Efficient Robotic Control Policies with Joint Coupled Motor Primitives

Ben Webb
Department Of Computer
Science, Colby College
Waterville, ME 04901

Caitrin Eaton
Department Of Computer
Science, Colby College
Waterville, ME 04901

Abstract - **Very few living creatures are capable of fine motor skills in the way humans are. Fine motor skills require precise and efficient control policies that are true to the desired motion. Traditional inverse kinematics based control policies struggle to achieve the computational efficiency required for a task as simple as drawing a square. The goal of this work is to explore the accuracy and computational efficiency of robotic arm control policies required to produce drawing motion. The work sought to elicit relationships between joint trajectories in cartesian space and joint space, as well as introduce positional feedback into the control loop. Each control policy was evaluated on a 4-DOF prototype arm as well as in simulation to evaluate the performance of each solution. The results suggest that there is a recognizable pattern of joint position during these movements. With positional feedback, this pattern could be exploited to create a novel control policy that produces a motion that is smoother and more computationally efficient than traditional inverse kinematics methods without sacrificing precision.**

I. Introduction

Fine motor skills are complex. For a human, even the simple task of drawing a line requires precise control and coordination between joints at the shoulder, elbow, wrist, and fingers on top of any supervisor level processing. The smallest perturbation or error at any of these joints results in a line that isn't straight. Fine motor control isn't a complex task just for humans, robots also struggle with these same constraints, both in motor power and a supervisor system development.

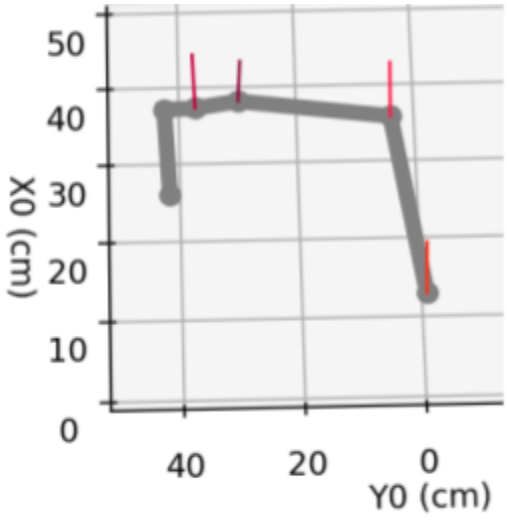
Traditional robotic control policies rely on inverse kinematics which relies on deriving the transformation from joint origin to end effector. This is precise, but humans do not think about a drawing line solely as joint angles, and the configuration of their arm and writing tool are constantly changing. In addition, these methods are computationally expensive, are susceptible to singularities and are not guaranteed to give the most thermodynamically efficient solution. Ijspeert demonstrated that a motion could be understood as control policies with distinct and recognizable submovements. These control policies are primitives: significantly more efficient than the jacobian and have recognizable patterns despite changes in the speed and amplitude of a motion.

These are coined *primitives* because despite change in the amplitude or speed of a motion they retain their shape parameters. The shape parameters are the activity of each joint during each submovement of the larger motion. This was demonstrated to be useful when developing control policies where the underlying primitives of motion are difficult to discern.

The shape parameters used in Ijspeert however were unique for each joint. This was done to reduce training time with each joint converging on its own shape parameter. This work seeks to understand how the coupling of shape parameters affects the discovery and function of motor primitive solutions through the development of a prototype arm with trained primitives in the X and Y directions. Using these trained primitives and closed loop control, the learned lines can be used to write by hand. This solution does not derive primitives from a complex movement. Instead it seeks to derive sagittal and lateral movement primitives that could be combined to create complex movements where the individual primitives are all involved.

II. System Design

A 4-DOF arm using Dynamixel AX-18A motors and a BeagleBone Blue board was built to serve as the testbed for the arm controller. Drawing was chosen as the evaluation for the performance of the primitives. This was done because the solution space for drawing always lies in a single plane despite how many degrees of freedom an arm has. Furthermore, the movements in this plane can exist as two known primitives, one in the X direction and one in the Y direction.



i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	10.5	θ_1
2	0	22	0	θ_2
3	$\pi/2$	22.5	0	θ_3
4	$-\pi/2$	7	-1.5	θ_4
5	0	5	-8.5	0

Diagram 1. *Schematic diagram and modified Denavit-Hartenberg parameters of the 4-DOF arm.*

The kinematics of the arm represent the direct relationship between the arm positions and euclidean space. As can be seen in Diagram 1, θ_1 and θ_2 serve to over articulate control over the axial plane whereas θ_3 and θ_4 control precise movement and height. This was done to intentionally promote role coupling the activity of these two joints during movement. Role coupling refers to two adjacent joints which lie in a single place. Each θ refers to a motor. The position of the motor can be measured and lies in $Z: \{0, 1, 2, ..., 1021, 1022, 1023\}$. For kinematics however, the position of the motor is represented as its radian value with Z: $\{0\} = -\frac{5\pi}{6}$ and $\{1023\} = \frac{5\pi}{6}$.

$$
{}^0_4T = \begin{bmatrix}
-s_4s_{12} + c_3c_4c_{12} & -s_4c_3c_{12} - s_{12}c_4 & -s_3c_3 & 10s_3c_{12} - 5s_4s_{12} + 22c_1 + 5c_3c_4c_{12} + 7c_3c_{12} + 24.5c_{12} \\
s_4c_{12} + s_{12}c_3c_4 & -s_4s_{12}c_3 + c_4c_{12} & -s_3s_{12} & 22s_{12} + 10s_{12}s_3 + 5s_4c_{12} + 5s_{12}c_3c_4 + 7s_{12} + 24.5s_{12} \\
s_3c_4 & -s_3s_4 & c_3 & 5s_3c_4 + 7s_3 - 10c_3 + 10.5 \\
0 & 0 & 0 & 1
\end{bmatrix}
$$

Equation 1.

