Throw It - A Serial manipulator that projects a particle to a desired location

Prakash Baskaran[1] Abhijeet Thakan[2] Samruddhi Kadam[3]

***Abstract*— Many fields are being filled by automated machines performing human activities. This paper considers the trajectory planning and motion control problem of a 2 degree of freedom robot manipulator arm. The robot arm is required to operate in a 3D space. The objective is to project a particle (say ball) held in the manipulator arm, into a bin placed randomly in the 3D environment. To achieve this objective, the mathematical model of the system is designed, various controllers are tested on the system using MATLAB and a combination of controllers that work out for the system are obtained**

I. INTRODUCTION

Automation has become quite a common phenomenon now-a-days. The speed at which developed countries are heading towards automation is quite inspiring. This has led to easy life in the developed parts of the world. Luxury has become an essential factor, which was so far just a want, is now turning into a need, due to its easy accessibility. So, to compete with this life style, every home or office space should be accustomed with some automated servants with low cost and high efficiency. That is the motivation for our project. The world is plagued by the issue of trash. There is no doubt that cleanup is time consuming, and in some situations, it reduces work efficiency. By developing an autonomous robot arm that can orient, aim and project trash efficiently into the trash bin, the manpower needed for cleaning can be significantly reduced [1]. With this vision, Throw It! the autonomous trash throwing robot has been discussed in this paper. MATLAB R2017a [2] is used for the mathematical modeling, control system implementation and simulation. The programming algorithm for this robot was also developed with this research work. Several possible control systems were considered and some had to be modified before implementing them onto the system.

A. Related Work

- Serial manipulators have long been at the forefront of industrial automation. One of the main reasons behind this is the ease at which they can be designed and modeled. Most of the intuition in developing the dynamic model and controller for our manipulator came from the Pelican Robot (Fig. 1) developed at the CICESE Research Center in Mexico. It is relatively a simple planar arm with 2 links connected through revolute joints, i.e. it possesses 2 DOF. The links are driven by two electric motors located at the "shoulder" (base) and at the "elbow". This is a direct-drive mechanism, i.e. the axes of the motors are connected directly to the links without gears or belts. Though this robot looks simple, it proves to be quite useful in learning and understanding the math behind the controller design and lays a proper foundation in Robot Control before we jump to complex systems.

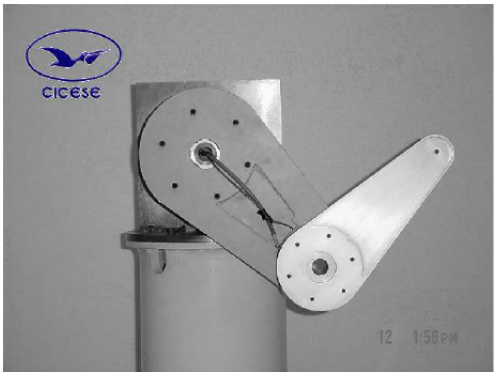

Fig. 1. Pelican Robot, CICESE Robotics Lab

- The robotic arm we have designed is similar to that of an Anthropomorphic Robotic Arm (Fig. 2) but with fewer degrees of freedom. Anthropomorphic arm, as the name suggests, is a robotic arm that resembles a human arm. A typical Anthropomorphic robot possesses 7-DOF including one revolute joint along the z-axis at the base and a spherical joint at the end effector. We have settled for a 2 DOF Anthropomorphic arm to solve our control problem.

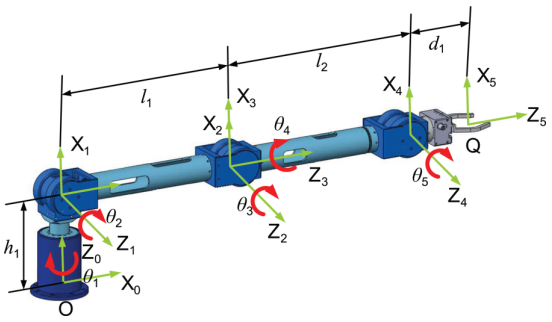

Fig. 2. 5-DOF Anthropomorphic Arm

[1]Prakash Baskaran is a Graduate Student of the Robotics Engineering Department, Worcester Polytechnic Institute, Worcester, MA 01609, USA. e-mail: pbaskaran@wpi.edu

[2]Abhijeet Thakan is a Graduate Student of the Robotics Engineering Department, Worcester Polytechnic Institute, Worcester, MA 01609, USA. e-mail: amthakan@wpi.edu

[3]Samruddhi Kadam is a Graduate Student of the Robotics Engineering Department, Worcester Polytechnic Institute, Worcester, MA 01609, USA. e-mail: spkadam@wpi.edu

B. Problem Description

The objective of this project is to design a 2 DOF serial manipulator (Fig. 3) that can launch a particle (ball) at a desired velocity and angle, so that it follows a projectile motion and lands on a user specified destination.

