use of the power supply included with the Bioloid kit, which is rated at 12 volts (measured by the servo's own feedback as 12.3 volts). Different supply voltages will yield different servo responses, an effect which is not quantified here.

The servos provide feedback on position, speed, load, voltage, and temperature. Only the first three parameters will be evaluated in depth in the next section.

The servos' controller exports parameters for torque limit, punch, and separate clockwise/counter-clockwise compliance margin and slope.

The 'margin' parameter controls how much error the servo will allow, the 'slope' specifies the proportional gain to reduce error, and the 'punch' provides an initial kick once the margin is exceeded.



Each servo is identified by a number 0-253. 254 is used as a broadcast address. Commands are sent by writing values into enumerated registers, such as registers 30-31 for the goal position and 32-33 for the moving speed. Thus, these two parameters are 2-byte values, even though the valid range of each is limited to 0-1023.

There is a variety of commands for writing to the registers, providing features such as buffered updates for synchronized motion. To read sensor values back, each servo must be polled individually.

III. Goals and Related Work

The servo's position, speed, and load parameters are of particular interest, but only the position has a defined mapping to physical units from the manufacturer. (0–1023 maps to 0–300°).

The speed parameter presents a minor complication, because it is not consistent between any of commanded speed, physically measured speed, and reported speed, although each exhibits a straightforward linear correlation with the others.

The load parameter is more complicated. An accurate model of the load parameter profile during unloaded motion would be instrumental in allowing us to detect external contacts and forces. This type of sensing has proven very helpful in our experience on the Aibo, where duty cycle information from the servos was a relatively clean signal and good indication of end-effector forces.

Some previous work in improving the accuracy of servos is presented in [4]. However this focuses on optimizing repetitive tasks, whereas this work is focused on generalized modeling the servo itself, applied to novel tasks.

On the other end of spectrum, there is a large body of work (e.g. [5]) regarding controlling direct-drive motors in torque space, but these are typically large, powerful motors with low (or no) gear ratios. The dynamics of these systems degrade with large gear trains found in small servo motors, and does not generalize well to this hardware.

IV. Measurement procedures

To calibrate these parameters into physical units, we will begin by verifying the accuracy of the servo's position for both commands and feedback. This portion of the calibration will be done by visual inspection through a camera facing the servo.

Using the servo's position feedback, we will then calibrate the speeds (commanded and reported) to consistent physical units.

The servo will also be connected to known weights at known lever-arm distances to map its load values to physical torque. After running through a series of motions while recording servo state, a model will be developed to allow measurement of the external load so that its dynamics can be predicted.

A. Position Calibration

The first step for visual calibration of servo position is to identify a marker attached to the servo horn and the center of rotation in the image. We would like to use these methods with other servos, so servo-specific feature detectors are disfavored for identifying the servo position directly. Instead, a piece of cardboard with a thin green stripe is attached to the servo. The position and orientation of this stripe can be easily detected using the DualCoding portion of the Tekkotsu framework. [6]

To avoid errors from inaccurate construction and reduce reliance on precise detection, it is preferable to use a rough calibration target with few assumptions. In particular, we will make no assumptions of the size, position, or orientation of the stripe relative to the servo.

As the servo is moved slowly through a 180° rotation, the camera samples the position of the line detected in each image. The center of rotation can then be determined by iterating through each pair of roughly perpendicular sample lines, finding the point of intersection and the bisector running through it. The common intersection of these bisectors indicates the center of rotation.

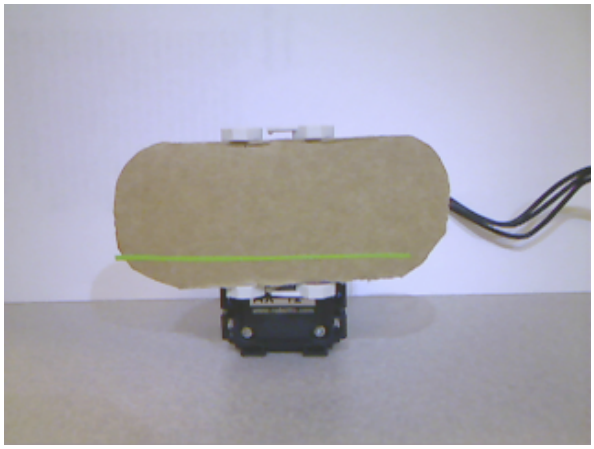


Figure 1: Servo with a haphazard strip of green tape mounted to the horn. It is a good idea to clamp or bolt the servo in place.