The forward kinematics of the arm was determined using Equation 1. The transformation matrix is used to translate between spherical and cartesian space. The transformation matrix was additionally used to create an arm simulator in matplotlib that did not communicate with the dynamixels. The simulation was used to evaluate the performance of the solution produced by each control policy on the dynamics of the arm.

$$
J(\theta) = \begin{bmatrix}
-10s_3s_{12} - 5s_4c_{12} - 22s_1 - 5c_3c_4s_{12} - 7c_3s_{12} - 24.5s_{12} & -10s_3s_{12} - 5s_4c_{12} - 5c_3c_4s_{12} - 7c_3s_{12} - 24.5s_{12} & 10c_3c_{12} - 5s_3c_4c_{12} - 7s_3c_{12} & -5c_4s_{12} - 5c_3s_4c_{12} \\
22c_1 + 10s_3c_{12} - 5s_4s_{12} + 5c_{12}c_3c_4 + 7c_{12}c_3 + 24.5c_{12} & 10s_3c_{12} - 5s_4s_{12} + 5c_{12}c_3c_4 + 7c_{12}c_3 + 24.5c_{12} & 10c_3s_{12} + 5c_3c_4c_{12} - 7s_3s_{12} & 5c_4c_{12} - 5s_{12}c_3c_4 \\
0 & 0 & 5c_3c_4 + 7c_3 + 10s_3 & -5s_3s_4
\end{bmatrix}
$$

Equation 2.

Taking the partial derivative of the displacement terms on the right column of Eq. 1 produces the jacobian matrix in Eq. 2. The Jacobian and its inverse are the solution to the partial derivation of the displacement terms in the right-most column of equations in the transformation matrix. They are used to take the derivative of terms between spherical and cartesian transformation allowing control for inverse dynamics. The matrix operations this requires are computationally intensive and produce errors in certain configurations of the arm -- called singularities -- in which the Jacobian's determinant becomes zero. Jacobian-based IK is also not guaranteed to produce efficient, lifelike, or safe solutions, only mathematically correct solutions.

III. System Control and Evaluation

Four control policies were explored: Open Loop Jacobian-based (OLJB), Closed Loop Jacobian-based (CLJB), Open Loop Motor Primitive (OLMP), and Closed Loop Motor Primitive (CLMP). The evaluation of each was done by measuring the performance of each control policy in simulation and on the prototype. The System Control and Evaluation is divided into subsections focusing on the development of features and applications for each control policy. Target trajectories were communicated to for all control policies using a waypoint method input from the supervisor. Each waypoint step communicated a target d_{xyz} step based on the state of the supervisor.

Jacobian-Based Inverse Kinematics

The Inverse Jacobian is the solution to the partial derivation of the direct transformation between joint and cartesian space at all points where the transformation is differentiable. Because the inverse jacobian matrix is too complex to be hard coded, the control policy relied on first evaluating the forward jacobian and then taking the pseudoinverse of the result. For each control step in jacobian based control policies, the forward jacobian had to be evaluated requiring reading of all joint positions. This produced the Open Loop Jacobian-based (OLJB) control policy.

Motor Primitives

Learning by demonstration where the prototype was guided through a trajectory was done to elicit two motor primitives: one for the X or sagittal direction and one for the Y or coronal direction. The elicited motor primitives conformed to the guidelines for motor primitives as follows:

(1) The control policy cannot be time dependent.
(2) The shape parameters of a motor primitive must remain constant across changes in amplitude or speed of the primitive.
(3) The shape parameters of a motor primitive must be distinct and independent of different motor primitives.

Ijspeert used these as standards for motor primitives in order to create a control policy that was flexible to environmental constraints. However, unlike Ijspeert, primitives were not detected as a subsection of a more complex movement. Here, primitives were elicited that could be combined to later form more complex or dynamic movements.

Seeds for the X and Y primitives were then taught through demonstration of multiple lines drawn parallel to the XY axes of the arm kinematics. The training set of data was recorded through passive measurement of joint positions as the end effector of the prototype was guided along lines parallel to the X and Y axes as seen in Figure 1. The relationship between each join and the primitives was further explored through a heatmap looking at covariance.

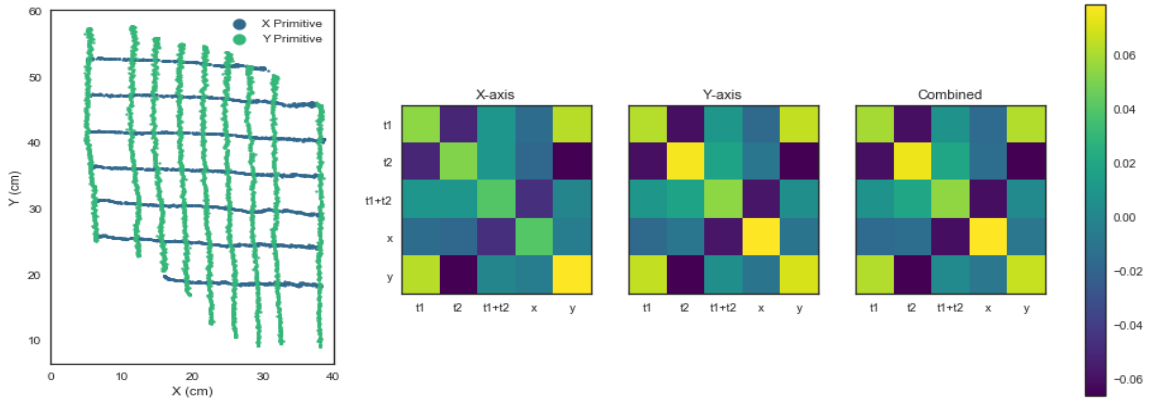


Figure 1. *Training space and covariance matrices of demonstrated movements along the X and Y axes.*

As seen in Figure 1, there is not a significant difference between the normalized covariance matrices of individual primitives or that of the combined data. Neither θ_1 nor θ_2 individually show covariance with X while the sum of the two does. Contrarily, θ_1 and θ_2 both show covariance of equal magnitude but in opposite directions with Y while the sum does not.