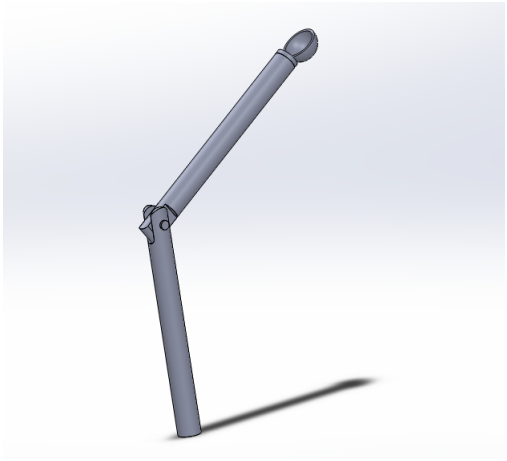

Fig. 3. 2-DOF Serial Manipulator

The 1st link will align itself to the direction of throw and the 2nd link will then project the particle at a desired angular velocity into the bin.

This control problem contains 3 aspects:

- Position control to orient the arm in the direction of throw.
- Motion control of the arm so that the particle follows a desired projectile motion.
- Reducing the inertia of the arm immediately to zero after launching the particle.

II. Mathematical Model

A. Conventions followed:

I_1 = Moment of Inertia of Link 1
I_2 = Moment of Inertia of Link 2
m_1 = Mass of Link 1
m_2 = Mass of Link 2
m = Mass of the particle
h = Length of Link 1
l = Length of Link 2
g = Acceleration due to gravity
$q1$ = Joint angle of Link 1
$q2$ = Joint angle of Link 2

B. Position of center mass of the links in 3D Space:

$$X_1 = \begin{bmatrix} 0 \\ 0 \\ \frac{h}{2} \end{bmatrix}$$

$$X_2 = \begin{bmatrix} \frac{l}{2}cos(q_1)cos(q_2) \\ \frac{l}{2}sin(q_1)cos(q_2) \\ h + \frac{l}{2}sin(q_2) \end{bmatrix}$$

C. Calculating the Jacobian Matrix:

Position vector of the end-effector is given by:

$$X_e = \begin{bmatrix} lcos(q_1)cos(q_2) \\ lsin(q_1)cos(q_2) \\ h + lsin(q_2) \end{bmatrix}$$

Differentiating X_e with respect to time gives us

$$\dot{X}_e = J\dot{q} \tag{1}$$

$$J = \begin{bmatrix} -lsin(q_1)cos(q_2) & -lcos(q_1)sin(q_2) \\ lcos(q_1)cos(q_2) & -lsin(q_1)sin(q_2) \\ 0 & lcos(q_2) \end{bmatrix}$$

$$q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}$$

$$\dot{q} = \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \end{bmatrix}$$

where:

J = Jacobian matrix
q = Joint angle vector
$\dot{q}$ = Joint velocity vector

D. Calculating the Kinetic and Potential energy:

$$KE_1 = \frac{1}{2}m_1\dot{X_1^2} + \frac{1}{2}I_1\dot{q_1^2} \tag{2}$$

$$PE_1 = m_1g(\frac{h}{2}) \tag{3}$$

$$KE_2 = \frac{1}{2}m_2\dot{X_2^2} + \frac{1}{2}I_2\dot{q_2^2} \tag{4}$$

$$PE_2 = m_2g(h + \frac{l}{2}sin(q_2)) \tag{5}$$

E. Dynamical Model of the system

The Lagrangian of any dynamical system is defined as the difference between total kinetic energy and total potential energy of the system.

$$L = (KE_1 + KE_2) - (PE_1 + PE_2) \tag{6}$$

Using the Lagrangian, we derive the Euler-Lagrange equations of motion.

