Carving Robot Arm Control Using Two Different Control Methods
Team 2

Andy Choi
Mechanical and Aerospace Engineering
University of California, Los Angeles
Los Angeles, CA
Email: akchoi@ucla.edu

Dooseop Song
Mechanical and Aerospace Engineering
University of California, Los Angeles
Los Angeles, CA
Email: sds10117@ucla.edu

***Abstract*—Team 2's main goal, which was motivated by a simple drawing robot, was to put emphasis on making robots more useful in daily lives. Robot design with kinematics and trajectory following with dynamics calculation of the manipulator have been taught throughout the series of the 263 robotics courses at UCLA. This paper presents the simulation of the carving robot arm to prove the ability that the robotic arm can both control force and motion along the trajectory on an wooden cube. First, simple descriptions of the control methods are presented. Then, the robot designs with its parameters are shown. Then, the paper describes the necessary assumptions needed for the simulation with the general two controls algorithm. Next, snapshots of the simulation are described in detail with the discussion of the effects of different gain values that were used during the simulation. The results of the project will be applicable to artistic works that deal with small objects, miniature carving robot control methods and nano-scale etching machine in the IT industry field.**

Keywords - Carving Robot, Control of Robotic Systems, Joint Space Inverse Dynamics Control, Hybrid Force/Motion Control

I. Overview of the Project

Different types of manipulator robots are being researched by scientists to increase the productivity in the real world by developing robot's ability to recognize the environment and perform more detail works such as selective object picking in random situations [1]. Especially in the industrial sites, manipulator robots are mainly developed to perform works that are difficult for humans or to assist humans[2]. These robots are generally big size, heavy and costly, people assume that they are not close to our daily life yet. Therefore, numerous efforts have been made for people to study, learn or understand robots in the field of education and science; a simple drawing robot[3] has an application in the art industry, which is very effective in terms of easy accessibility for daily life usages. The drawing robot's mechanism is a simple motion control along the trajectory without considering force and the orientation of the end-effector.

In order to perform more artistic works such as painting or carving, the manipulator should be able to be precisely control and stably interact with the environment[4]. In this sense, it is significant to conduct researches to focus on the detail interaction between the robot and humans or between the environment, for that is the very basic condition to co-live with people. To go beyond more than just a painting robot, this paper presents a simple carving robot.

II. Objectives of the Controller Design

The main purpose of this robot is to carve an wooden block using two different control methods: **Joint Space Inverse Dynamics Control** and **Hybrid Force / Motion Control**.

A. Joint Space Inverse Dynamics Control

Figure 1[5] shows the Joint Space Inverse Dynamics Control Block Diagram, where the inner feedback loop, covered in green, is used to obtain a linear and decoupled input/output relationship and the outer feedback loop, covered in red, is used to stabilize the overall system. The main goal of this control is to perform exact linearization of system dynamics by using nonlinear state feedback. With a given undamped natural frequency, ω_n and a damping ratio, ζ, K_P and K_D are calculated as following:

$$K_P = diag(\omega_{n1}^2, ..., \omega_{nn}^2)(1)$$

$$K_D = diag(2 * \zeta_1 * \omega_{nn}, ..., 2 * \zeta_n * \omega_{nn})(2)$$

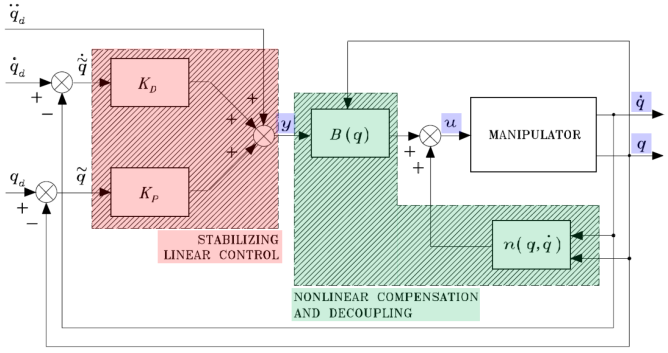

Fig. 1. Joint Space Inverse Dynamics Control Block Diagram

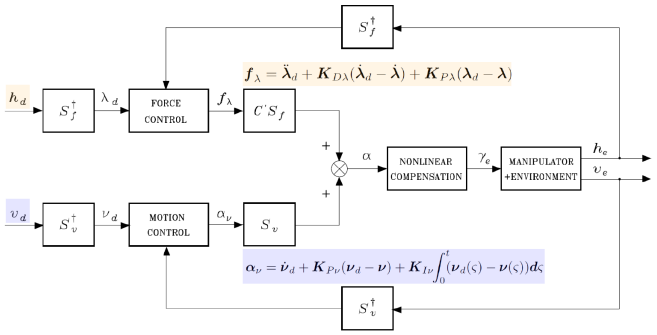

Fig. 2. Hybrid Force/Motion Control Block Diagram

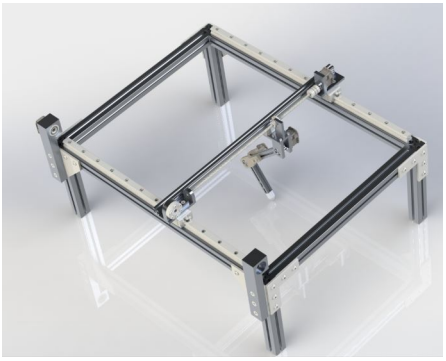

Fig. 3. Bird eye view of the Carving robot manipulator originated from RTiST

The two equations[6] give a decoupled system. This control method is used to move the arm as a free motion in the z-direction of 10 cm. The exact path of the robot is given, which allows to calculate corresponding joint angles through inverse kinematics. The q_d, angle desired, values are then put as inputs to the controller.

B. Hybrid Force / Motion Control

Figure 2[7] shows the Hybrid Force/Motion Control Block Diagram. The main goal of this controller is to separate control of end-effector motion and forces into two separate decoupled problems. The reference values are force-based and motion based that's treated separately depending on the decoupled controllable sub-spaces denoted as S_f and S_v.