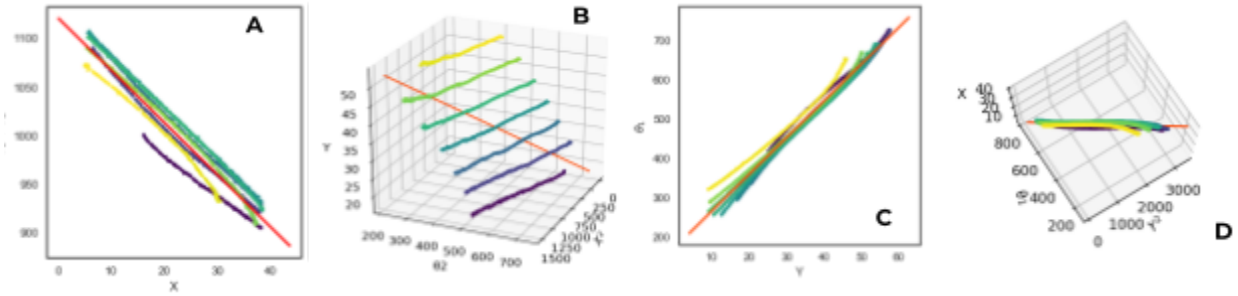


Figure 2. *Linear approximations for θ_1 and θ_2 from offline learning for X and Y ordinal directions. Plots A & B are the derivation with respect to change in X. Plots C & D are the derivation with respect to change in Y.*

The explicit covariate relationship was defined using a linear regression for the primary direction term (Fig. 2 A & C) and a multiple linear regression for the accommodating joint solution (Fig. 2 B & D). There is a linear and proportional change in the $\theta_1 + \theta_2$ with respect to change in X (Fig. 2 A). This is also true for change in θ_1 with respect to change in Y (Fig. 2 C). As can be seen in Fig. 2 B & C, the acceleration of θ_1 and θ_2 with respect to its primitive is linear.

$$A(\theta_1,\theta_2 \sim XY) = \begin{bmatrix} -5.3632658 + 0.271193 * X & 9.3842993 \\ -0.271193 * X & -0.2650688 * Y \end{bmatrix}$$

Equation 3.

Equation 3 shows the partial derivatives with respect with X and Y of the lines of best fit from Fig. 2. Unlike the Jacobian which needs to be evaluated at every measurable timestep, the primitive can approximate each joint's current activity without reading the position of the joint itself. The activity of each joint is a product of the cartesian position of the end effector which can be approximated after the first joint transformation. The communication demand between the motors as well as the computational complexity of the solution is significantly lower for the primitive solution which is calculated 27 times faster allowing for the time of each control step to

surround around motor communication time. This produced the Open Loop Motor Primitive (OLMP) control policy.

Closed Loop Control

Closed loop control was added to control policies to minimize X and Y errors between the control policy trajectory and the target trajectory. The recorded XY position was determined by reading the positions of the motor and Eq. 1. The error in position was measured by comparing the record XY position with that of the trajectory target position. The movement was modified using online learning in the form of convergent PID tuning. The PID controller was formatted to measure and correct error in X and Y independently, making use of the primitives to correct error generated by the motor control loop. The residual error measured in the PID controller represents the error at each timestep, allowing for the controller to also account for integrated position and derivative acceleration of X and Y errors. PID weights were adjusted using binary search to minimize the MSE of the end effector and its goal position over the course of a set of movements.

$$\mathrm{PID}_{\mathrm{measured}} = \begin{bmatrix} d_x & p_x & a_x \\ d_y & p_y & a_y \end{bmatrix}$$

Equation 4.

$$\mathrm{PID}_{\mathrm{weight}} = \begin{bmatrix} d_{weightX} & d_{weightY} \\ p_{weightX} & p_{weightY} \\ a_{weightX} & a_{weightY} \end{bmatrix}$$

Equation 5.

Equations 4 and 5 are the two components of the PID controller. Error correction was measured and solved separately for the X and Y. Equation 4 represents the measured PID error values. At each control step, PID error was measured directly producing p_{xy} error terms, as a function of the rate change of the p_{xy} terms in the d_{xy} terms, and finally as a function of the acceleration of the p_{xy} error term in a_{ax}. The measured errors were then adjusted by each term's respective weight in determining the PID output.

$$\begin{bmatrix} S_{xX} & S_{xY} \\ S_{yX} & S_{yY} \end{bmatrix} = \begin{bmatrix} d_x & p_x & a_x \\ d_y & p_y & a_y \end{bmatrix} \cdot \begin{bmatrix} 0.068316 & 0.068658 \\ 0.053957 & 0.053991 \\ 0.098652 & 0.086783 \end{bmatrix}$$

$$S_{xX}, S_{yY} = \mathrm{PID}_{\mathrm{measured}} \cdot \mathrm{PID}_{\mathrm{weight}}$$

Equation 6.