$$\frac{d}{dt}(\frac{\partial L}{\partial \dot{q}}) - \frac{\partial L}{\partial q} = \tau \tag{7}$$

where, τ = External torque acting on the system

Equation (7) can be written in compact matrix form as follows:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau \tag{8}$$

$$M(q) = \begin{bmatrix} I_1 + \frac{1}{4}m_2 l^2 cos^2 q_2 & 0 \\ 0 & I_2 + \frac{1}{4}m_2 l^2 \end{bmatrix}$$

$$C(q, \dot{q}) = \begin{bmatrix} -\frac{1}{8}m_2 l^2 sin(2q_2)\dot{q}_2 & -\frac{1}{8}m_2 l^2 sin(2q_2)\dot{q}_1 \\ \frac{1}{8}m_2 l^2 sin(2q_2)\dot{q}_1 & 0 \end{bmatrix}$$

$$G(q) = \begin{bmatrix} 0 \\ \frac{1}{2}m_2 g l cos(q_2) \end{bmatrix}$$

where:

$M(q)$ = Inertia Matrix
$C(q, \dot{q})$ = Coriolis Forces Matrix
$G(q)$ = Gravitational Torque Vector

F. Projectile Motion

The desired launch velocity can be calculated from the equations of projectile motion.

$$S = v_0 t - \frac{1}{2}gt^2 \tag{9}$$

Let the desired final XY co-ordinate be (x_{des}, y_{des}) and the launch angle (α) be 0 rad.

Motion along the Z-axis,

$$\begin{aligned} -(h + lcos(\alpha)) &= v_0 sin(\alpha)t - \frac{1}{2}gt^2 \\ -(h + lcos(0)) &= v_0 sin(0)t - 4.9t^2 \\ -(h + l) &= -4.9t^2 \\ t &= \sqrt{\frac{h+l}{4.9}} \end{aligned} \tag{10}$$

Motion along the XY plane,

$$\begin{aligned} d &= \sqrt{x_{des}^2 + y_{des}^2} \\ d &= v_0 cos(\alpha)t + \frac{1}{2}at^2 \\ &= v_0 cos(0)t + \frac{1}{2}(0)t^2 \\ &= v_0 t \\ v_0 &= \frac{d}{\sqrt{\frac{(h+l)}{4.9}}} \end{aligned} \tag{11}$$

And, thus the desired angular velocity will be

$$\omega_0 = \frac{v_0}{l} \tag{12}$$

Therefore, for the particle to follow the desired trajectory $\dot{q_2}$ should equal ω_0 and q_2 should equal $\frac{\pi}{2}$ so that we that the desired velocity v_0 and launch angle $(\alpha = 0)$ conditions are met.

III. Controller Design

The purpose of any serial robot control is to maintain the dynamic response of the manipulator in accordance with some specified system performance and goals. We are using 3 different controllers at various time instances to achieve our end goal.

- First, we use PD Control with gravity compensation [3] to align the manipulator in the direction of throw.

  Proportional action provides an instantaneous response to the control error. This is useful for improving the response of a stable system but cannot control an unstable system by itself. Additionally, the gain is the same for all frequencies leaving the system with a nonzero steady-state error. [4]

  Derivative action acts on the derivative or rate of change of the control error. This provides a fast response, as opposed to the integral action, but cannot accommodate constant errors (i.e. the derivative of a constant, nonzero error is 0). Derivatives have a phase of +90 degrees leading to an anticipatory or predictive response. However, derivative control will produce large control signals in response to high frequency control errors such as set point changes (step command) and measurement noise. [5]

  The control law for PD control with gravity compensation is given by:

  $$\tau = K_p \tilde{q} + K_v \dot{\tilde{q}} + G(q) - J^{\mathrm{T}} h_e \tag{13}$$

  where,
  $\tilde{q} = q_d - q$
  $\dot{\tilde{q}} = \dot{q}_d - \dot{q}$
  h_e = external force acting on the end-effector

  In this case, h_e is the gravitational force of the particle. Therefore,

  $$h_e = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix}$$

  Since we are controlling only the position, $\dot{q}_d = 0$ Therefore, $\dot{\tilde{q}} = -\dot{q}$
  Thus, the closed loop control equation becomes

  $$M(q)\ddot{q} + C(q, \dot{q})\dot{q} = K_p \tilde{q} - K_v \dot{q} - J^{\mathrm{T}} h_e \tag{14}$$

- Then we use the Computed Torque Control [6] on the 2nd link to control the velocity of the end effector to launch the ball at a desired velocity.
  The control law is stated as:

  $$\tau = M(q)[\ddot{q_d} + K_p \tilde{q} + K_v \dot{\tilde{q}}] + C(q, \dot{q})\dot{q} + G(q) - J^{\mathrm{T}} h_e \tag{15}$$

  Here again h_e is caused due to the gravitational force of the particle.