The user designs $K_{P\nu}$ and $K_{D\nu}$ from the control law[8],

$$\alpha_\nu = \dot{\nu} + K_{P\nu}(\nu_d - \nu) + K_{I\nu}\int_0^t (\nu_d(s) - \nu(s))ds (3)$$

The user also designs $K_{P\lambda}$ and $K_{D\lambda}$ from the control law[8],

$$f_\lambda = \ddot{\lambda_d} + K_{D\lambda}(\dot{\lambda_d} - \dot{\lambda}) + K_{P\lambda}(\lambda_d - \lambda) (4)$$

Hybrid Force/Motion Control is used to carve wood by 1 cm deep in the z-direction, while moving along the trajectory line of 20 cm or 15 cm along the y-direction.

III. Methods

A. Robot Design

The robot design was originated from the RTiST, a drawing robot developed in 263A course in the Fall quarter. The detail

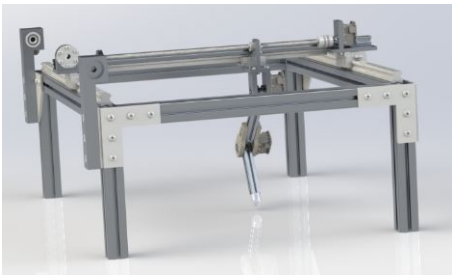

Fig. 4. Side view of the Carving robot manipulator originated from RTiST

features can be obtained in Figure 3 and 4. The operating mechanism is similar with a commercial 3d printer mounted on a gantry frame. The X, Y position of the end-effector are determined by the 2 prismatic joints moving along the square frame and the z position is controlled by the 2 revolute joints of the 2-link planar arm. Brief specifics of the robot are that the link lengths are 18 cm each and each joint is operated by a MX-28A motor. The milling tool is at the end of link 2 and the total height of the gantry frame is 30 cm.

B. Assumptions

The first and main assumption for the project is that the robot moves along the Y direction with no friction. Specifically, in the real world, the robot carves along X, Y, Z axis as the arm follows the trajectory on a cube, which means the robot has to control force and motion at the same time for all directions. The reciprocity condition[9],

$$h_e^T * v_e = 0 (5)$$

gives the constraint on the robot that it cannot control force and motion along same direction for the hybrid force / motion control. Therefore, for the carving performance, the force control method is applied along the z-axis that carves the cube 1 cm deep and the motion control method is applied along the y-axis direction that follows the straight line trajectory.

The next assumption is that the end-effector's X and Y positions, which corresponds the starting point of the trajectory to carve the cube, are fixed. This is mainly due to the insufficient knowledge and skills of the prismatic joint controls; the original idea was to move the robot in the X and Y direction by controlling prismatic joints on the gantry frame. The compensation for this limitation is to fix the starting point and carve a straight line, and then the cube would be rotated 90 degrees to continue the carving trajectory of a desired rectangle trajectory.

Third, the parameters for the robot and the motors are from Solidworks and from the instructions of the MX-28A motor. Parameters below are decided values for the project. The mass of the each link

$$m_{link} = 1.87 \ kg$$

The length of the each link

$$l_{link} = 0.18 \ m$$

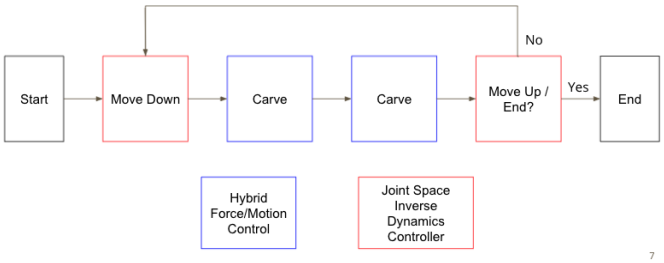

Fig. 5. Algorithm Flow Diagram

The inertia moments relative to the center of the mass of each link

$$I_{cm} = 0.09 \ \ kg * m^2$$

The mass of the each MX-28SA motor

$$m_{motor} = 0.04 \ \ kg$$

The inertia moments of each motor

$$I_{motor} = 0.01 \ \ kg * m^2$$

In addition to the indicated parameters, another assumption is that the tool does not affect the dynamics of the robot. The tool has a function of drilling, but the mass is negligible for the simplicity of the calculation of dynamics.

For the simulation, the stiffness of the wood being much more elastic than the real wood's stiffness is considered. Because of the robot's small dimension, light weight, and low motor power, it cannot carve the hard wood. As a result, the assumed stiffness of the cube is $1500\frac{N}{m}$ for this project, which is much less than the actual stiffness of the wooden cube. For the stiffness calculation, the wooden block is assumed to be a Cedar, Northern White wood, which has a Young's Modulus of 4400 MPa[10]. Then, the area that the wood is cut is $0.01cm^2$, with the height of $L = 2cm$. Using the relationship of the stiffness and Young's modulus,

$$k = Y * \frac{A}{L} (5)$$

$$k = 2200\frac{N}{cm} = 220000\frac{N}{m}$$

Lastly, the desired trajectory was calculated with a trapezoidal velocity profile introduced in the provided functions in 263C course homeworks.

C. Algorithm Flow Diagram

Figure 5 shows the general flow diagram of the algorithm. The robot is placed at an initial position. Then, using Joint Space Inverse Dynamics Control, the arm moves down to the top of the block. Once the milling tip reaches the top of the block, the robot uses Hybrid Force/Motion Control to carve back and forth of the line segment. Once it comes back to the original position, the robot then moves up using Joint Space Inverse Dynamics Control, then the cube is re-positioned; this concludes the first carving line segment. Then, this process repeats until the end.