S_{xX} and S_{yY} are the solutions with X and Y PID weights corresponding to measured X and Y errors. S_{xY} and S_{yX} are the solutions with X and Y PID weights solving the pairwise opposite of measured X and Y errors. As can be seen in Equation 6, the weights for the controller were similar between X and Y directions suggesting it might be possible to simplify this controller further. Using the PID controller with the jacobian and motor primitives produces two novel control policies: the Closed Loop Jacobian-based (CLJB) and the Closed Loop Motor Primitive (CLMP) control policy.

Network Applications

In order to quickly acquire skills consisting of sets of motor primitives, as well as improve the precision at which the trajectory of the end effector was measured, computer vision using OpenCV and libfreenect was used to track the drawing produced by the arm with computer vision. The BBB does not have the processing power to perform real time video analysis so the supervisor state was migrated to function over computer networks. Different network protocols between the supervisor and the arm controller were evaluated seeking a protocol that produced the more accurate trajectory. The protocols evaluated were TCP and UDP sockets which vary from one another in their error detection, latency, and connection states.

For a closed loop control policy utilizing the transformation matrix, the PID controller still operates locally on the beaglebone. For a computer vision based closed loop control policy, the PID controller operates on the networked supervisor. Keeping PID local to the arm controller allowed for online demonstration of movements in real-time while building computer vision based PID allows for more direct measurement of the attractor landscape. The computer vision based PID could also be used to track a line being drawn by a human and communicate the direction and magnitude of that line to the BBB. This means that the supervisor state communicates any drawn trajectory in over the network to the arm controller which can locally mimic the motion.

IV. Results

This work sought to explore the implications of allowing the shape parameter of a primitive to control the activity of more than one joint over the course of the primitive. The first two joints of the arm were primarily focused on as joints responsible for the creation of trajectories in the XY-plane. The testing set of movements was chosen to be one that was novel to the control policies with shape parameters of known control policies. Multiple epochs of the test set were run for each control policy across the entire solution space. The origin of each test set was translated to a single origin to compare the mean trajectories across different control policies. Each movement within the set of testing movements was smoothed using a sinusoidal activation function to minimize rapid changes in the dynamics of the arm. The performance of the arm was measured by taking the mean squared error of the target position and recorded position of each control policy through the testing epoch. The recorded position was determined using the integer position of the motor resulting in more vibration recorded than produced by the end effector.

Open Loop Jacobian-Based Inverse Kinematics

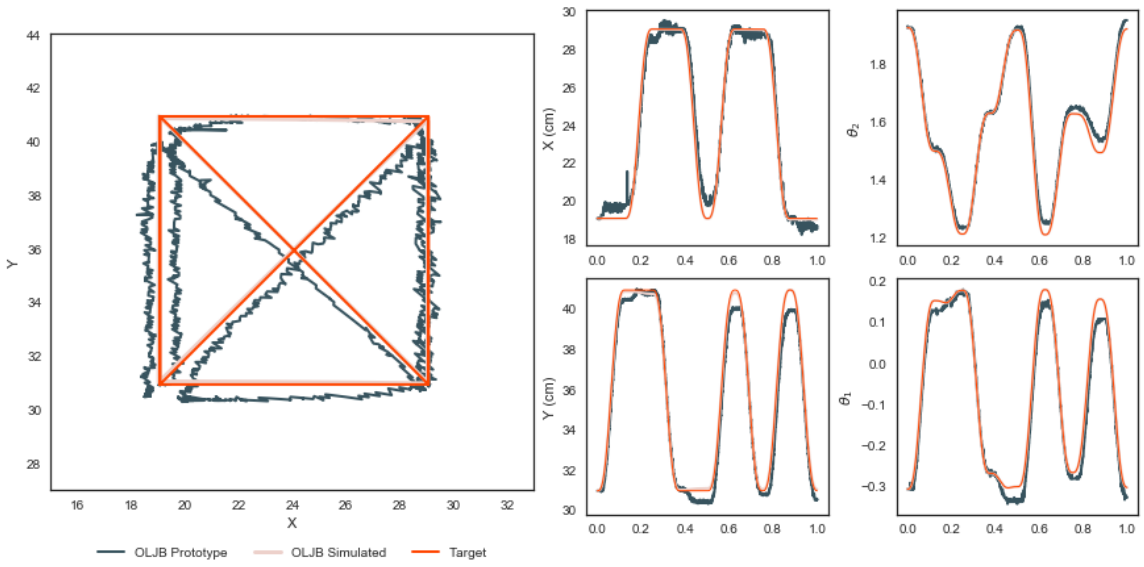


Figure 3. *Trajectory of open loop inverse kinematics control policy (MSE X:0.421 cm & Y: 0.513 cm).*

The OLJB control policy produced a trajectory that maintained proportionality between movement in the X and Y axes. The control policy has the advantage of representing a finite solution for all poses the arm may be in and thus represents a global solution in a finite solution space. Movement in the Y direction involved more total change in the positions of the first two joints. The centripetal forces generated by faster joint acceleration in the Y direction generated more error in the X direction. Additionally, this prototype as well as all prototype trajectories will display an apparent vibration that is not present in the motion itself. The vibration is a product of finite motor position resolution resulting in an imprecise transformation of the end effector position; such is the motivation for pursuing additional controllers for PID.

Open Loop Motor Primitives

The determined solutions for the primitives do not explicitly couple the activity of the joints: instead the X primitive represents a solution that couples the activity while the Y primitive does not. The coupling of activity can be seen in identical terms shared between the solutions for joints θ_1 and θ_2 in Eq. 2 and 3.