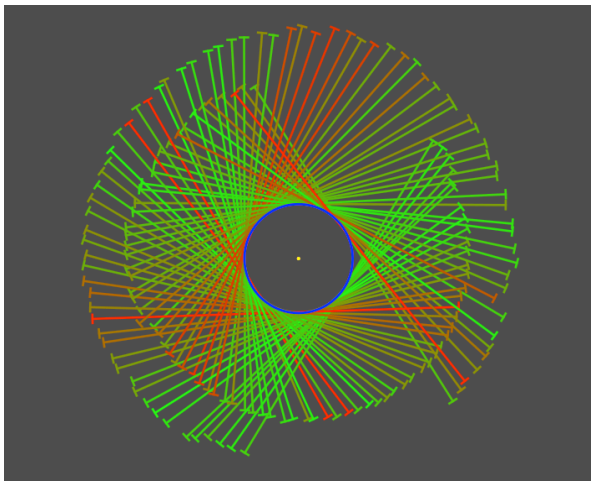


Figure 2: Line samples, green-red color indicates variance from predicted center of rotation (green being lowest variance). Rotation center is predicted with standard deviation (σ) of 0.21 pixels.

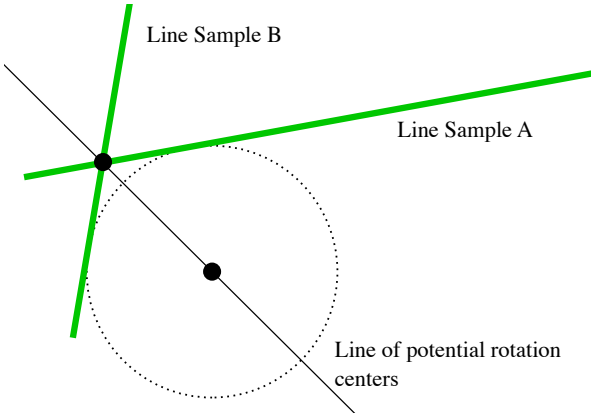


Figure 3: Determination of center of rotation from the common intersection of bisectors.

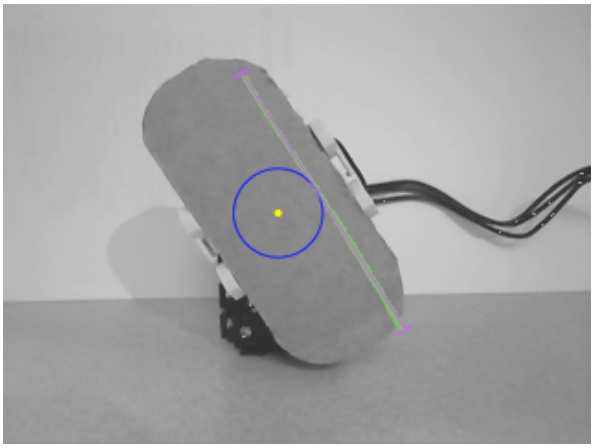


Figure 4: Detected center of rotation (yellow dot)

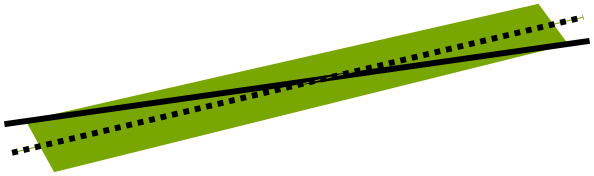


Figure 5: Example of line detector bias due to camera perspective or actual physical tapering of the ends of a line segment

This method is robust against biases in line detection (figure 5), and in practice identifies the center of rotation with a variance of less than half a pixel. Once the center of rotation is identified, we can ensure that future samples are consistent and automatically drop erroneous outliers.

To verify the accuracy of the camera's measurements, a series of positions is sampled and physically measured on each of seven Dynamixel servos. The deviation of the values from the camera measurement and the servo feedback was less than the resolution of the physical measurements (1°).

Additional automated tests using the camera are then performed to verify servo position accuracy across the entire range of motion and measure inter-servo variance.