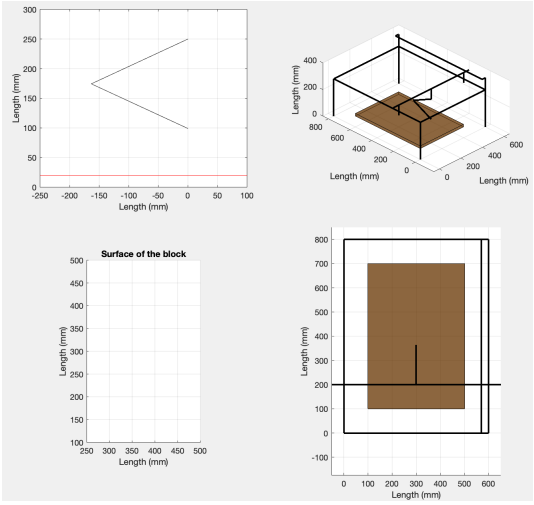

Fig. 6. Initial Position of the Robot

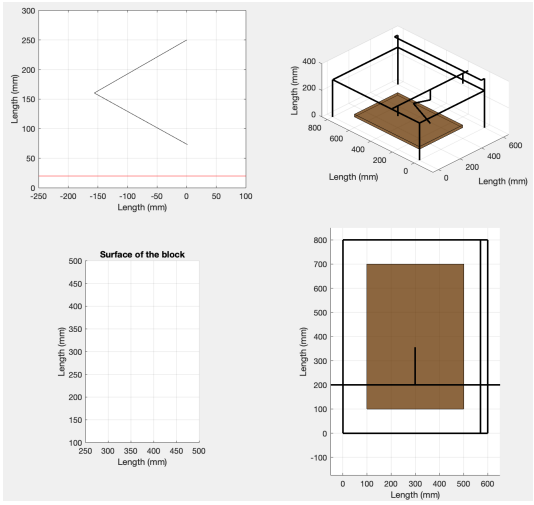

Fig. 7. Robot Arm Going Down Using Joint Space Inverse Dynamics Control

IV. Results

A. Video Simulation

A YouTube link: ***Video Link*** is attached for the simulation.

B. Joint Space Inverse Dynamics Control

The top left plot of Figure 6 from the simulation shows the robot in the Y-Z plane. The red horizontal line indicates the location of the surface block, which is at 2 cm. The top right shows the camera view of the set up with the robot and the block. The bottom left shows the bird's-eye view of the surface of the block. The bottom right shows the bird's-eye view of the system's set up.

Figure 7 shows the robot arm as it is going down to top of the block using Joint Space Inverse Dynamics Control. The initial and final angles are calculated from inverse kinematics of two revolute planar arm, which are then inserted as inputs to the control algorithm. Figure 8 shows the actual Joint 1 trajectory and the desired Joint 1 trajectory as the robot goes

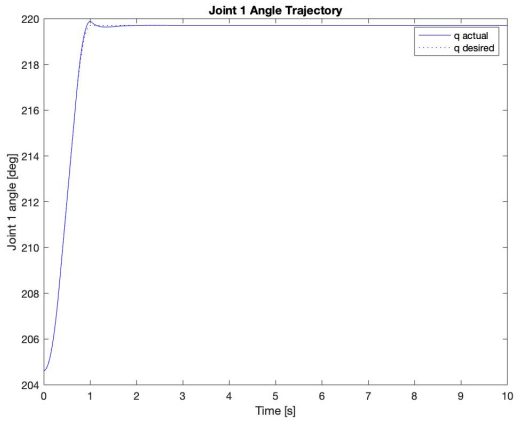

Fig. 8. Joint 1 Trajectory as the robot goes from (0,100) to (0,20)

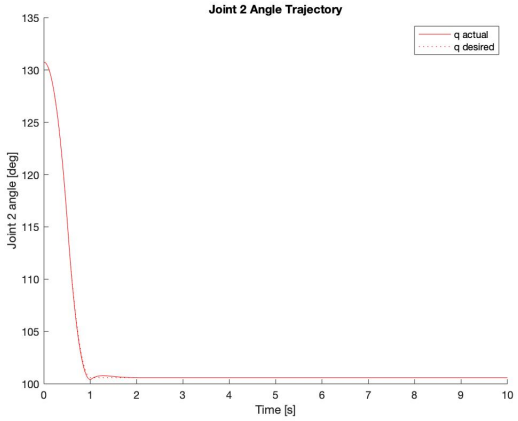

Fig. 9. Joint 2 Trajectory as the robot goes from (0,100) to (0,20)

from (0,100 mm) to (0,20 mm). Figure 9 shows the joint 2 angle trajectory. These trajectories have been calculated with a trapezoidal velocity profile. Figure 10 shows both the joint angle errors with respect to time; the error is in the magnitude of 10^{-3} and the error is stabilized at around two seconds.

C. Hybrid Force/Motion Control

Figure 11 shows the snapshot of the simulation as the robot is carving the block by 1 cm as it travels in the y-direction. The bottom left of the simulation shows the trajectory of how the block is being carved; the red line indicates the path that the robot carving is following. After the robot carves by 20 cm, it carves back by 20 cm to come back to the original position. Then, the robot arm moves up to complete the algorithm mentioned in Section III. Methods Part C.

After the arm is lifted up, the block is then rotated and placed in order to cut the block horizontally. The current limitations of the robot only allows it to move in the y-direction and not in the x-direction, which forces the cube to be rotated. Figure 10 shows the robot carving the block horizontally as it travels in the y-direction by 10 cm. This process is repeated until all four lines are cut.

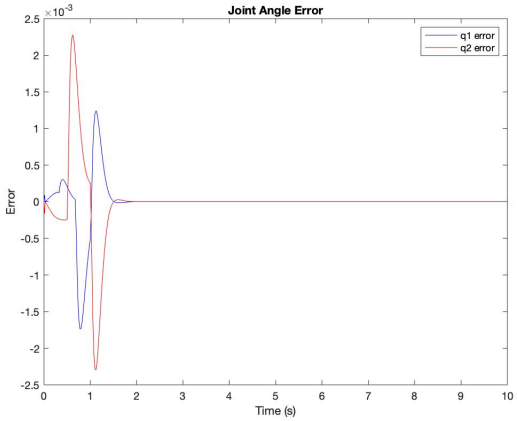

Fig. 10. Joint Angle Error as the robot goes from (0,100) to (0,20) with $K_P = 100$, $K_D = 16$