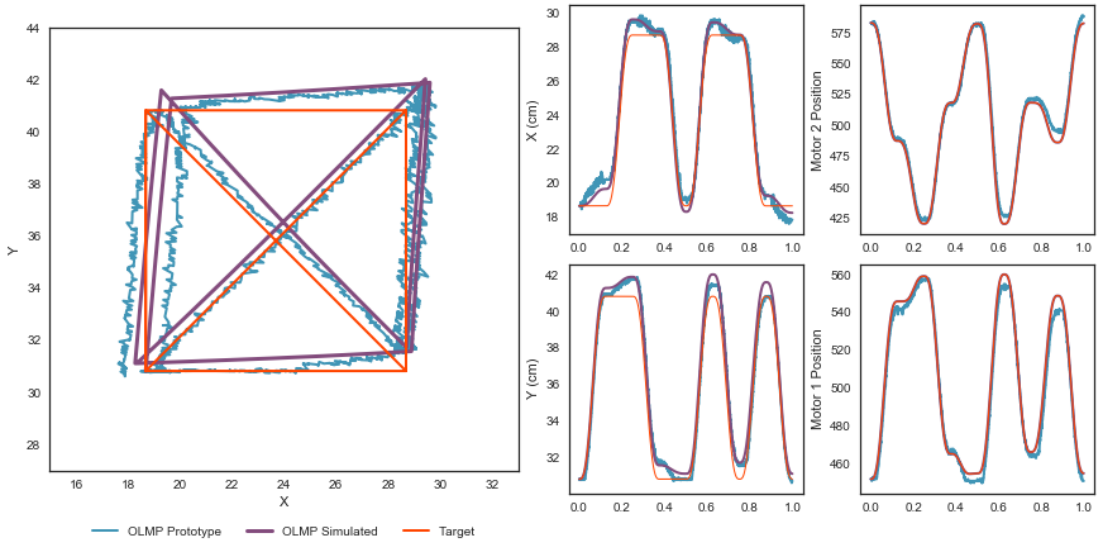
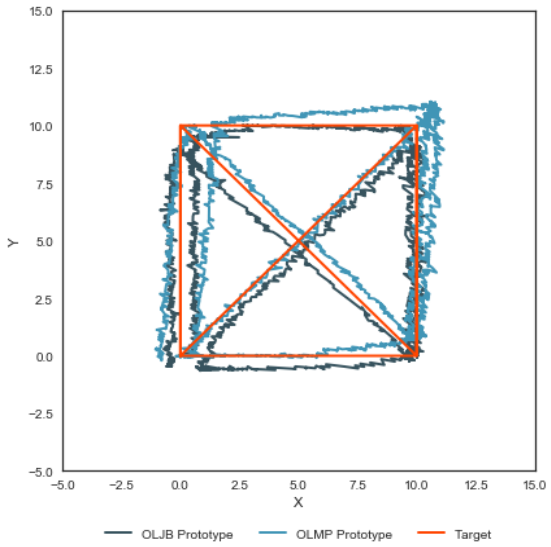


Figure 4. *Trajectory of open loop inverse kinematics control policy (MSE X:0.628 cm & Y: 0.539 cm).*

The OLMP trajectory produced by the set of motor primitives demonstrates that a control policy based on the set of joints with grouped and isolated activity successfully can form a framework for trajectory planning. The trajectory of the simulated and prototype are less precise than the OPJB. This is expected because primitives do not represent a precise transformation, instead they reduce both precision and computational cost of the control policy by representing the proportional activity of the first two joint activities in relation to a change in X or Y.



As seen in Fig. 3 & 4 θ_1 θ_2, the joint activity profiles of the OLJB and OLMP are relatively similar. It appears the primitive produced a solution with the same shape as the jacobian based control policy. However, there is a skew in the Y direction during movement in the X which suggests that the activity profiles are not perfectly orthogonal to one another. That being said, the motor primitive is capable of combining multiple primitives to form a composite motion that is constant in the shape parameters of the motion.

Figure 5. *Performance of open loop motor primitive and inverse kinematics control policies.*

Similar to the OLJB, the vibration in the recorded trajectory is primarily a product of movement in the Y direction. The OLJB prototype outperformed the simulation and prototype trajectories of the motor primitive. It did so by producing a more precise trajectory with significantly increased computational overhead. The duration of the OLJB control step was twice that of the primitive control step with a frequency of 50 Hz.