Servo #	Fit Slope $P_{vis} = x \cdot P_{Fdbk}$	Fit σ (°)	Target σ (°)
1	0.9992	0.42	0.30
2	0.9980	0.60	0.30
3	0.9996	0.44	0.31
4	1.0003	1.63	0.36
5	1.0101	1.33	0.35
6	0.9994	0.52	0.34
7	1.0037	0.48	0.31
Aggregate	1.0014	1.39	0.36
$\mu_x = 1.0015,\ \sigma_x = 0.0042$			

Table 2: Results for each of seven servos, linear fit of position feedback to visually measured position, with standard deviation from that fit, and the deviation of feedback position from target position, in milliradians. The first three servos are from a single Bioloids kit, whereas the remaining 4 were purchased individually.

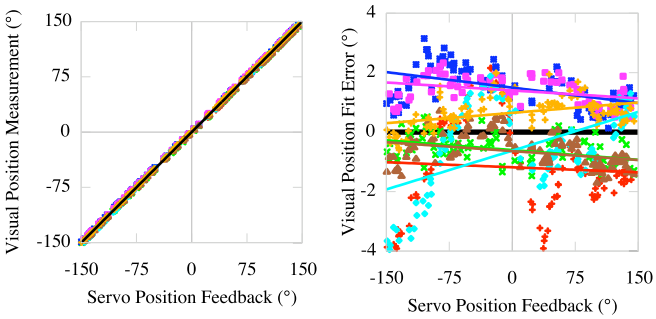


Figure 6: Position fit (left) and residuals (right), all values in degrees.

The linear least squares fit of the aggregate feedback values to visually measured positions indicates that the average uncalibrated servo has a position bias within ±0.21° at the ends of its range of motion. This is less than the verified accuracy of the camera measurements and less than the servo's own resolution, indicating there is no statistical nor practical basis for applying a global positional calibration based on these findings.

The overall standard deviation of position feedback from target position is 0.36°, corresponding to 1.21 units of servo resolution, and demonstrating the servo accurately reaches its target position when unloaded.

B. Speed Calibration

Once the Dynamixel servo is placed into "free spin" mode, it can produce continuous rotation, passing through the 60° dead-zone.

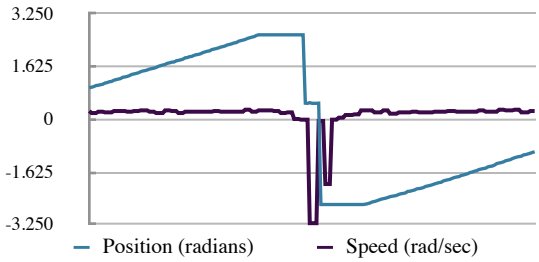


Figure 7: Position and speed during dead-zone traversal by free spin

As the servo passes through this dead-zone, the position feedback reports the closer of the minimum or maximum value, except in the middle of the dead-zone, where it reports a not quite constant mid-range value.

The speed feedback continues reporting values normally until the point at which these mid-range values are returned, where it leaps between a large value in the opposite direction of motion and zero, before resuming normal operation.

Speed feedback is updated at a reduced frequency, on the order of every 130 milliseconds. So, even though it is possible to poll the servo at a much higher rate and get current position data each time, the speed register continues to report the same values between each of the internal updates.

To calibrate the commanded speed and corresponding feedback to physical units, a servo is commanded to a random speed, given 0.4 seconds to accelerate, and then its average feedback and physical displacement are recorded for the following 1.5 seconds. The servo is then brought to a full stop, and the process is repeated. If the servo enters the dead zone, the sample is stopped early; if less than half of the run time has passed, the sample is discarded.

The resulting data for a single servo is shown in figure 8, where the commanded and reported speed are plotted for each sample.

Of note is the vertical jump in requested speeds required to produce a rotation, most likely due to gear friction preventing any motion below a minimum torque. This indicates the servo is not performing integrative control of the speed, as it will not ramp up torque to maintain speed. This can be confirmed by manually holding the servo horn during a slow rotation, where the servo is easily immobilized and does not attempt to increase its torque.

This presents a problem for slow rotations, even after a linear calibration is performed as shown in figure 9. Slow rotations are limited to very low torque, and are not reliably produced as small perturbations in gear friction can halt the servo mid-rotation. This yields the anomaly seen in figure 9, where some samples are stuck at zero radians per second, and other samples are able to move at the intended speed.