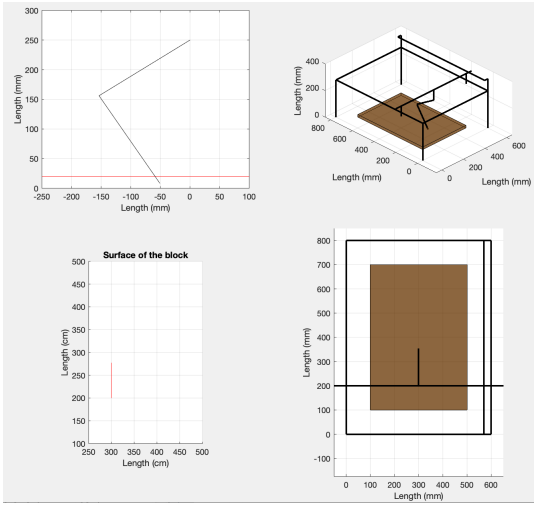

Fig. 11. Robot carving the block as it travels in the y-direction

Figure 12 shows the final result of how the block is cut from the top-down view. The rectangle that is cut is 20 cm long and 15 cm wide centered at (375 mm, 300 mm).

V. Discussion

A. Joint Space Inverse Dynamics Control

1) Gain Value Comparisons: For the joint space control, angular velocity is set to

$$\zeta = 0.8, w_n = 10\frac{rad}{s}$$

. Then, with the equation (1) and equation (2), the K_P and K_D values are calculated as $K_P = 100$ and $K_D = 16$; the two joints have equal magnitudes in the gain values. To test the effects of different gain values, the angular velocity has been modified to

$$\zeta = 0.8, w_n = 5\frac{rad}{s}$$

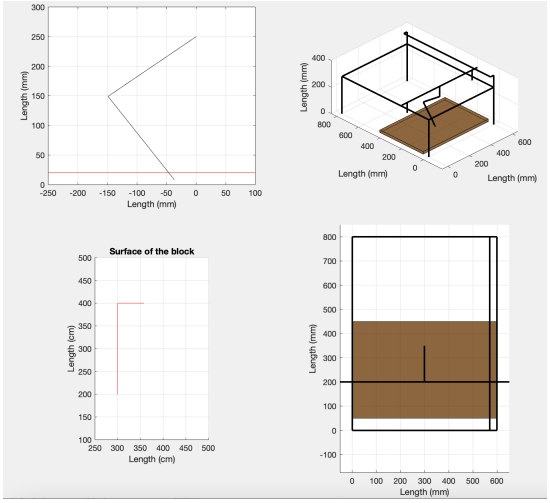

Fig. 12. Robot carving the block horizontally as it travels in the y-direction

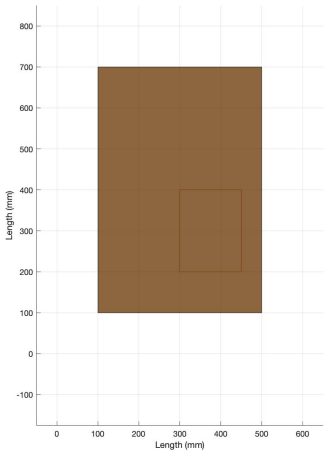

Fig. 13. Final Result of the carved block in the bird's-eye view

. Then, with the same calculation methods, the K_P and K_D values are calculated as $K_P = 25$ and $K_D = 8$.

Figure 14 shows both the joint angle errors as the robot goes from (0,100 mm) to (0,20 mm). Comparing with Figure 10 where the gain values are different, $K_P = 100$ and $K_D = 16$, the error magnitude of Figure 14 is twice that of Figure 10; this results in that the higher the gain values, the smaller the error becomes.

2) Capabilities: Due to small errors shown in Figure 10, the robot is able to move down in almost a straight path using the Joint Space Inverse Dynamics Control. Even though there is an error towards the end of the trajectory, the error is small enough to not have significant effects to the movements of the robot.

3) Limitations: The limitations of the Joint Space Inverse Dynamics Control mainly come from the inverse kinematics solution approach. In a normal 2 revolute joint planar arm, there are four possible solutions (elbow up front, elbow down front, elbow up back, elbow down back). However, the current inverse kinematics approach chooses only one possible solution. This caused some solutions to project inaccurate trajectories that resulted from trapezoidal velocity profile, which resulted in the robot suddenly moving from elbow up to elbow down positions or from elbow down front to elbow up back, which is not ideal in realistic situations.

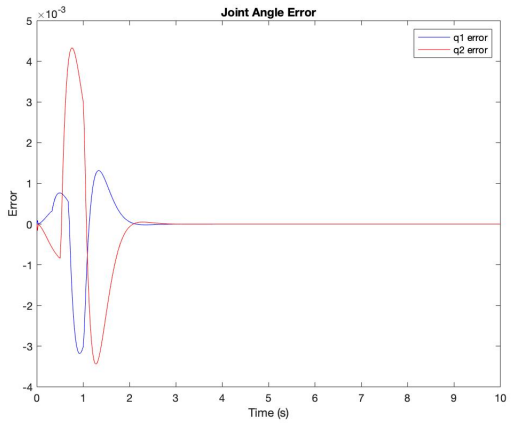

Fig. 14. Joint Angle Error as the robot goes from (0,100) to (0,20) with $K_P = 25$, $K_D = 8$

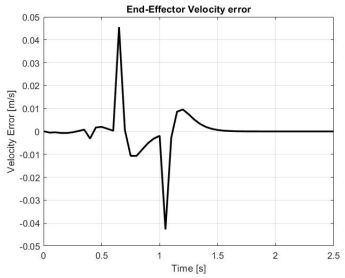

Fig. 15. Velocity error of the end effector with $K_{P\nu} = 20$, $K_{I\nu} = 100$ and $K_P = 100$, $K_D = 20$

B. Hybrid Force/Motion Control

1) Gain Value Comparisons: Given desired contact force being 15 $\frac{N}{m}$, the robot carves with depth of 1 cm deep while travelling in a straight line of the trajectory of 20 cm long. The first approach to set the gain values for the hybrid force and motion controller as

$$\zeta = 1, w_n = 10\frac{rad}{s}$$

to make the system to be critically damped. Considering second-order system's control equation,

$$\omega_n^2 = 100 = K_{I\nu} = K_P$$

$$2 * \zeta * \omega_n = 20 = K_{P\nu} = K_D$$

Figure 15 to 17 shows the results of the Hybrid Force/Motion control with given the gains. Velocity errors are small and stabilize within 1.5 seconds, which is satisfactory for the