Closed Loop Jacobian-Based Inverse Kinematics

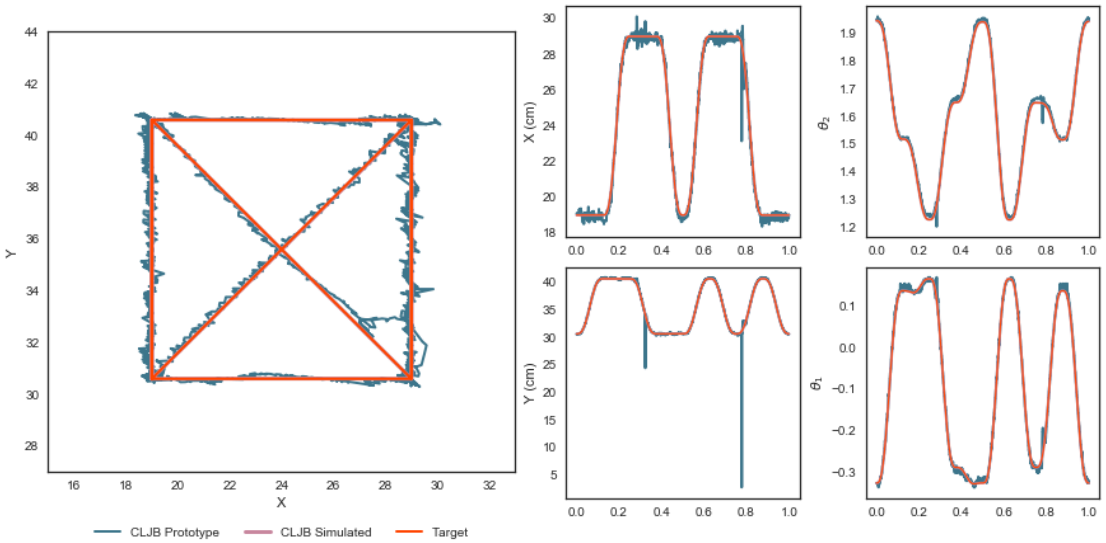


Figure 4. *Trajectory of closed loop inverse kinematics control policy (MSE X:0.204 cm & Y: 0.163 cm).*

The CLJB control policy seen in Fig. 4 demonstrates near perfect trajectory planning and performance. The simulated trajectory is nearly indiscernible from the target trajectory. The prototype also does a significantly better job following the trajectory but is still identifiable due to the jittery vibration seen in all prototype trajectories. On occasion, a communication error would result in the motor position of the fourth motor being read incorrectly. This prompted the PID controller to respond to computer noise instead of measured error as can be seen in the instantaneous change in measured Y of Fig. 4. To amend this, an observation window of 10 was applied to the measured X and Y positions of the Supervisor. The result of which can be seen in Figure 5.

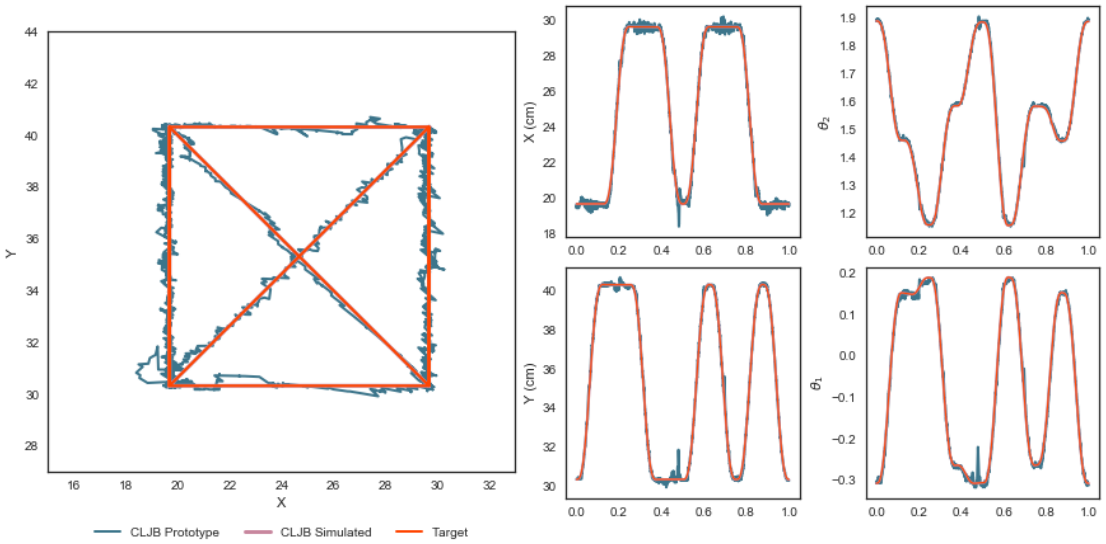


Figure 5. *Trajectory of closed loop with observation window inverse kinematics control policy (MSE X:0.153 cm & Y: 0.120 cm).*

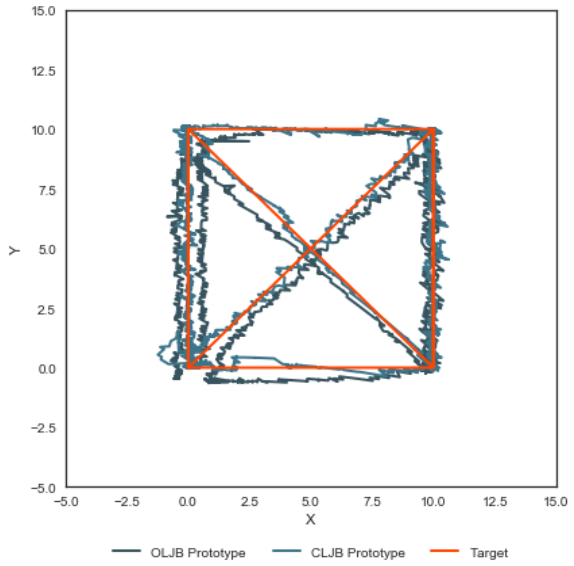


Figure 6. *Performance of open and closed loop inverse kinematics control policies.*

As can be seen in Fig. 5, the window effectively reduced the impact of erroneous readings while maintaing . Any inaccuracies seen in the OLJB simulation were efficiently rectified by the PID controller. The CLJB trajectory significantly outperformed the OLJB. This was expected because the PID controller contributed a fraction to the control time step while reducing small errors due to centripetal forces that the Jacobian otherwise did not account for. However, the PID controller, in attempting to correct the vibration in X during movement in Y, tends to create more vibration rather than reduce the vibration.