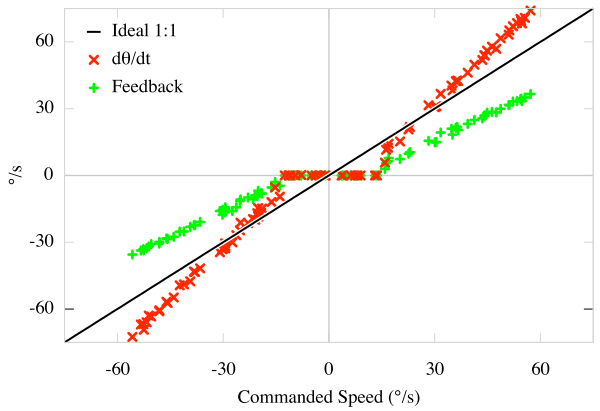


Figure 8: Uncalibrated command vs. produced speeds

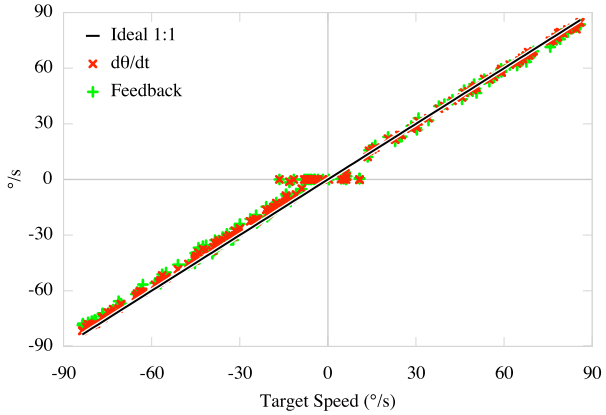


Figure 9: Calibrated command vs. produced speeds

The "punch" servo parameter would ideally be set to counter static gear friction, but trials indicate that none of the controller parameters (slope, punch, margin) affect the free spin mode.

Free spin mode may be more useful for torque control than speed control. For slow rotations which do not need to pass through the dead zone, consecutive high-frequency position commands will more reliably produce a desired speed invariant to gravity and external loads. However, in ideal conditions "free spin" rotations do produce slightly smoother motion, as shown in figure 10.

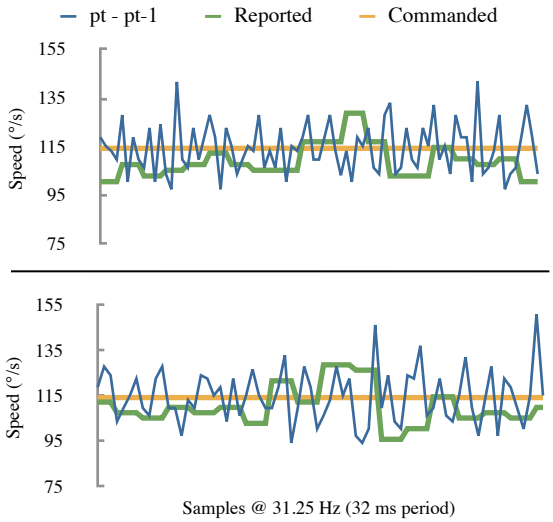


Figure 10. Top: Values following calibrated 115 °/s rotation command. Measured speed σ = 0.119. Bottom: 115 °/s rotation by consecutive position commands, σ = 0.204.

After applying this process to a set of servos, the resulting command calibration parameters are shown in Table 3.

Servo #	Fit: $S_{cmd} = x \cdot S_{tgt} + sign(S_{tgt}) \cdot b$		Target σ (°/sec)
	x	b	
1	0.728	7.506	0.859
2	0.695	9.454	0.917
3	0.671	7.907	0.974
4	0.776	14.381	3.610
5	0.764	15.699	3.839
6	0.742	11.517	1.604
7	0.682	14.954	1.662
Aggregate	0.705	12.834	5.959
μ of Fits:	0.722	11.631	
σ of Fits	0.041	3.438	

Table 3: Speed command calibration results for the same seven servos shown previously in table 2.

Although the aggregate fit provides a ballpark default calibration suitable for general tasks, it is clear from the residuals (Table 3) that more precise speed control can be obtained by applying separate calibration parameters to individual servos, which would reduce the standard deviation of the speed error in some cases by an order of magnitude beyond that provided by the aggregate fit.