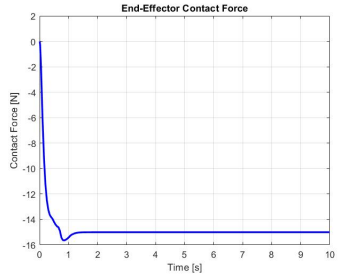

Fig. 16. Contact force of the end effector with $K_{P\nu} = 20$, $K_{I\nu} = 100$ and $K_P = 100$, $K_D = 20$

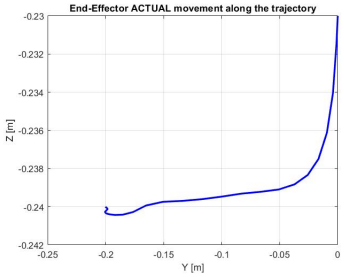

Fig. 17. Actual movement of the end-effector along the trajectory with $K_{P\nu} = 20$, $K_{I\nu} = 100$ and $K_P = 100$, $K_D = 20$

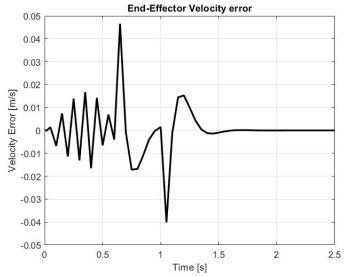

Fig. 18. Velocity error of the end effector with $K_{P\nu} = 20$, $K_{I\nu} = 100$ and $K_P = 500$, $K_D = 50$

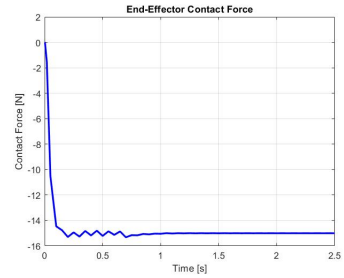

Fig. 19. Contact force of the end effector with $K_{P\nu} = 20$, $K_{I\nu} = 100$ and $K_P = 500$, $K_D = 50$

robot motion. In Figure 16 and 17, the trajectory following along the y direction is exactly 20cm as desired. However, the force control aspects are not suitable for clear carving because there's an overshoot in the contact force graph, which makes the robot to apply more than desired force on cube and the strange and unstable performance of the end effector has been observed.

In order to eliminate those two undesired effects, increased force controller gains are applied to the system, which expects the trajectory and contact force to be faster than before. The results are from Figure 18 to 20 and the performance seems to be more ideal. However, according to the force and trajectory graphs, the carving output shows very rough and bumpy surface with fast oscillation in the force control. However, the gain values can't be increased to infinity or be too large for the better performance because it causes the damage on the motor due to the limitation of the motor capacity.

After some trial and error, suitable gains, $K_{P\nu} = 20$, $K_{I\nu} = 100$ and $K_P = 240$, $K_D = 40$, were found for the cube carving performance as seen from the Figure 21 to 23. The arm reacts to follow the trajectory in moderate time and values and the desired contact force is achieve in approximately 0.7 seconds so that the carved line shows quite smooth surface.

The remaining problem is that the robot cannot carve the cube 1 cm deep for the whole line because it does not exactly follow the trajectory due to the errors in transient phase. To compensate for the lack of the performance, the robot is planned to carve the line two times from right to left and left right so that both ends of the straight line are cut clearly with fine surface. This is why the arm comes back to the starting point again in the simulation.

2) Capabilities: The application of the Hybrid Force / Motion controller provides carving straight line on the cube of 1 cm deep with clear cut of both left and right ends on the line. The robot can also apply force of 15 N exactly on the cube.

3) Limitations: During the process of the project, the gain tuning issues for the hybrid controller was the main limitation. Obtaining the ideal performance in both motion and force control was difficult mainly because gain values for motion and force controller are dependent, which made the designer difficult to determine the suitable values for the project. When the force performance is satisfied, the oscillation on velocity error becomes larger and in the case that the velocity error and stabilized condition are suitable, carving depth becomes lower than the goal. It is similar with the "waterbed effect" in that an enhancing one side of the performance results in decreasing the quality of the other side of the performance. The designer had to decide controller gains which nearly satisfies the actual performance to be compatible with the desired outcome. From the experience of the project, deeper knowledge on force / motion control is required for the realistic robot arm control to interact with the environment.

C. Challenges

The critical challenge was that one of the group members decided to drop the class, so the project had to be done with two members; this caused the team to rush through the schedule of the project. Also, the first desired concept

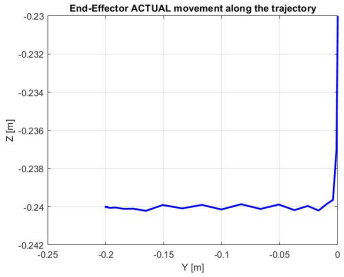

Fig. 20. Actual movement of the end-effector along the trajectory with $K_{P\nu} = 20$, $K_{I\nu} = 100$ and $K_P = 500$, $K_D = 50$

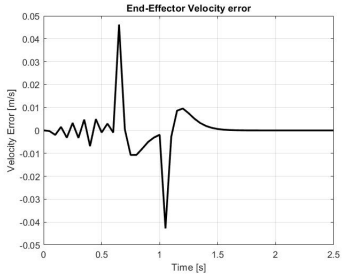

Fig. 21. Velocity error of the end effector with $K_{P\nu} = 20$, $K_{I\nu} = 100$ and $K_P = 240$, $K_D = 40$

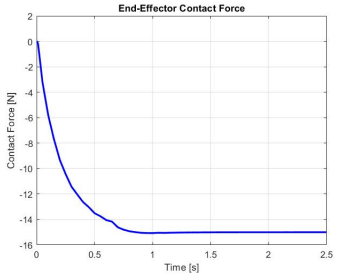

Fig. 22. Contact force of the end effector with $K_{P\nu} = 20$, $K_{I\nu} = 100$ and $K_P = 240$, $K_D = 40$

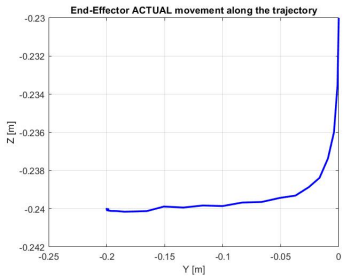

Fig. 23. Actual movement of the end-effector along the trajectory with $K_{P\nu} = 20$, $K_{I\nu} = 100$ and $K_P = 240$, $K_D = 40$

for the carving machine was to utilize the Hybrid Force / Motion control during the entirety of the trajectory, but without understanding the difficult contents and core concepts of the controller without a class, the project had to be done after the material has been taught.