Closed Loop Motor Primitives

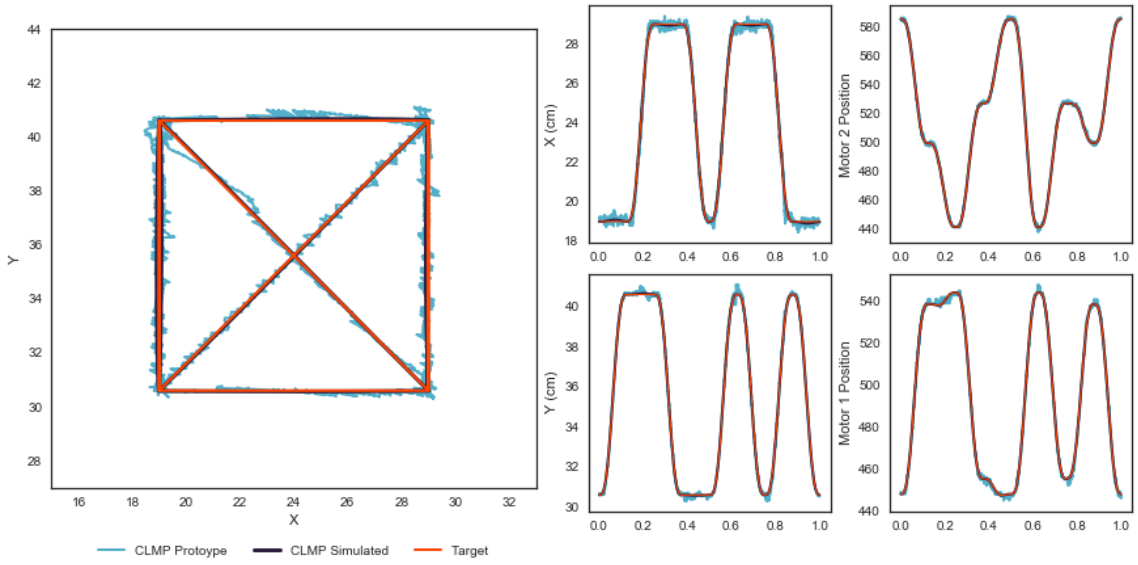


Figure 7. *Trajectory of closed loop motor primitive control policy (MSE X:0.154 cm & Y: 0.097 cm).*

As can be seen from the simulated primitive trajectory as well as the closed loop trajectory, primitives can not only recreate a motion with static shape parameters, they can also be used to correct the errors created by their imprecision. The closed loop control policy is able to modify the activity of the joints within a primitive such that they do not change the shape parameters of the primitive (Figure 5, 6 Motor Position), while more accurately following the target trajectory (Figure 6, Trajectory).

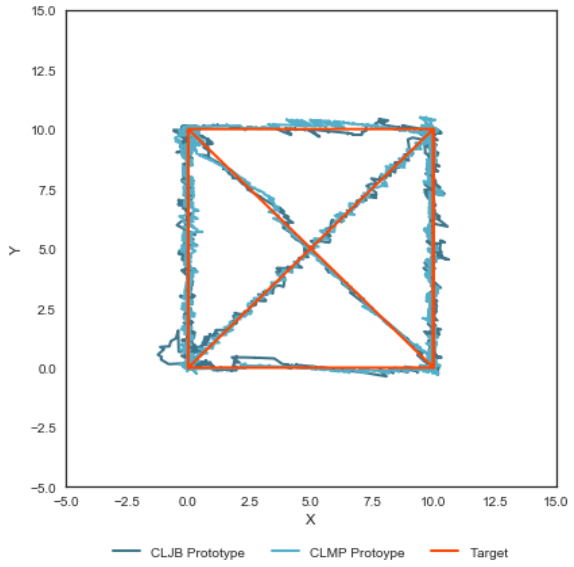


Figure 8. *Performance of closed loop primitive control policy.*

The online learning control policy using the transformation matrix converged quickly with minimal training epochs. Fig. 8 shows the performance of both closed loop control policies. Because the motor primitive has a smaller time step, the PID controller is able to respond and correct for error faster. This results in a smoother overall trajectory that is more precise than the inverse kinematics solution.

Network Applications

In order to allow higher level processing for the arm, a Network enabled supervisor was built. This supervisor could communicate movements in the XYZ planes to the controller on the arm as well as the control policy the arm should use to navigate cartesian space. Two different network protocols were investigated to explore the feasibility of communication over a wireless network for robotic control.