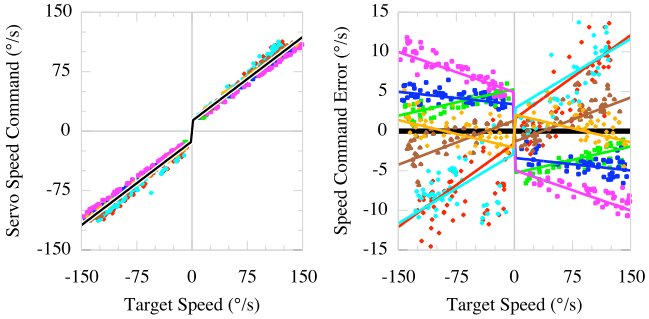


Figure 11: Aggregate speed command fit (left), and residuals (right)

However, the results for speed feedback (Table 4) are much more consistent between servo units, allowing a straightforward global calibration.

Servo #	Fit Slope $S_{actual} = x \cdot S_{Fdbk}$	Fit σ (rad/sec)
1	2.084	1.4152
2	2.066	1.8106
3	2.051	1.6215
4	2.074	1.9653
5	2.068	2.1543
6	2.080	1.9080
7	2.050	1.6730
Aggregate	2.066	1.8965
μ of Fits:	2.0675	
σ of Fits	0.0133	

Table 4: Speed feedback calibration results. Aggregate fit standard deviation is on par with individual unit fit deviations.

C. *Load Calibration*

There are multiple goals for understanding how the servo measures and applies torque:

1) Anticipate and counter positional deflection
2) Break static friction stick/slip for small motions
3) Map between free-spin "speed" and applied torque
4) Gauge external loads, e.g. objects in the gripper

As an example of how these would be useful, imagine launching a snowball. Each snowball is slightly different, and must be weighed to predict its trajectory. Throwing the snowball requires accurate dynamics, and cannot be practiced. (You can only throw a snowball once!)

Alternatively, a walking robot can predict the pressure on each of its legs, and adjust its joint trajectories to account for servo deflection to produce smoother motion.

To measure and model these load related issues, a rigid metal bar is affixed to the servo, providing attachment points at known radii. The servo is clamped to the corner of a table, where the bar and attached masses can hang over the side. (Figure 12)

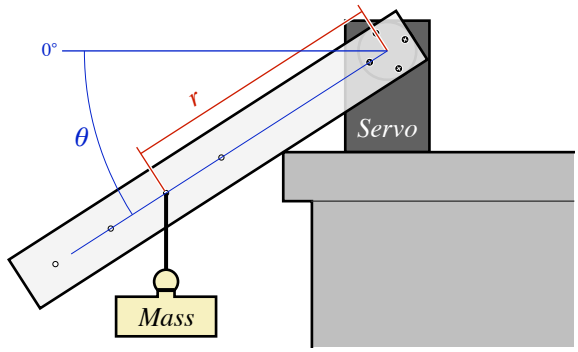


Figure 12: Torque testing rig, servo is clamped to the table.

Several motions are produced, in each case raising the bar to 45° above the horizontal and then lowering back to the table, as shown in figure 13. Only data from sections where the servo is in motion is used, so portions where the servo has not yet lifted from the table and where it is at its apex are dropped.

Tests included a variety of masses, radii, and periods:

- Masses: 0, 113, 200, 313, 500 g; each plus 27 g for the bar and hanger
- Radii: 75, 100, 125, 150 mm
- Periods: 2, 3, 4, 6, 8 seconds

An expected characteristic of a proportional controller is that deflection from target position increases with load, as shown in figure 14. The "required torque" parameter used here only accounts for the gravitational and inertial forces. This exaggerates the plotted spread in figure 14 because it does not account for frictional or latency effects, so points are not at their correct horizontal positions.

We will implicitly model gear friction by considering lifting and lowering motions separately. These models are found to be:

$$defl_{up} = -0.138 \cdot torque + 0.0713 \cdot speed$$
$$defl_{down} = -0.0520 \cdot torque + 0.0777 \cdot speed$$

The $defl_{up}$ equation specifies deflection when moving upward, in opposition to gravity, where the servo must overcome both the external force (gravity and inertia) and