Additionally, the work to rotate the cube after lifting the arm up and down for the each segment carving had to be done because of the inability to control prismatic joints. This required the robot controller to incorporate the two control methods: Joint Space Inverse Dynamics controller and Hybrid Motion / Force controller.

Lastly, the robot's current design had limitations to actually carve the wood, which became a serious consideration for the project. The current robot is too small and the motors don't have enough power to carve the wood with stiffness that's been calculated in Section III. Part B. The addition of one more assumption, which was to reduce the magnitude of the stiffness, was mandatory for the reasonable simulation. The detail considerations in the design step and relating it with the performance were great lessons for this project.

VI. Future directions

There are multiple directions in which this project can proceed in the future. The robot was initially designed to move in the x,y direction with prismatic joints and z direction with two revolute joints. However, the scope of 263C: Control of Robotics didn't cover the controls of prismatic joints as it only covered control of revolute joints. Control in both x,y directions will allow the robot to cut in any directions, without having to re-position the block every time it completes one segment; this will allow the robot to cut through curved lines. Also, realistically, control in both force and motion along the same direction is necessary especially for authentic designs with dynamic curves and different depth of the carving performance, which means that friction should be considered in both x,y direction. Lastly, the robot that was built for the simulation purposes cannot actually cut through the wood block because it is not big enough and doesn't have strong motors to apply enough force to cut through an actual wood block. The control algorithm can be redesigned to have enough force to carve the actual wood.

Acknowledgment

The authors would like to thank Professor Veronica Santos and TA Jesse Cha for a successful spring quarter despite the COVID-19 pandemic.

References

[1] K. Harada, W.Wan, T. Tsuji, K. Kikuchi, K. Nagata, H.Onda, *Experiments on Learning Based Industiral Bin-picking with Iterative Visual Recognition* (2018). "https://arxiv.org/abs/1805.08449"

[2] M. Raessa, J. Chen, W. Wan, K. Harada, *Human-in-the-loop Robotic Manipulationi Planning for Collaborative Assembly* (2019). "https://arxiv.org/abs/1909.11280"

[3] A. Hamori, J. Lengyel, B. Resko *3DOF drawing robot using LEGO-NXT* (2011). "https://ieeexplore.ieee.org/abstract/document/5954761"

[4] V. Duchaine and C. Gosselin, *Safe, Stable and Intuitive Control for Physical Human-Robot Interaction* (2009). "https://ieeexplore.ieee.org/abstract/document/5152664"

[5] Sicillano, et al. _Robotics: Modelling, Planning, and Control **Fig. 8.22** Londong: Springer-Verlag, 2009

[6] Sicillano, et al. _Robotics: Modelling, Planning, and Control **Pg. 331** Londong: Springer-Verlag, 2009

[7] Sicillano, et al. _Robotics: Modelling, Planning, and Control **Fig. 9.15** Londong: Springer-Verlag, 2009

[8] Sicillano, et al. _Robotics: Modelling, Planning, and Control **Pg. 398** Londong: Springer-Verlag, 2009
[9] Sicillano, et al. _Robotics: Modelling, Planning, and Control **Pg. 487** Londong: Springer-Verlag, 2009
[10] Ames Web, *Young's Modulus (Modulus of Wood)*, https://amesweb.info/Materials/Youngs-Modulus-of-Wood.aspx

VII. Appendix

1. https://www.youtube.com/watch?v=nihArZl8k4I :Link of the simulation video

2. Contributions

Contributions	
Andy	- Joint Space Inverse Dynamics Control - Simulation - Report
Dooseop	- Hybrid Force / Motion Control - CAD modifications - Report