Substituting, equation(15) in the mathematical model we get

$$\ddot{q_d} + K_p\tilde{q} + K_v\dot{\tilde{q}} + J^{\mathrm{T}}h_e = 0 \tag{16}$$

- To reduce the arm's inertia to zero at 90° (immediately after launching the particle) we used a slightly modified version of compliance control [7]. We assumed that an external force of infinite magnitude is virtually acting on the end-effector (active compliance) that compels the arm to stop after reaching 90°.
  We adopted the following modified version of compliance control:

$$\tau = J^{\mathrm{T}}K_pJ\tilde{q} + J^{\mathrm{T}}K_vJ\dot{\tilde{q}} + G(q) - J^{\mathrm{T}}h_e \tag{17}$$

  For active compliance h_e can be approximated to,

$$h_e = K_p(X_d - X_r) \tag{18}$$

  Where, X_d is the desired end-effector position after launching the particle. We set this position to be $q_2 = \pi$. X_r is the position at which the end-effector interacts with the environment (here, the forces are virtual and assumed to be infinite) after projecting the particle and that would be $q_2 = \frac{\pi}{2}$

IV. Results

The following are the parameters with which we have designed our 2-DOF manipulator:

$I_1 = 0.125\ Kgm^2$
$I_2 = 0.25\ Kgm^2$
$m_1 = 3.0\ Kg$
$m_2 = 3.0\ Kg$
$m = 0.1\ Kg$
$h = 0.5\ m$
$l = 0.5\ m$
$g = 9.81\ ms^{-2}$

The test case we took into consideration is to project the particle at the desired location of (1,1) in the XY Plane. For the given (x_{des}, y_{des}), from Equations (11 and 12), we calculate the desired angular velocity to be 6.261 rad/s . Thus to achieve this motion, the manipulator first aligns with the direction of throw using PD control as stated above. The direction of throw in this case is $\frac{-\pi}{4} rad$ (i.e $tan^{-1}(x_{des}, -y_{des})$).

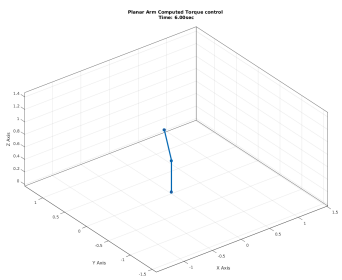

Fig. 4. Simulating the Arm in MATLAB

After which the manipulator continues to execute the Computed Torque motion control on the second link to make it move from 0 to $\frac{\pi}{2}$ with the desired joint velocity of $\dot{q_2}$ equal to 6.261 rad/s when q_2 reaches $\frac{\pi}{2}$.

Since we have three different controllers each of which are independent of the other two, we have complete liberty in tuning them separately. The gains of the system were tuned on a trail and error basis. By increasing the gains beyond a certain point, we obtained better results but at the cost of very high control inputs which were practically unrealizable. After a few tuning iterations, we optimized the gains so that the performance of the controller was satisfactory in terms of tracking the desired set point at a feasible control input.

For PD position control we used the following gain matrices:

$$K_p = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$$

$$K_v = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$$

For Computed Torque control we used the following gain matrices:

$$K_p = \begin{bmatrix} 25 & 0 \\ 0 & 25 \end{bmatrix}$$

$$K_v = \begin{bmatrix} 25 & 0 \\ 0 & 25 \end{bmatrix}$$

For the modified compliance control we used the following gain matrices:

$$K_p = \begin{bmatrix} 20 & 0 \\ 0 & 20 \end{bmatrix}$$

$$K_v = \begin{bmatrix} 40 & 0 \\ 0 & 40 \end{bmatrix}$$

All the simulations (Fig. 4) and experiments were performed on MATLAB. The following are the results obtained while running these simulations.

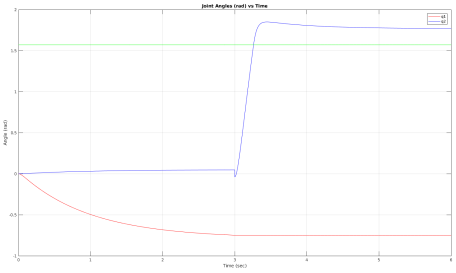

Fig. 5. Joint Angles vs Time

- From the joint angles v/s time graph (Fig. 5) we observe that in almost one-fourth of a second, the second link performs the actual projectile motion. (i.e between 3 to 4 second time period) after which both the links try to maintain their positions. Maximum energy is required during this period.