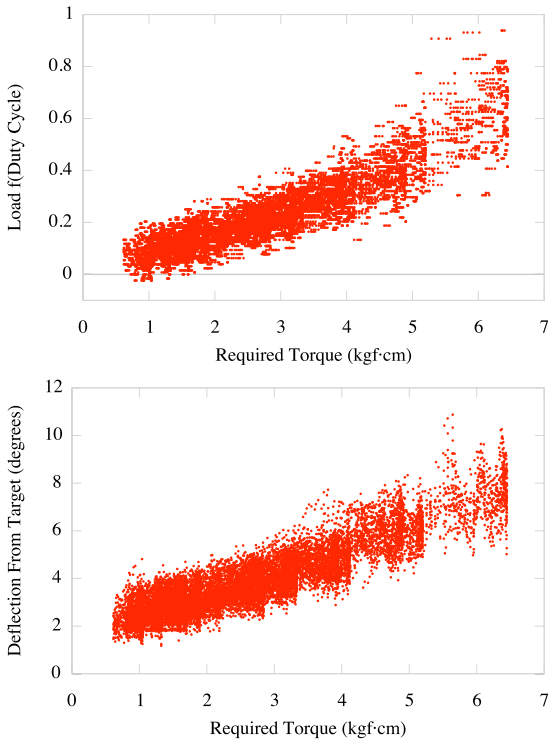


Figure 14: Load (top) and position error (bottom) while lifting a variety of weights over a set of periods and radii.

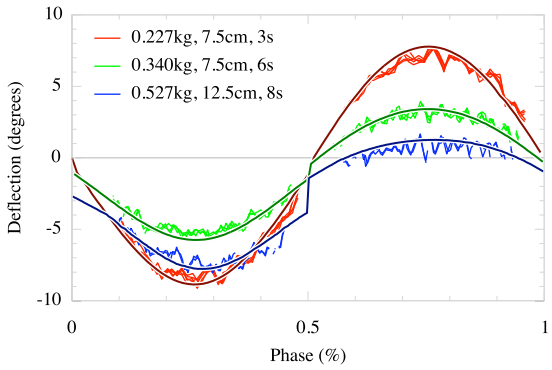


Figure 15: Prediction of deflection (position error) while lifting and lowering weights. Black is predicted, colored lines are sampled.

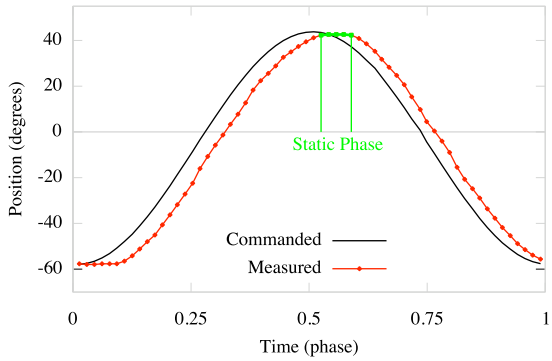


Figure 13: Example measurement indicating position error over time, highlighting the plateau where static gear friction takes hold.

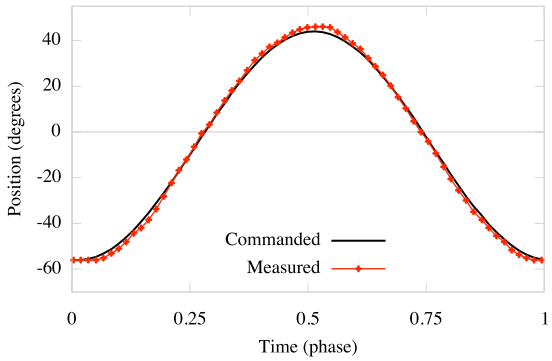


Figure 16: Application of the deflection prediction to the servo commands yields more accurate motion

gear friction. $defl_{down}$ specifies deflection when moving downward, where gear friction opposes the external force and the servo does not need to do as much work. The speed parameter in both cases is similar, and indicates latency of the servo response. (If moving at constant speed with no load, the servo still lags behind the target position.)

This deflection value can be interpreted either as a preemptive error correction measure (an offset applied to commands to yield desired position, given torque and speed) or as a torque control strategy (an offset from current position in order to produce a desired torque and speed).

However, another way to emulate torque control is the aforementioned "free spin" mode, as demonstrated in figure 17.

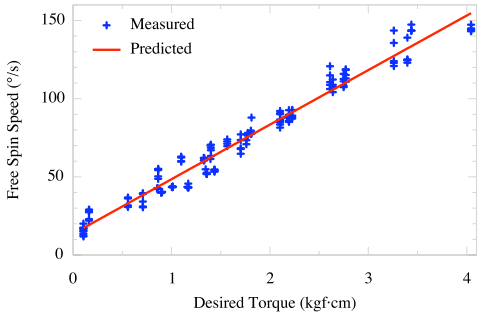


Figure 17: Free Spin Speed as torque control

To produce this data, the servo "speed" parameter was slowly increased until it was able to lift a weight from rest. The speeds shown here are after applying the generic speed calibration in the previous section. The conversion from kilograms-force-centimeters to degrees per second is determined to be a linear factor of approximately 34.9, with some minor per-servo constant offset (here, 13.6°/s) to counter gear friction from using the aggregate calibration.