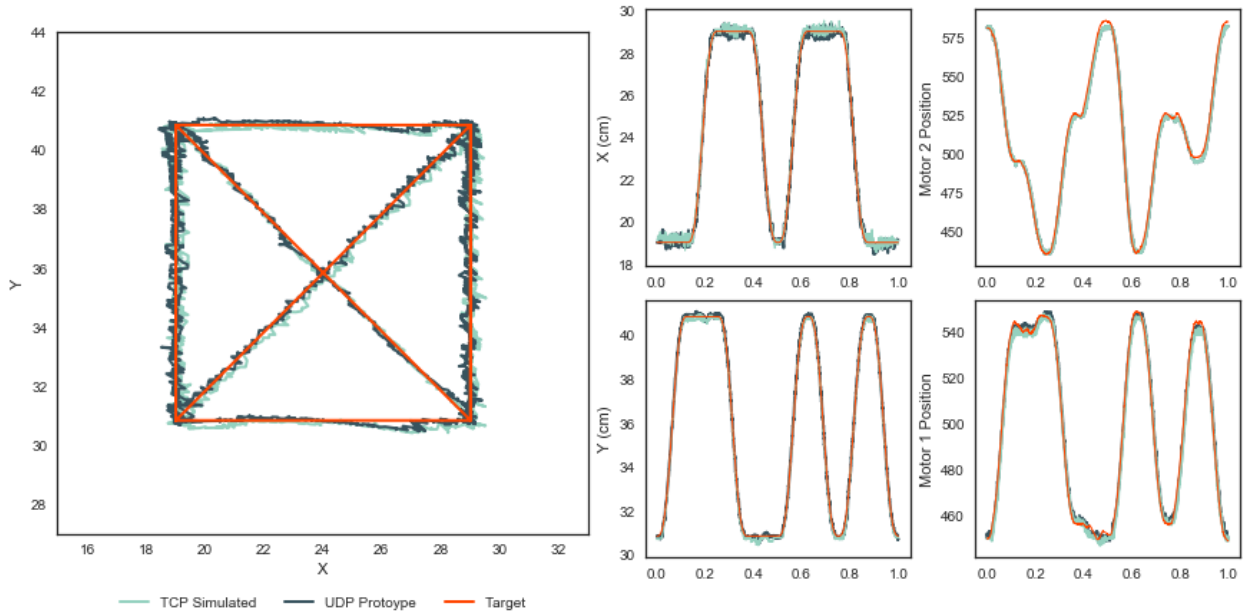


Figure 9. *Trajectory of TCP and UDP sockets (MSE TCP X:0.128 cm & Y: 0.077 cm, UDP X:0.126 cm & Y: 0.077 cm).*

As can be seen in Fig. 9, communication over TCP and UDP sockets performed well. This took advantage of the BBB hosting its own WLAN network. This did take use of WLAN hardware and software, but on an empty network. On a busy home network, round trip time between packets was far too variable for effective

communication between the networked supervisor and the arm controller. The networked supervisor had to be synced with the arm controller. For the TCP socket, this happened during the handshake the controllers would agree on a control timestep before running either control loop. For the UDP socket, the control step was built to match that of the control time step on the arm and ran without waiting for a response to see if the packets arrived. The implementation of the network interface, however, resulted in a larger control step. This reduced the MSE by reducing the inertia of the arm during a movement.

IV. Conclusion and Future Work

Control Policy		OLJB	OLMP	CLJB	CLMP
Mean Square Error (MSE)	**X**	*0.421 cm*	*0.628 cm*	*0.204 cm*	*0.154 cm*
	Y	*0.513 cm*	*0.539 cm*	*0.163 cm*	*0.097 cm*

Table 1. Mean Square error of measured X and Y at each control step.

The CLMP had the lowest MSE and the second shortest control step of all control policies evaluated. The shorter control step resulted in a smoother motion while the lowest error resulted in the most precise motion. It is a more stable and efficient solution than the jacobian and much more efficiently applicable for a prototype arm. Adding closed loop control significantly reduced the measured error. Closed loop control reduced the measure error for the jacobian and primitive solutions. Both closed loop control policies had an error that was twice that of the control step. That means on average the same was four control steps in position from the target trajectory which is significant considering the precision of readable and writable motor positions.

After training, a significant factor in the perceptive error between the target trajectory and that produced by the arm was the imprecision of readable motor positions for each joint. All implementations of the closed loop control policies however relied on kinematics for positional reading. The noise in position is because the motors only have a resolution of 0.29˚. The vibration in the motor position is 1 step or 0.29˚ above and below the goal trajectory. For any kinematics based solution involving fine motor skills, the motors would require positional feedback improvement and precision. This highlights another significant benefit of the motor primitives over the jacobian: it does not require read motor positions. The current implementation relies on the XY interpretations to happen as a result of forward kinematics but the solutions do not require any known joint positions or degrees of freedom, it only requires an arm with similar proportions and a way to transform the end effector into cartesian space.

This was the primary motivation for exploring computer vision where the computer could be precise in end effector space to the pixel. The prototype holds the writing tool vertically, but this is not how humans hold writing tools, humans rest their arm on a surface to minimize centripetal forces and hold the tip of the writing tool so it is always clearly visible. Unfortunately due to limited time, this implementation of the tracker was not more precise than the end effector position from the kinematics solution. The current solution tracks the line as it is being drawn instead of the tip of the marker as it is drawing. This has the benefit of avoiding orientation issues but can also be obstructed by the arm as it is writing.

However, the computer vision controller is able to trace deliberately oriented writing tools held much less perpendicularly than the prototypes holds a hold. This allowed for real time recording of anything written by a human. Using this, the direction of the line being drawn was recorded as a primitive representation of a character for some of the letters of the english language. These recorded directions could then be communicated over the network enabled supervisor to allow the arm to mimic the recorded trajectories, effectively enabling the arm to write.

V. References

[1] S. Dziemian, W. W. Abbott and A. A. Faisal, "Gaze-based teleprosthetic enables intuitive continuous control of complex robot arm use: Writing & drawing," 2016. . DOI: 10.1109/biorob.2016.7523807

[2] A. J. Ijspeert, J. Nakanishi and S. Schaal, "Learning Rhythmic Movements by Demonstration using Nonlinear Oscillators," 2002.

[3] A. J. Ijspeert *et al*, "Learning Attractor Landscapes for Learning Motor Primitives," 2002.

[4] J. Peters *et al*, "Towards motor skill learning for robotics," in Anonymous Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 469-482.

[5] S. Schaal and C. G. Atkeson, "Constructive Incremental Learning from Only Local Information," *Neural Computation,* vol. 10, *(8),* pp. 2047-2084, 1998.

[6] S. Schaal, A. Ijspeert and A. Billard, "Computational approaches to motor learning by imitation," *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences,* vol. 358, *(1431),* pp. 537-547, 2003.