3. Custom Solidworks CAD designs

The CAD models have been submitted with this report

4. Custom Code

The simulation code is attached at the bottom of this report.

```
clear all;clc;close all;
global links H commands ;
global frame_data link1_data link2_data link3_data;
global link4_data ;
global T_W T_0 speed link_vars goal ;
global should_stop;

should_stop = false;

% Robot attributes
% Frame 0 offset
% Link lengths
links = [0 0 0 180 180];
H = 250;

% Path data
rr = 0.75;
xx = sqrt(3)/2*rr;

% How much to move the pen
top = 250;

commands = {
    {'d', 300, 200, 300,400,0,400,600,20,0}...
    {'v', 300, 400, 300,400,0,400,600,20,0}...
    {'m', 300, 200, 300,400,0,400,600,20,0}...
    {'u', 300, 400, 300,400,0,400,600,20,0}...
    {'d', 300, 400, 300,250,0,600,400,20,0}...
    {'h', 450, 400, 300,250,0,600,400,20,0}...
    {'m', 300, 400, 300,250,0,600,400,20,0}...
    {'u', 450, 400, 300,250,0,600,400,20,0}...
    {'d', 450, 400, 400,200,0,400,600,20,0}...
    {'v', 450, 200, 400,200,0,400,600,20,0}...
    {'m', 450, 400, 400,200,0,400,600,20,0}...
    {'u', 450, 200, 400,200,0,400,600,20,0}...
    {'d', 450, 200, 550,250,0,600,400,20,0}...
    {'h', 300, 200, 550,250,0,600,400,20,0}...
    {'m', 450, 200, 550,250,0,600,400,20,0}...
    {'u', 450, 200, 550,250,0,600,400,20,0}...
};
% d = pencil down, v = vertical line, m = move back, u = pencil up,h =
% horizontal line
% .

% Wireframe data
frame_data = [
    0 0, 0 0, 0 280;
    600 600, 0 0, 0 280+90;
    0 0, 800 800, 0 280;
    600 600, 800 800, 0 280+90;
    0 600, 0 0, 280 280;
    0 600, 800, 800 280 280;
```

```
    0 0, 0 800, 280 280;
    600 600, 0 800, 280 280;
    600 600-30, 0 0, 280+90 280+90;
    600 600-30, 800 800, 280+90 280+90;
    600-30 600-30, 0 800, 280+90 280+90
];

link1_data = [
    0 78 0 0 0 0;
    0 78 600 600 0 0;
     78 78 -50 650 0 0
];

link2_data = [0 -75 0 0 0 0];
link3_data = [0 180 0 0 0 0];
link4_data = [0 180 0 0 0 0];

T_W = eye(4);
T_0 = T_W*ht(fixed_angle(-pi/2, -pi/2, 0), [0; 0; H]);

% Initialize variables
speed = 10;

ox = 300;
oy = 200;
goal = [0 0 150];

link_vars = [oy ox ik_new([0 0 100], links)];

% Set up figure
fig = figure(1); clf;

% global vid;
% vid = VideoWriter('sim.avi', 'Uncompressed AVI');
% open(vid);

% update_robot(cmd);
%
subplot(2, 2, 3);
hold on
axis equal
axis([250 500 100 500]);
grid on
title("Surface of the block")
ylabel("Length (mm)")
xlabel("Length (mm)")
sy = 0;
sx = 0;
% Main loop
for i=1:length(commands)
    cmd = commands{i};

    % If start of move, move arm to the bottom
    if cmd{1} == 'd' || cmd{1} == 'u'
```

```
    if cmd{1} == 'd'
        z = linspace(150,220,2);
    else
        z = linspace(240,150,2);
    end
    actual_angle = main_contr(z,cmd{1},links);
    actual_angle = actual_angle{1};
    for w = 2:length(actual_angle)
        prev_ang = actual_angle(w-1,:);
        current_ang = actual_angle(w,:);
        prev_ang = round(prev_ang,3);
        current_ang = round(current_ang,3);
        if prev_ang ~= current_ang
            link_vars(3:4) = prev_ang;
            update_robot(cmd);
        end
    end

end
subplot(2, 2, 3);
k = 1;
if cmd{1} == 'v'
    previous_cmd = commands{i-1};
    % this y is robot frame's y
    y = -1*abs(cmd{3}-previous_cmd{3});
    pos_actual = main_contr(y,cmd{1},links);
    pos_actual = pos_actual*1000;
    starting_x = previous_cmd{2};
    end_x = cmd{2};
    starting_y = previous_cmd{3};
    end_y = cmd{3};

    for w = 2:length(pos_actual)
        prev_pos = pos_actual(w-1,:);
        current_pos = pos_actual(w,:);
        prev_pos = round(prev_pos,3);
        current_pos = round(current_pos,3);
        if prev_pos ~= current_pos
            actualpose(k,:) = prev_pos;
            k = k+1;
        end
    end
    x_traj = linspace(starting_x,end_x,length(actualpose));
    y_traj = linspace(starting_y,end_y,length(actualpose));
    for j = 2:length(actualpose)
        ox = x_traj(j-1);
        nx = x_traj(j);
        oy = y_traj(j-1);
        ny = y_traj(j);
        plot_line(ox,nx,oy,ny)
        pose = actualpose(j-1,:);
        angle = ik_new([0 pose], links);
        link_vars(3:4) = angle;
        update_robot(cmd);
```

```
    end
end
k = 1;
if cmd{1} == 'h'
    previous_cmd = commands{i-1};
    % this y is robot frame's y
    y = -1*abs(cmd{2}-previous_cmd{2});
    pos_actual = main_contr(y,cmd{1},links);
    pos_actual = pos_actual*1000;

    starting_x = previous_cmd{2};
    end_x = cmd{2};
    starting_y = previous_cmd{3};
    end_y = cmd{3};
    for w = 2:length(pos_actual)
        prev_pos = pos_actual(w-1,:);
        current_pos = pos_actual(w,:);
        prev_pos = round(prev_pos,3);
        current_pos = round(current_pos,3);
        if prev_pos ~= current_pos
            actualpose(k,:) = prev_pos;
            k = k+1;
        end
    end
    x_traj = linspace(starting_x,end_x,length(actualpose));
    y_traj = linspace(starting_y,end_y,length(actualpose));
    for j = 2:length(actualpose)
        ox = x_traj(j-1);
        nx = x_traj(j);
        oy = y_traj(j-1);
        ny = y_traj(j);
        plot_line(ox,nx,oy,ny)
        pose = actualpose(j-1,:);
        angle = ik_new([0 pose], links);
        link_vars(3:4) = angle;
        update_robot(cmd);

    end
end
if cmd{1} == 'm'
    previous_cmd = commands{i-1};
    % this y is robot frame's y
    if previous_cmd{1} == 'v'