An unfortunate restriction of this type of control is that the speed control parameter is limited to ±150°/s, which maps to a maximum torque of 4.2 kgf·cm, although the servo is capable of significantly greater torque. The servo will also cap its torque once the specified rotational speed is reached.

The third goal of this section is to classify the bounds of static friction to enable small motions while under load.

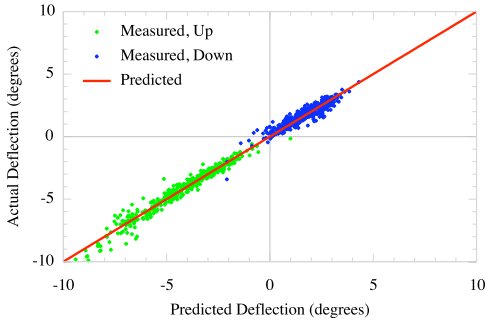


Figure 18: Prediction of minimum position change required to break static friction and initiate motion

This indicates the minimum command offset required to produce motion. Again, since gear friction's influence flips sign when moving with, versus against, an external force, separate models are used for each case:

$$\begin{aligned} defl_{up} &= 0.412 \cdot torque + 0.137 \cdot speed \\ &\quad - 0.0114 \cdot acc - 5.36 \cdot mass \cdot radius \\ defl_{down} &= -0.514 \cdot torque + 0.182 \cdot speed \\ &\quad - 0.0229 \cdot acc - 3.23 \cdot mass \cdot radius \end{aligned}$$

As the final analysis of this section, it would be valuable to use the servo load feedback to measure external loads so that their dynamics can be modeled. For example, the deflection prediction described above depends on knowing the applied torque. So if a robot arm is to lift an object accurately, the mass and position of the object is needed to compute its expected inertia and gravitational force. One way to do this is to lift an object against gravity, and record the servo feedback to determine the external influence.

Dynamixel servos provide load feedback, based on the servo's duty cycle. This gives an indication of how much torque the servo is applying to the gears, but is not a direct measurement of resulting torque at the axle. The load feedback follows the same ~8Hz update as the speed feedback. Unfortunately, this coarse temporal resolution is compounded by significant noise, making load estimates during fast motions untenable.

However, we can instead invert the deflection prediction obtained previously so that we now obtain a torque estimate based on the recorded deflection. Usually we will also know the location of the object, so we can directly solve for the object's mass:

$$\begin{aligned} defl &= a \cdot torque + b \cdot speed \\ \frac{1}{a} \cdot defl - \frac{b}{a} \cdot speed &= torque \\ &= mass \cdot r \cdot g \cdot \cos\theta + mass \cdot r^2 \cdot \ddot{\theta} \\ &= mass \cdot (r \cdot g \cdot \cos\theta + r^2 \cdot \ddot{\theta}) \\ mass &= a' \frac{defl}{r \cdot g \cdot \cos\theta + r^2 \cdot \ddot{\theta}} - b' \frac{speed}{r \cdot g \cdot \cos\theta + r^2 \cdot \ddot{\theta}} \end{aligned}$$

However, the load feedback from the servo does provide some additional information, so providing both the deflection and the load feedback does help reduce noise.

$$mass = \frac{-5.24 \cdot defl + 0.346 \cdot speed - 0.245 \cdot load}{r \cdot g \cdot \cos\theta + r^2 \cdot \ddot{\theta}}$$

The instantaneous estimates over several representative trials is shown in figure 19. Although separate models are used for each of the raising and lowering phases to account for opposing frictional effects, the mass estimates are clumped together when lowering the mass. Because gravity does most of the work in the lowering phase, the servo feedback does not provide much information on its workload. This means the estimates are only useful in the lifting phase, although even then still noisy. However, averaging the estimates over the course of a motion should give an accurate estimate of the mass. (Figure 20)

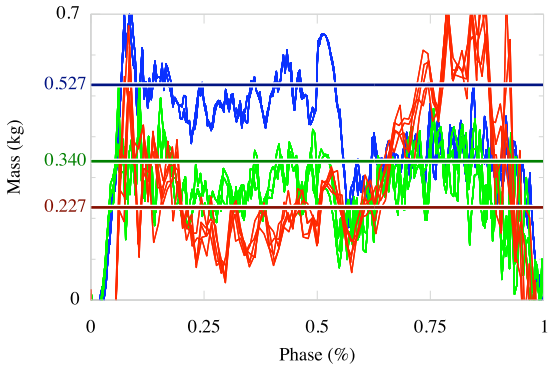


Figure 19: Recovery of attached mass given known radius of revolution. Black lines are the ground truth, colored lines are instantaneous estimates over several runs.