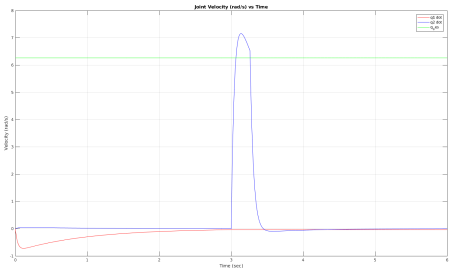

Fig. 6. Joint Velocity vs Time

- From fig. 6, we observe that the angular velocity at the time of projection is about 6.454 rad/s which is slightly more than the desired ω_0 of 6.261 rad/s. This induces an error of about 6 cm when following the desired projectile motion. After completing the projectile motion, the angular velocity of the second joint steeply decreases to zero because of the counter torque applied onto the system by the compliance control.

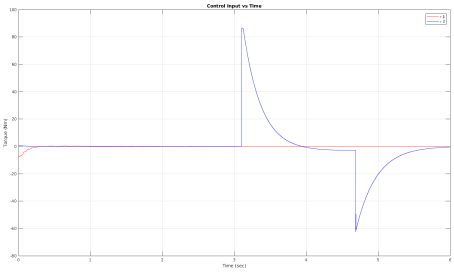

Fig. 7. Control Input vs Time

- The performance of the controller is directly dependent on the torque required by the system. It can be clearly seen from the control output v/s time graph (Fig. 7) that the torque required by the first motor to drive the first link is negligible when compared to the torque required for performing the projectile motion. At the start of projection, a torque of as high as 90Nm is required, while for stabilizing the system a counter torque of 60Nm is required. This torque is provided by the high-torque motor that drives the link attached to the second joint.
- The red curve denotes the desired trajectory of the system in consideration while the blue curve shows the actual trajectory of the motion in the final graph (Fig. 8). The error in the system is the difference in the distance traveled in the horizontal direction. From this graph, it can be seen that the error is about 6cm. Keeping in mind the application for which the robot control system is designed (for throwing garbage in the garbage bin) this system performance is quite satisfactory.

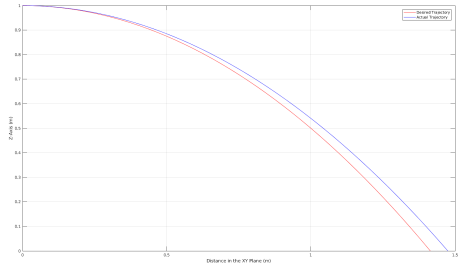

Fig. 8. Desired Trajectory vs Actual Trajectory

V. Conclusion and Future Work

The code written for the control of this serial manipulator was simulated in MATLAB R2017a . From the results we conclude that the system achieves all the control objectives stated in this paper. A minimum of two motors (one low-torque motor and a high-torque motor) are required to actuate this system. The error in the system is about 6cm which is almost negligible considering the bin size to be at least 30-40cm in diameter. Therefore, the system fairly accomplishes the task of orienting, aiming and projecting a particle to a desired location.

The future work lies on expanding the degrees of freedom of the manipulator, making them redundant and try solving the same problem maintaining the control input as low as possible. The programming framework can be switched to ROS and Gazebo that provides us with a simulation environment that is very close to the real-world. We can also improve the mathematical model of the system by taking joint friction and joint masses into consideration. This increases the overall robustness of the system aiding us in understanding the problem better.

References

[1] G. J.Monkman, S. Hesse, R. Steinmann, and Henrikschunk, "Robot grippers," *ISBN:978-3-527-40619-7 pp 463*, November 2006.

[2] S. N. Alam, "Understanding matlab, a textbook for beginners," 30 August 2013.

[3] M. Takegaki and S. Arimoto, "A new feedback method for dynamic control of manipulators," *Transactions ASME, Journal of Dynamic Systems, Measurement and Control, Vol. 103, pp. 119125*, 1981.

[4] G. Goodwin, S. Graebe, and M. Salgado, "Control system design," *Prentice Hall PTR. ISBN 0139586539*, 2000.

[5] G. Franklin, A. Emami-Naeini, and J. Powell, "Feedback control of dynamic systems," *Wesley Longman Publishing Co., Inc. ISBN 0201527472*, 1993.

[6] K. Fu, R.Gonzalez, and C.Lee, "Robotics: Control, sensing, vision and intelligence," *McGrawHill.*, 1987.

[7] C. Canudas, B. Siciliano, and G. Bastin, "Theory of robot control," *Springer-Verlag, London*, 1996.

[8] M. Spong and M. Vidyasagar, "Robot dynamics and control." *John Wiley and Sons.*, 1989.

[9] C. Canudas, B. Siciliano, and G. Bastin, "Theory of robot control," *Springer-Verlag, London.*, 1996.