        y = 1*abs(cmd{3}-previous_cmd{3});
    else
        y = 1*abs(cmd{2}-previous_cmd{2});
    end
    pos_actual = main_contr(y,cmd{1},links);
    pos_actual = pos_actual*1000;
    pos_actual(:,1) = y-pos_actual(:,1);
    for w = 2:length(pos_actual)
        prev_pos = pos_actual(w-1,:);
        current_pos = pos_actual(w,:);
```

```
                prev_pos = round(prev_pos,3);
                current_pos = round(current_pos,3);
                if prev_pos ~= current_pos

                    angle = ik_new([0 prev_pos], links);
                    link_vars(3:4) = angle;
                    update_robot(cmd);
                end
            end
        end

        if i == length(commands)
            dwg = subplot(2,2,3);
            figure;
            x = axes;
            hold on
            copyobj(dwg.Children,x)
            draw_cube([300,400,0],[400,600,0],0);
            view(0,90);
            axis vis3d;
            axis equal;
            grid on;
            axis([-50 650 -175 850 0 400]);
            ylabel("Length (mm)")
            xlabel("Length (mm)")


        end

    end

    % close(vid);
    % disp("Done");
    % done = 1;

    function update_robot(cmd)
        global should_stop;
        if ~ishandle(1)
            should_stop = true;
            return;
        end

        subplot(2, 2, 1);
        block_surface_x = linspace(-250,100);
        block_surface_y = ones(1,100)*20;
        plot(block_surface_x,block_surface_y,'r')
        hold on
        draw_arm_fig();
        axis equal;
        axis([-250 100 0 300]);
        ylabel("Length (mm)")
        xlabel("Length (mm)")
        grid on;
```

```
    dwg = subplot(2, 2, 3);
    x = subplot(2, 2, 2);

    cla;
    copyobj(dwg.Children, x);
    draw_cube([cmd{4},cmd{5},cmd{6}],[cmd{7},cmd{8},cmd{9}],cmd{10});
    draw_iso_fig();
    view(-45, 30);
    axis vis3d;
    axis equal;
    grid on;
    axis([-50 650 -175 850 0 400]);
    ylabel("Length (mm)")
    xlabel("Length (mm)")
    zlabel("Length (mm)")

    y = subplot(2, 2, 4);
    cla;
    copyobj(x.Children, y);
    view(0, 90);
    axis vis3d;
    axis equal;
    grid on;
    axis([-50 650 -175 850 0 400]);
    ylabel("Length (mm)")
    xlabel("Length (mm)")

    drawnow;
    pause(0.01);

%     global vid;
%     writeVideo(vid, getframe(gcf));
end

function plot_line(ox,nx,oy,ny)

    subplot(2, 2, 3);

    plot([ox nx], [oy ny], 'Color', 'r', 'MarkerSize', 2);
    ylabel("Length (cm)")
    xlabel("Length (cm)")

%     global vid;
%     writeVideo(vid, getframe(gcf));

%     plot([ox nx], [oy ny], 'Color', pen{1}, 'MarkerSize', pen{2});

%     update_robot(cmd)
end

function T=generate_frames(T_0, link_vars)
    L1 = 180;
    L2 = 125;
    L3 = 50;
```

```
    T = zeros(4, 4, 8);
    T(:, :, 1) = T_0*TF(0, 0, link_vars(1), 0);
    T(:, :, 2) = T(:, :, 1)*TF(-pi/2, 0, link_vars(2), pi);
    T(:, :, 3) = T(:, :, 2)*TF(0, 0, 0, -pi/2-link_vars(3));
    T(:, :, 4) = T(:, :, 3)*TF(0, L1, 0, -link_vars(4));
%     T(:, :, 5) = T(:, :, 4)*TF(0, L2, 0, -link_vars(5));
%     T(:, :, 6) = T(:, :, 5)*TF(pi/2, L3, 0, link_vars(6));
end

% Drawing stuff
function draw_arm_fig()
    global link_vars links H;
    thetas = link_vars(3:4);


    x0 = 0;
    y0 = H;
    x1 = x0 + links(4)*cos(thetas(1));
    y1 = y0 + links(4)*sin(thetas(1));
    x2 = x1 + links(5)*cos(thetas(1)+thetas(2));
    y2 = y1 + links(5)*sin(thetas(1)+thetas(2));

    plot([x0 x1], [y0 y1], 'k');
    hold on;
    plot([x1 x2], [y1 y2], 'k');
    hold off;
end

function draw_iso_fig()
    global T_W T_0 link_vars frame_data;
    global link1_data link2_data link3_data;
    global link4_data;

    hold on;
    T = generate_frames(T_0, link_vars);

    draw_wireframe(frame_data, 'k', 2, T_W);
    draw_wireframe(link1_data, 'k', 2, T(:, :, 1));
    draw_wireframe(link2_data, 'k', 2, T(:, :, 2));
    draw_wireframe(link3_data, 'k', 2, T(:, :, 3));
    draw_wireframe(link4_data, 'k', 2, T(:, :, 4));

    hold off;
end

% Drawing auxillary functions
function draw_wireframe(d, col, ls, T)
    for i=1:size(d, 1)
        p = T*[d(i, 1:2); d(i, 3:4); d(i, 5:6); 1 1];
        plot3(p(1, :), p(2, :), p(3, :), col, 'LineWidth', ls);
    end
end
```

```
function draw_vector(frame, x, y, z, c, thk)
    hold on;
    p = frame*[x; y; z; 1];
    p0 = [frame(1, 4); frame(2, 4); frame(3, 4); 1];
    dp = p - p0;
    quiver3(p0(1), p0(2), p0(3), dp(1), dp(2), dp(3), 0, c, ...
        'LineWidth', thk, 'MaxHeadSize', 0.3);
    hold off;
end

function T = TF(alpha, a, d, theta)
    T = [cos(theta) -sin(theta) 0 a
     sin(theta)*cos(alpha) cos(theta)*cos(alpha) -sin(alpha) -
sin(alpha)*d
     sin(theta)*sin(alpha) cos(theta)*sin(alpha)  cos(alpha)
 cos(alpha)*d
     0 0 0 1];
end

function T=ht(R, Porg)
    T = [R Porg; 0 0 0 1];
end

function R=fixed_angle(g, b, a)
    R = R_Z(a)*R_Y(b)*R_X(g);
end

function R=R_X(t)
    R = [1 0 0; 0 cos(t) -sin(t); 0 sin(t) cos(t)];
end

function R=R_Y(t)
    R = [cos(t) 0 sin(t); 0 1 0; -sin(t) 0 cos(t)];
end

function R=R_Z(t)
    R = [cos(t) -sin(t) 0; sin(t) cos(t) 0; 0 0 1];
end
```

```
Undefined function 'jointspace_contr' for input arguments of type
 'double'.

Error in main_contr (line 8)
        ang_actual{i-1} = jointspace_contr(ang(i-1,:),ang(i,:));

Error in simulation (line 112)
        actual_angle = main_contr(z,cmd{1},links);
```

Published with MATLAB® R2019b