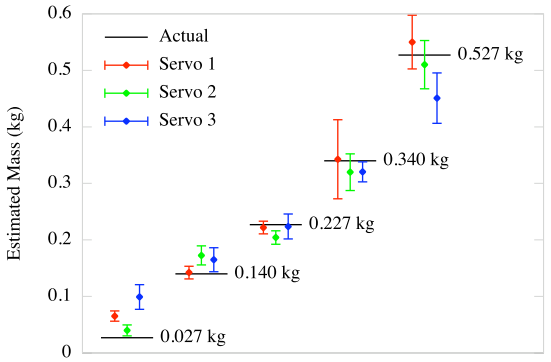


Figure 20: Std. deviations of mass estimates for three servos sharing a global fit, points encompass 20 permutations of speed and radius.

V. Identified Issues

With a 1 megabaud communication line, the servos are capable of sampling position feedback at very high temporal resolution. However, many USB-to-serial chipsets are configured to buffer communication data received from the serial line up to 16 milliseconds, although data sent to the serial line is unbuffered. This configuration limits servo polling to 62.5 Hz. The tests shown here all utilize a single servo at any given time, and were reliably polled at 31.25 Hz.

However, on a hexapod robot with 18 servos, the per-servo polling would drop to only 3.5 Hz. One improvement is to blindly transmit several read requests, spread over the buffer period, and then read the responses en masse when the buffer is flushed. We have been able to reliably poll up to three servos per flushing with this method, thus tripling the poll rate when using multiple servos.

Regarding maximum torque, AX-12 servos are able to hold significantly more than their rated 12 kgf·cm when powered at 12 volts. However, they stall significantly lower, around 8-9 kgf·cm. This means that although a large applied load can be resisted from a given position, if the servo is moved in the direction of the force, it may be unable to later reclaim its original position.

VI. Future Work

Different voltage levels have been noted to change the torque characteristics of these (and other) servos. In practical terms, this means a battery powered robot will perform differently as its battery runs down. However, the influence of voltage has not been examined here, where all tests were done using an external power supply. Similarly, the role of temperature on the motor efficiency has not been examined.

The servo controller parameters have been left at their default values during these tests, but adjusting these parameters may allow more accurate feedback in some situations. For instance, when attempting to measure external load, decreasing the proportional response should allow the servo to deflect further from the target position. This should result in more precision for the corresponding load estimate.

Finally, there is a newly introduced wCK servo series found in the RoboBuilder kits, which offer features similar to the Dynamixel. These were not yet available when this work initiated, but a comparison of the performance of these servos would be valuable.

VII. Conclusions

A methodology for modeling and calibrating key parameters of fully digital servos has been described and demonstrated on a specific model. The servo can be used in each of position, speed, and torque control modes, with calibrated feedback for closed-loop interaction.

The result of this work is that these servos may be better utilized in a variety of robotics applications, fostering better collaboration and reproducibility of results by the use of common, off-the-shelf parts.

VIII. References

[1] http://www.openservo.com/

[2] Wright, C., Johnson, A., Peck, A., McCord, Z., Naaktgeboren, A., Gianfortoni, P., Gonzalez-Rivero, M., Hatton, R., and Choset, H.: Design of a modular snake robot, Intelligent Robots and Systems, 2007

[3] http://www.hitecrobotics.com/manager/control/down_file.php?upFile=HITEC%20HMI%20Robot%20Servo%20General%20informationV1.00_1.pdf&tableName=board_14

[4] F. Lange and G. Hirzinger. Learning Force Control with Position Controlled Robots. In International Conference on Robotics and Automation, Minneapolis, April 1996

[5] C. H. An, C. G. Atkeson, and J. M. Hollerbach. *Model-Based Control of a Robot Manipulator.* The MIT Press, Cambridge, MA, 1998

[6] David S. Touretzky, Neil S. Halelamien, Ethan J. Tira-Thompson, Jordan J. Wales, Kei Usui: Dual-coding representations for robot vision programming in Tekkotsu, Autonomous Robots, Vol. 22, No. 4, pp. 425-435, 2007